

Charu C. Aggarwal

# Outlier Analysis

*Second Edition*

 Springer

---

# Outlier Analysis

---

Charu C. Aggarwal

# Outlier Analysis

Second Edition

 Springer

Charu C. Aggarwal  
IBM T.J. Watson Research Center  
Yorktown Heights, New York, USA

ISBN 978-3-319-47577-6      ISBN 978-3-319-47578-3 (eBook)  
DOI 10.1007/978-3-319-47578-3

Library of Congress Control Number: 2016961247

© Springer International Publishing AG 2017

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made.

Printed on acid-free paper

This Springer imprint is published by Springer Nature  
The registered company is Springer International Publishing AG  
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

To my wife, my daughter Sayani,  
and my late parents Dr. Prem Sarup and Mrs. Pushplata Aggarwal.

---

---

# Contents

---

---

<b>1</b>	<b>An Introduction to Outlier Analysis</b>	<b>1</b>
1.1	Introduction . . . . .	1
1.2	The Data Model is Everything . . . . .	5
1.2.1	Connections with Supervised Models . . . . .	8
1.3	The Basic Outlier Detection Models . . . . .	10
1.3.1	Feature Selection in Outlier Detection . . . . .	10
1.3.2	Extreme-Value Analysis . . . . .	11
1.3.3	Probabilistic and Statistical Models . . . . .	12
1.3.4	Linear Models . . . . .	13
1.3.4.1	Spectral Models . . . . .	14
1.3.5	Proximity-Based Models . . . . .	14
1.3.6	Information-Theoretic Models . . . . .	16
1.3.7	High-Dimensional Outlier Detection . . . . .	17
1.4	Outlier Ensembles . . . . .	18
1.4.1	Sequential Ensembles . . . . .	19
1.4.2	Independent Ensembles . . . . .	20
1.5	The Basic Data Types for Analysis . . . . .	21
1.5.1	Categorical, Text, and Mixed Attributes . . . . .	21
1.5.2	When the Data Values have Dependencies . . . . .	21
1.5.2.1	Times-Series Data and Data Streams . . . . .	22
1.5.2.2	Discrete Sequences . . . . .	24
1.5.2.3	Spatial Data . . . . .	24
1.5.2.4	Network and Graph Data . . . . .	25
1.6	Supervised Outlier Detection . . . . .	25
1.7	Outlier Evaluation Techniques . . . . .	26
1.7.1	Interpreting the ROC AUC . . . . .	29
1.7.2	Common Mistakes in Benchmarking . . . . .	30
1.8	Conclusions and Summary . . . . .	31
1.9	Bibliographic Survey . . . . .	31
1.10	Exercises . . . . .	33

<b>2</b>	<b>Probabilistic Models for Outlier Detection</b>	<b>35</b>
2.1	Introduction . . . . .	35
2.2	Statistical Methods for Extreme-Value Analysis . . . . .	37
2.2.1	Probabilistic Tail Inequalities . . . . .	37
2.2.1.1	Sum of Bounded Random Variables . . . . .	38
2.2.2	Statistical-Tail Confidence Tests . . . . .	43
2.2.2.1	$t$ -Value Test . . . . .	43
2.2.2.2	Sum of Squares of Deviations . . . . .	45
2.2.2.3	Visualizing Extreme Values with Box Plots . . . . .	45
2.3	Extreme-Value Analysis in Multivariate Data . . . . .	46
2.3.1	Depth-Based Methods . . . . .	47
2.3.2	Deviation-Based Methods . . . . .	48
2.3.3	Angle-Based Outlier Detection . . . . .	49
2.3.4	Distance Distribution-based Techniques: The Mahalanobis Method . . . . .	51
2.3.4.1	Strengths of the Mahalanobis Method . . . . .	53
2.4	Probabilistic Mixture Modeling for Outlier Analysis . . . . .	54
2.4.1	Relationship with Clustering Methods . . . . .	57
2.4.2	The Special Case of a Single Mixture Component . . . . .	58
2.4.3	Other Ways of Leveraging the EM Model . . . . .	58
2.4.4	An Application of EM for Converting Scores to Probabilities . . . . .	59
2.5	Limitations of Probabilistic Modeling . . . . .	60
2.6	Conclusions and Summary . . . . .	61
2.7	Bibliographic Survey . . . . .	61
2.8	Exercises . . . . .	62
<b>3</b>	<b>Linear Models for Outlier Detection</b>	<b>65</b>
3.1	Introduction . . . . .	65
3.2	Linear Regression Models . . . . .	68
3.2.1	Modeling with Dependent Variables . . . . .	70
3.2.1.1	Applications of Dependent Variable Modeling . . . . .	73
3.2.2	Linear Modeling with Mean-Squared Projection Error . . . . .	74
3.3	Principal Component Analysis . . . . .	75
3.3.1	Connections with the Mahalanobis Method . . . . .	78
3.3.2	Hard PCA versus Soft PCA . . . . .	79
3.3.3	Sensitivity to Noise . . . . .	79
3.3.4	Normalization Issues . . . . .	80
3.3.5	Regularization Issues . . . . .	80
3.3.6	Applications to Noise Correction . . . . .	80
3.3.7	How Many Eigenvectors? . . . . .	81
3.3.8	Extension to Nonlinear Data Distributions . . . . .	83
3.3.8.1	Choice of Similarity Matrix . . . . .	85
3.3.8.2	Practical Issues . . . . .	86
3.3.8.3	Application to Arbitrary Data Types . . . . .	88
3.4	One-Class Support Vector Machines . . . . .	88
3.4.1	Solving the Dual Optimization Problem . . . . .	92
3.4.2	Practical Issues . . . . .	92
3.4.3	Connections to Support Vector Data Description and Other Kernel Models . . . . .	93
3.5	A Matrix Factorization View of Linear Models . . . . .	95

3.5.1	Outlier Detection in Incomplete Data . . . . .	96
3.5.1.1	Computing the Outlier Scores . . . . .	98
3.6	Neural Networks: From Linear Models to Deep Learning . . . . .	98
3.6.1	Generalization to Nonlinear Models . . . . .	101
3.6.2	Replicator Neural Networks and Deep Autoencoders . . . . .	102
3.6.3	Practical Issues . . . . .	105
3.6.4	The Broad Potential of Neural Networks . . . . .	106
3.7	Limitations of Linear Modeling . . . . .	106
3.8	Conclusions and Summary . . . . .	107
3.9	Bibliographic Survey . . . . .	108
3.10	Exercises . . . . .	109
<b>4</b>	<b>Proximity-Based Outlier Detection</b> . . . . .	<b>111</b>
4.1	Introduction . . . . .	111
4.2	Clusters and Outliers: The Complementary Relationship . . . . .	112
4.2.1	Extensions to Arbitrarily Shaped Clusters . . . . .	115
4.2.1.1	Application to Arbitrary Data Types . . . . .	118
4.2.2	Advantages and Disadvantages of Clustering Methods . . . . .	118
4.3	Distance-Based Outlier Analysis . . . . .	118
4.3.1	Scoring Outputs for Distance-Based Methods . . . . .	119
4.3.2	Binary Outputs for Distance-Based Methods . . . . .	121
4.3.2.1	Cell-Based Pruning . . . . .	122
4.3.2.2	Sampling-Based Pruning . . . . .	124
4.3.2.3	Index-Based Pruning . . . . .	126
4.3.3	Data-Dependent Similarity Measures . . . . .	128
4.3.4	ODIN: A Reverse Nearest Neighbor Approach . . . . .	129
4.3.5	Intensional Knowledge of Distance-Based Outliers . . . . .	130
4.3.6	Discussion of Distance-Based Methods . . . . .	131
4.4	Density-Based Outliers . . . . .	131
4.4.1	LOF: Local Outlier Factor . . . . .	132
4.4.1.1	Handling Duplicate Points and Stability Issues . . . . .	134
4.4.2	LOCI: Local Correlation Integral . . . . .	135
4.4.2.1	LOCI Plot . . . . .	136
4.4.3	Histogram-Based Techniques . . . . .	137
4.4.4	Kernel Density Estimation . . . . .	138
4.4.4.1	Connection with Harmonic $k$ -Nearest Neighbor Detector . . . . .	139
4.4.4.2	Local Variations of Kernel Methods . . . . .	140
4.4.5	Ensemble-Based Implementations of Histograms and Kernel Methods . . . . .	140
4.5	Limitations of Proximity-Based Detection . . . . .	141
4.6	Conclusions and Summary . . . . .	142
4.7	Bibliographic Survey . . . . .	142
4.8	Exercises . . . . .	146
<b>5</b>	<b>High-Dimensional Outlier Detection</b> . . . . .	<b>149</b>
5.1	Introduction . . . . .	149
5.2	Axis-Parallel Subspaces . . . . .	152
5.2.1	Genetic Algorithms for Outlier Detection . . . . .	153
5.2.1.1	Defining Abnormal Lower-Dimensional Projections . . . . .	153
5.2.1.2	Defining Genetic Operators for Subspace Search . . . . .	154



5.2.2	Finding Distance-Based Outlying Subspaces . . . . .	157
5.2.3	Feature Bagging: A Subspace Sampling Perspective . . . . .	157
5.2.4	Projected Clustering Ensembles . . . . .	158
5.2.5	Subspace Histograms in Linear Time . . . . .	160
5.2.6	Isolation Forests . . . . .	161
5.2.6.1	Further Enhancements for Subspace Selection . . . . .	163
5.2.6.2	Early Termination . . . . .	163
5.2.6.3	Relationship to Clustering Ensembles and Histograms . . . . .	164
5.2.7	Selecting High-Contrast Subspaces . . . . .	164
5.2.8	Local Selection of Subspace Projections . . . . .	166
5.2.9	Distance-Based Reference Sets . . . . .	169
5.3	Generalized Subspaces . . . . .	170
5.3.1	Generalized Projected Clustering Approach . . . . .	171
5.3.2	Leveraging Instance-Specific Reference Sets . . . . .	172
5.3.3	Rotated Subspace Sampling . . . . .	175
5.3.4	Nonlinear Subspaces . . . . .	176
5.3.5	Regression Modeling Techniques . . . . .	178
5.4	Discussion of Subspace Analysis . . . . .	178
5.5	Conclusions and Summary . . . . .	180
5.6	Bibliographic Survey . . . . .	181
5.7	Exercises . . . . .	184
<b>6</b>	<b>Outlier Ensembles</b> . . . . .	<b>185</b>
6.1	Introduction . . . . .	185
6.2	Categorization and Design of Ensemble Methods . . . . .	188
6.2.1	Basic Score Normalization and Combination Methods . . . . .	189
6.3	Theoretical Foundations of Outlier Ensembles . . . . .	191
6.3.1	What is the Expectation Computed Over? . . . . .	195
6.3.2	Relationship of Ensemble Analysis to Bias-Variance Trade-Off . . . . .	195
6.4	Variance Reduction Methods . . . . .	196
6.4.1	Parametric Ensembles . . . . .	197
6.4.2	Randomized Detector Averaging . . . . .	199
6.4.3	Feature Bagging: An Ensemble-Centric Perspective . . . . .	199
6.4.3.1	Connections to Representational Bias . . . . .	200
6.4.3.2	Weaknesses of Feature Bagging . . . . .	202
6.4.4	Rotated Bagging . . . . .	202
6.4.5	Isolation Forests: An Ensemble-Centric View . . . . .	203
6.4.6	Data-Centric Variance Reduction with Sampling . . . . .	205
6.4.6.1	Bagging . . . . .	205
6.4.6.2	Subsampling . . . . .	206
6.4.6.3	Variable Subsampling . . . . .	207
6.4.6.4	Variable Subsampling with Rotated Bagging (VR) . . . . .	209
6.4.7	Other Variance Reduction Methods . . . . .	209
6.5	Flying Blind with Bias Reduction . . . . .	211
6.5.1	Bias Reduction by Data-Centric Pruning . . . . .	211
6.5.2	Bias Reduction by Model-Centric Pruning . . . . .	212
6.5.3	Combining Bias and Variance Reduction . . . . .	213
6.6	Model Combination for Outlier Ensembles . . . . .	214
6.6.1	Combining Scoring Methods with Ranks . . . . .	215

6.6.2	Combining Bias and Variance Reduction . . . . .	216
6.7	Conclusions and Summary . . . . .	217
6.8	Bibliographic Survey . . . . .	217
6.9	Exercises . . . . .	218
<b>7</b>	<b>Supervised Outlier Detection</b>	<b>219</b>
7.1	Introduction . . . . .	219
7.2	Full Supervision: Rare Class Detection . . . . .	221
7.2.1	Cost-Sensitive Learning . . . . .	223
7.2.1.1	MetaCost: A Relabeling Approach . . . . .	223
7.2.1.2	Weighting Methods . . . . .	225
7.2.2	Adaptive Re-sampling . . . . .	228
7.2.2.1	Relationship between Weighting and Sampling . . . . .	229
7.2.2.2	Synthetic Over-sampling: SMOTE . . . . .	229
7.2.3	Boosting Methods . . . . .	230
7.3	Semi-Supervision: Positive and Unlabeled Data . . . . .	231
7.4	Semi-Supervision: Partially Observed Classes . . . . .	232
7.4.1	One-Class Learning with Anomalous Examples . . . . .	233
7.4.2	One-Class Learning with Normal Examples . . . . .	234
7.4.3	Learning with a Subset of Labeled Classes . . . . .	234
7.5	Unsupervised Feature Engineering in Supervised Methods . . . . .	235
7.6	Active Learning . . . . .	236
7.7	Supervised Models for Unsupervised Outlier Detection . . . . .	239
7.7.1	Connections with PCA-Based Methods . . . . .	242
7.7.2	Group-wise Predictions for High-Dimensional Data . . . . .	243
7.7.3	Applicability to Mixed-Attribute Data Sets . . . . .	244
7.7.4	Incorporating Column-wise Knowledge . . . . .	244
7.7.5	Other Classification Methods with Synthetic Outliers . . . . .	244
7.8	Conclusions and Summary . . . . .	245
7.9	Bibliographic Survey . . . . .	245
7.10	Exercises . . . . .	247
<b>8</b>	<b>Categorical, Text, and Mixed Attribute Data</b>	<b>249</b>
8.1	Introduction . . . . .	249
8.2	Extending Probabilistic Models to Categorical Data . . . . .	250
8.2.1	Modeling Mixed Data . . . . .	253
8.3	Extending Linear Models to Categorical and Mixed Data . . . . .	254
8.3.1	Leveraging Supervised Regression Models . . . . .	254
8.4	Extending Proximity Models to Categorical Data . . . . .	255
8.4.1	Aggregate Statistical Similarity . . . . .	256
8.4.2	Contextual Similarity . . . . .	257
8.4.2.1	Connections to Linear Models . . . . .	258
8.4.3	Issues with Mixed Data . . . . .	259
8.4.4	Density-Based Methods . . . . .	259
8.4.5	Clustering Methods . . . . .	259
8.5	Outlier Detection in Binary and Transaction Data . . . . .	260
8.5.1	Subspace Methods . . . . .	260
8.5.2	Novelties in Temporal Transactions . . . . .	262
8.6	Outlier Detection in Text Data . . . . .	262

8.6.1	Probabilistic Models . . . . .	262
8.6.2	Linear Models: Latent Semantic Analysis . . . . .	264
8.6.2.1	Probabilistic Latent Semantic Analysis (PLSA) . . . . .	265
8.6.3	Proximity-Based Models . . . . .	268
8.6.3.1	First Story Detection . . . . .	269
8.7	Conclusions and Summary . . . . .	270
8.8	Bibliographic Survey . . . . .	270
8.9	Exercises . . . . .	272
<b>9</b>	<b>Time Series and Streaming Outlier Detection</b>	<b>273</b>
9.1	Introduction . . . . .	273
9.2	Predictive Outlier Detection in Streaming Time-Series . . . . .	276
9.2.1	Autoregressive Models . . . . .	276
9.2.2	Multiple Time Series Regression Models . . . . .	279
9.2.2.1	Direct Generalization of Autoregressive Models . . . . .	279
9.2.2.2	Time-Series Selection Methods . . . . .	281
9.2.2.3	Principal Component Analysis and Hidden Variable-Based Models . . . . .	282
9.2.3	Relationship between Unsupervised Outlier Detection and Prediction . . . . .	284
9.2.4	Supervised Point Outlier Detection in Time Series . . . . .	284
9.3	Time-Series of Unusual Shapes . . . . .	286
9.3.1	Transformation to Other Representations . . . . .	287
9.3.1.1	Numeric Multidimensional Transformations . . . . .	288
9.3.1.2	Discrete Sequence Transformations . . . . .	290
9.3.1.3	Leveraging Trajectory Representations of Time Series . . . . .	291
9.3.2	Distance-Based Methods . . . . .	293
9.3.2.1	Single Series versus Multiple Series . . . . .	295
9.3.3	Probabilistic Models . . . . .	295
9.3.4	Linear Models . . . . .	295
9.3.4.1	Univariate Series . . . . .	295
9.3.4.2	Multivariate Series . . . . .	296
9.3.4.3	Incorporating Arbitrary Similarity Functions . . . . .	297
9.3.4.4	Leveraging Kernel Methods with Linear Models . . . . .	298
9.3.5	Supervised Methods for Finding Unusual Time-Series Shapes . . . . .	298
9.4	Multidimensional Streaming Outlier Detection . . . . .	298
9.4.1	Individual Data Points as Outliers . . . . .	299
9.4.1.1	Proximity-Based Algorithms . . . . .	299
9.4.1.2	Probabilistic Algorithms . . . . .	301
9.4.1.3	High-Dimensional Scenario . . . . .	301
9.4.2	Aggregate Change Points as Outliers . . . . .	301
9.4.2.1	Velocity Density Estimation Method . . . . .	302
9.4.2.2	Statistically Significant Changes in Aggregate Distributions . . . . .	304
9.4.3	Rare and Novel Class Detection in Multidimensional Data Streams . . . . .	305
9.4.3.1	Detecting Rare Classes . . . . .	305
9.4.3.2	Detecting Novel Classes . . . . .	306
9.4.3.3	Detecting Infrequently Recurring Classes . . . . .	306
9.5	Conclusions and Summary . . . . .	307
9.6	Bibliographic Survey . . . . .	307
9.7	Exercises . . . . .	310

<b>10</b>	<b>Outlier Detection in Discrete Sequences</b>	<b>311</b>
10.1	Introduction . . . . .	311
10.2	Position Outliers . . . . .	313
10.2.1	Rule-Based Models . . . . .	315
10.2.2	Markovian Models . . . . .	316
10.2.3	Efficiency Issues: Probabilistic Suffix Trees . . . . .	318
10.3	Combination Outliers . . . . .	320
10.3.1	A Primitive Model for Combination Outlier Detection . . . . .	322
10.3.1.1	Model-Specific Combination Issues . . . . .	323
10.3.1.2	Easier Special Cases . . . . .	323
10.3.1.3	Relationship between Position and Combination Outliers . . . . .	324
10.3.2	Distance-Based Models . . . . .	324
10.3.2.1	Combining Anomaly Scores from Comparison Units . . . . .	326
10.3.2.2	Some Observations on Distance-Based Methods . . . . .	327
10.3.2.3	Easier Special Case: Short Sequences . . . . .	327
10.3.3	Frequency-Based Models . . . . .	327
10.3.3.1	Frequency-Based Model with User-Specified Comparison Unit . . . . .	327
10.3.3.2	Frequency-Based Model with Extracted Comparison Units . . . . .	328
10.3.3.3	Combining Anomaly Scores from Comparison Units . . . . .	329
10.3.4	Hidden Markov Models . . . . .	329
10.3.4.1	Design Choices in a Hidden Markov Model . . . . .	331
10.3.4.2	Training and Prediction with HMMs . . . . .	333
10.3.4.3	Evaluation: Computing the Fit Probability for Observed Sequences . . . . .	334
10.3.4.4	Explanation: Determining the Most Likely State Sequence for Observed Sequence . . . . .	334
10.3.4.5	Training: Baum-Welch Algorithm . . . . .	335
10.3.4.6	Computing Anomaly Scores . . . . .	336
10.3.4.7	Special Case: Short Sequence Anomaly Detection . . . . .	337
10.3.5	Kernel-Based Methods . . . . .	337
10.4	Complex Sequences and Scenarios . . . . .	338
10.4.1	Multivariate Sequences . . . . .	338
10.4.2	Set-Based Sequences . . . . .	339
10.4.3	Online Applications: Early Anomaly Detection . . . . .	340
10.5	Supervised Outliers in Sequences . . . . .	340
10.6	Conclusions and Summary . . . . .	342
10.7	Bibliographic Survey . . . . .	342
10.8	Exercises . . . . .	344
<b>11</b>	<b>Spatial Outlier Detection</b>	<b>345</b>
11.1	Introduction . . . . .	345
11.2	Spatial Attributes are Contextual . . . . .	349
11.2.1	Neighborhood-Based Algorithms . . . . .	349
11.2.1.1	Multidimensional Methods . . . . .	350
11.2.1.2	Graph-Based Methods . . . . .	351
11.2.1.3	The Case of Multiple Behavioral Attributes . . . . .	351
11.2.2	Autoregressive Models . . . . .	352
11.2.3	Visualization with Variogram Clouds . . . . .	353
11.2.4	Finding Abnormal Shapes in Spatial Data . . . . .	355

11.2.4.1	Contour Extraction Methods . . . . .	356
11.2.4.2	Extracting Multidimensional Representations . . . . .	360
11.2.4.3	Multidimensional Wavelet Transformation . . . . .	360
11.2.4.4	Supervised Shape Discovery . . . . .	360
11.2.4.5	Anomalous Shape Change Detection . . . . .	361
11.3	Spatiotemporal Outliers with Spatial and Temporal Context . . . . .	362
11.4	Spatial Behavior with Temporal Context: Trajectories . . . . .	363
11.4.1	Real-Time Anomaly Detection . . . . .	363
11.4.2	Unusual Trajectory Shapes . . . . .	363
11.4.2.1	Segment-wise Partitioning Methods . . . . .	363
11.4.2.2	Tile-Based Transformations . . . . .	364
11.4.2.3	Similarity-Based Transformations . . . . .	365
11.4.3	Supervised Outliers in Trajectories . . . . .	365
11.5	Conclusions and Summary . . . . .	366
11.6	Bibliographic Survey . . . . .	366
11.7	Exercises . . . . .	367
<b>12</b>	<b>Outlier Detection in Graphs and Networks</b>	<b>369</b>
12.1	Introduction . . . . .	369
12.2	Outlier Detection in Many Small Graphs . . . . .	371
12.2.1	Leveraging Graph Kernels . . . . .	371
12.3	Outlier Detection in a Single Large Graph . . . . .	372
12.3.1	Node Outliers . . . . .	372
12.3.1.1	Leveraging the Mahalanobis Method . . . . .	374
12.3.2	Linkage Outliers . . . . .	374
12.3.2.1	Matrix Factorization Methods . . . . .	374
12.3.2.2	Spectral Methods and Embeddings . . . . .	378
12.3.2.3	Clustering Methods . . . . .	379
12.3.2.4	Community Linkage Outliers . . . . .	380
12.3.3	Subgraph Outliers . . . . .	381
12.4	Node Content in Outlier Analysis . . . . .	382
12.4.1	Shared Matrix Factorization . . . . .	382
12.4.2	Relating Feature Similarity to Tie Strength . . . . .	383
12.4.3	Heterogeneous Markov Random Fields . . . . .	384
12.5	Change-Based Outliers in Temporal Graphs . . . . .	384
12.5.1	Discovering Node Hotspots in Graph Streams . . . . .	385
12.5.2	Streaming Detection of Linkage Anomalies . . . . .	386
12.5.3	Outliers Based on Community Evolution . . . . .	388
12.5.3.1	Integrating Clustering Maintenance with Evolution Analysis . . . . .	388
12.5.3.2	Online Analysis of Community Evolution in Graph Streams . . . . .	390
12.5.3.3	GraphScope . . . . .	390
12.5.4	Outliers Based on Shortest Path Distance Changes . . . . .	392
12.5.5	Matrix Factorization and Latent Embedding Methods . . . . .	392
12.6	Conclusions and Summary . . . . .	393
12.7	Bibliographic Survey . . . . .	394
12.8	Exercises . . . . .	396

<b>13 Applications of Outlier Analysis</b>	<b>399</b>
13.1 Introduction . . . . .	399
13.2 Quality Control and Fault Detection Applications . . . . .	401
13.3 Financial Applications . . . . .	404
13.4 Web Log Analytics . . . . .	406
13.5 Intrusion and Security Applications . . . . .	407
13.6 Medical Applications . . . . .	410
13.7 Text and Social Media Applications . . . . .	411
13.8 Earth Science Applications . . . . .	413
13.9 Miscellaneous Applications . . . . .	415
13.10 Guidelines for the Practitioner . . . . .	416
13.10.1 Which Unsupervised Algorithms Work Best? . . . . .	418
13.11 Resources for the Practitioner . . . . .	421
13.12 Conclusions and Summary . . . . .	422

---

---

# Preface

---

---

“All things excellent are as difficult as they are rare.”– Baruch Spinoza

## First Edition

---

Most of the earliest work on outlier detection was performed by the statistics community. While statistical methods are mathematically more precise, they have several shortcomings, such as simplified assumptions about data representations, poor algorithmic scalability, and a low focus on interpretability. With the increasing advances in hardware technology for *data collection*, and advances in software technology (databases) for *data organization*, computer scientists have increasingly been participating in the latest advancements of this field. Computer scientists approach this field based on their practical experiences in managing large amounts of data, and with far fewer assumptions– the data can be of any type, structured or unstructured, and may be extremely large. Furthermore, issues such as computational efficiency and intuitive analysis of the data are generally considered more important by computer scientists than mathematical precision, though the latter is important as well. This is the approach of professionals from the field of data mining, an area of computer science that was founded about 20 years ago. This has led to the formation of multiple academic communities on the subject, which have remained separated, partially because of differences in technical style and opinions about the importance of different problems and approaches to the subject. At this point, data mining professionals (with a computer science background) are much more actively involved in this area as compared to statisticians. This seems to be a major change in the research landscape. This book presents outlier detection from an integrated perspective, though the focus is towards computer science professionals. Special emphasis was placed on relating the methods from different communities with one another.

The key advantage of writing the book at this point in time is that the vast amount of work done by computer science professionals in the last two decades has remained largely untouched by a formal book on the subject. The classical books relevant to outlier analysis are as follows:

- P. Rousseeuw and A. Leroy. Robust Regression and Outlier Detection, *Wiley*, 2003.
- V. Barnett and T. Lewis. Outliers in Statistical Data, *Wiley*, 1994.
- D. Hawkins. Identification of Outliers, *Chapman and Hall*, 1980.

We note that these books are quite outdated, and the most recent among them is a decade old. Furthermore, this (most recent) book is really focused on the relationship between regression and outlier analysis, rather than the latter. Outlier analysis is a much broader area, in which regression analysis is only a small part. The other books are even older, and are between 15 and 25 years old. They are exclusively targeted to the statistics community. This is not surprising, given that the first mainstream computer science conference in data mining (KDD) was organized in 1995. Most of the work in the data-mining community was performed after the writing of these books. Therefore, many key topics of interest to the broader data mining community are not covered in these books. Given that outlier analysis has been explored by a much broader community, including databases, data mining, statistics, and machine learning, we feel that our book incorporates perspectives from a much broader audience and brings together different points of view.

The chapters of this book have been organized carefully, with a view of covering the area extensively in a natural order. Emphasis was placed on simplifying the content, so that students and practitioners can also benefit from the book. While we did not originally intend to create a textbook on the subject, it evolved during the writing process into a work that can also be used as a teaching aid. Furthermore, it can also be used as a reference book, since each chapter contains extensive bibliographic notes. Therefore, this book serves a dual purpose, providing a comprehensive exposition of the topic of outlier detection from multiple points of view.

## **Additional Notes for the Second Edition**

---

The second edition of this book is a significant enhancement over the first edition. In particular, most of the chapters have been upgraded with new material and recent techniques. More explanations have been added at several places and newer techniques have also been added. An entire chapter on outlier ensembles has been added. Many new topics have been added to the book such as feature selection, one-class support vector machines, one-class neural networks, matrix factorization, spectral methods, wavelet transforms, and supervised learning. Every chapter has been updated with the latest algorithms on the topic.

Last but not least, the first edition was classified by the publisher as a monograph, whereas the second edition is formally classified as a textbook. The writing style has been enhanced to be easily understandable to students. Many algorithms have been described in greater detail, as one might expect from a textbook. It is also accompanied with a solution manual for classroom teaching.



---

---

# Acknowledgments

---

---

## First Edition

---

I would like to thank my wife and daughter for their love and support during the writing of this book. The writing of a book requires significant time that is taken away from family members. This book is the result of their patience with me during this time. I also owe my late parents a debt of gratitude for instilling in me a love of education, which has played an important inspirational role in my book-writing efforts.

I would also like to thank my manager Nagui Halim for providing the tremendous support necessary for the writing of this book. His professional support has been instrumental for my many book efforts in the past and present.

Over the years, I have benefited from the insights of numerous collaborators. An incomplete list of these long-term collaborators in alphabetical order is Tarek F. Abdelzaher, Jiawei Han, Thomas S. Huang, Latifur Khan, Mohammad M. Masud, Spiros Papadimitriou, Guojun Qi, and Philip S. Yu. I would like to thank them for their collaborations and insights over the course of many years.

I would also like to specially thank my advisor James B. Orlin for his guidance during my early years as a researcher. While I no longer work in the same area, the legacy of what I learned from him is a crucial part of my approach to research. In particular, he taught me the importance of intuition and simplicity of thought in the research process. These are more important aspects of research than is generally recognized. This book is written in a simple and intuitive style, and is meant to improve accessibility of this area to both researchers and practitioners.

Finally, I would like to thank Lata Aggarwal for helping me with some of the figures created using PowerPoint graphics in this book.

## Acknowledgments for Second Edition

---

I received significant feedback from various colleagues during the writing of the second edition. In particular, I would like to acknowledge Leman Akoglu, Chih-Jen Lin, Saket Sathe, Jiliang Tang, and Suhang Wang. Leman and Saket provided detailed feedback on several sections and chapters of this book.

---

---

# Author Biography

---

---

**Charu C. Aggarwal** is a Distinguished Research Staff Member (DRSM) at the IBM T. J. Watson Research Center in Yorktown Heights, New York. He completed his undergraduate degree in Computer Science from the Indian Institute of Technology at Kanpur in 1993 and his Ph.D. from the Massachusetts Institute of Technology in 1996.



He has worked extensively in the field of data mining. He has published more than 300 papers in refereed conferences and journals and authored over 80 patents. He is the author or editor of 15 books, including a textbook on data mining and a comprehensive book on outlier analysis. Because of the commercial value of his patents, he has thrice been designated a Master Inventor at IBM. He is a recipient of an IBM Corporate Award (2003) for his work on bio-terrorist threat detection in data streams, a recipient of the IBM Outstanding Innovation Award (2008) for his scientific contributions to privacy technology, a recipient of two IBM Outstanding Technical Achievement Awards (2009, 2015) for his work on data streams and high-dimensional data, respectively. He received the EDBT 2014 Test of Time Award for his work on condensation-based privacy-preserving data mining. He is also a recipient of the IEEE ICDM Research Contributions Award (2015), which is one of the two highest awards for influential research contributions in the field of data mining.

He has served as the general co-chair of the IEEE Big Data Conference (2014) and as the program co-chair of the ACM CIKM Conference (2015), the IEEE ICDM Conference (2015), and the ACM KDD Conference (2016). He served as an associate editor of the IEEE Transactions on Knowledge and Data Engineering from 2004 to 2008. He is an associate editor of the ACM Transactions on Knowledge Discovery from Data, an associate editor of the IEEE Transactions on Big Data, an action editor of the Data Mining and Knowledge Discovery Journal, editor-in-chief of the ACM SIGKDD Explorations, and an associate editor of the Knowledge and Information Systems Journal. He serves on the advisory board of the Lecture Notes on Social Networks, a publication by Springer. He has served as the vice-president of the SIAM Activity Group on Data Mining and is a member of the SIAM industry committee. He is a fellow of the SIAM, ACM, and the IEEE, for “contributions to knowledge discovery and data mining algorithms.”

---

---

# Chapter 1

## An Introduction to Outlier Analysis

---

---

“Never take the comment that you are different as a condemnation, it might be a compliment. It might mean that you possess unique qualities that, like the most rarest of diamonds is . . . one of a kind.” – Eugene Nathaniel Butler

### 1.1 Introduction

---

An outlier is a data point that is significantly different from the remaining data. Hawkins defined [249] an outlier as follows:

“An outlier is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism.”

Outliers are also referred to as *abnormalities*, *discordants*, *deviants*, or *anomalies* in the data mining and statistics literature. In most applications, the data is created by one or more generating processes, which could either reflect activity in the system or observations collected about entities. When the generating process behaves unusually, it results in the creation of outliers. Therefore, an outlier often contains useful information about abnormal characteristics of the systems and entities that impact the data generation process. The recognition of such unusual characteristics provides useful application-specific insights. Some examples are as follows:

- **Intrusion detection systems:** In many computer systems, different types of data are collected about the operating system calls, network traffic, or other user actions. This data may show unusual behavior because of malicious activity. The recognition of such activity is referred to as intrusion detection.
- **Credit-card fraud:** Credit-card fraud has become increasingly prevalent because of greater ease with which sensitive information such as a credit-card number can be compromised. In many cases, unauthorized use of a credit card may show different patterns, such as buying sprees from particular locations or very large transactions. Such patterns can be used to detect outliers in credit-card transaction data.

- **Interesting sensor events:** Sensors are often used to track various environmental and location parameters in many real-world applications. Sudden changes in the underlying patterns may represent events of interest. Event detection is one of the primary motivating applications in the field of sensor networks. As discussed later in this book, event detection is an important *temporal* version of outlier detection.
- **Medical diagnosis:** In many medical applications, the data is collected from a variety of devices such as magnetic resonance imaging (MRI) scans, positron emission tomography (PET) scans or electrocardiogram (ECG) time-series. Unusual patterns in such data typically reflect disease conditions.
- **Law enforcement:** Outlier detection finds numerous applications in law enforcement, especially in cases where unusual patterns can only be discovered over time through multiple actions of an entity. Determining fraud in financial transactions, trading activity, or insurance claims typically requires the identification of unusual patterns in the data generated by the actions of the criminal entity.
- **Earth science:** A significant amount of spatiotemporal data about weather patterns, climate changes, or land-cover patterns is collected through a variety of mechanisms such as satellites or remote sensing. Anomalies in such data provide significant insights about human activities or environmental trends that may be the underlying causes.

In all these applications, the data has a “normal” model, and anomalies are recognized as deviations from this normal model. Normal data points are sometimes also referred to as *inliers*. In some applications such as intrusion or fraud detection, outliers correspond to *sequences* of multiple data points rather than individual data points. For example, a fraud event may often reflect the actions of an individual in a particular sequence. The specificity of the sequence is relevant to identifying the anomalous event. Such anomalies are also referred to as *collective anomalies*, because they can only be inferred collectively from a set or sequence of data points. Such collective anomalies are often a result of unusual *events* that generate anomalous patterns of activity. This book will address these different types of anomalies.

The output of an outlier detection algorithm can be one of two types:

- **Outlier scores:** Most outlier detection algorithms output a score quantifying the level of “outlierness” of each data point. This score can also be used to rank the data points in order of their outlier tendency. This is a very general form of output, which retains all the information provided by a particular algorithm, but it does not provide a concise summary of the small number of data points that should be considered outliers.
- **Binary labels:** A second type of output is a binary label indicating whether a data point is an outlier or not. Although some algorithms might directly return binary labels, outlier scores can also be converted into binary labels. This is typically achieved by imposing thresholds on outlier scores, and the threshold is chosen based on the statistical distribution of the scores. A binary labeling contains less information than a scoring mechanism, but it is the final result that is often needed for decision making in practical applications.

It is often a subjective judgement, as to what constitutes a “sufficient” deviation for a point to be considered an outlier. In real applications, the data may be embedded in a

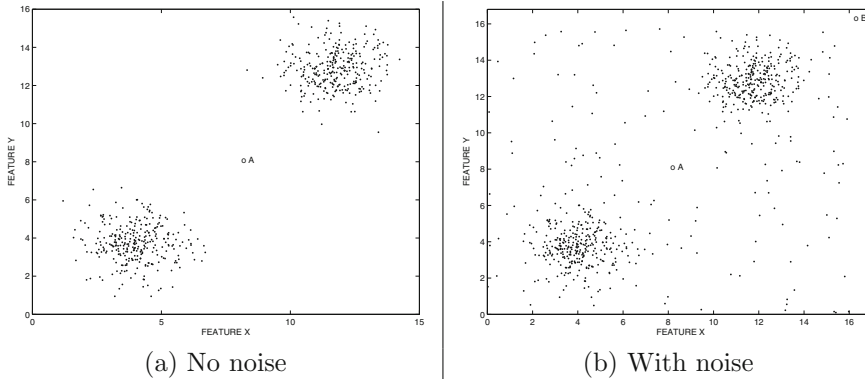


Figure 1.1: The difference between noise and anomalies

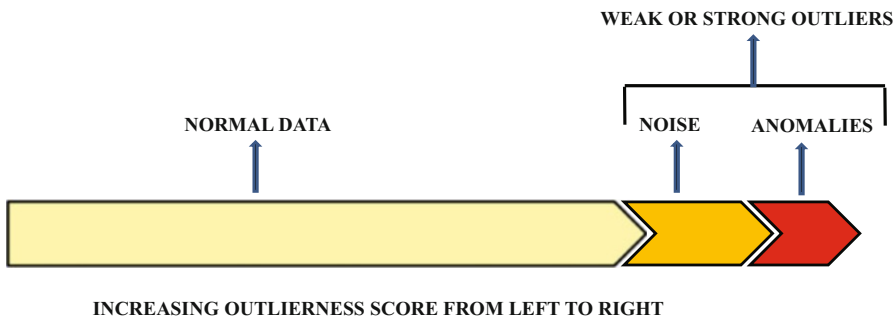


Figure 1.2: The spectrum from normal data to outliers

significant amount of noise, and such noise may not be of any interest to the analyst. It is usually the *significantly interesting deviations* that are of interest. In order to illustrate this point, consider the examples illustrated in Figures 1.1(a) and (b). It is evident that the main patterns (or clusters) in the data are identical in both cases, although there are significant differences outside these main clusters. In the case of Figure 1.1(a), a single data point (marked by ‘A’) seems to be very different from the remaining data, and is therefore very obviously an anomaly. The situation in Figure 1.1(b) is much more subjective. While the corresponding data point ‘A’ in Figure 1.1(b) is also in a sparse region of the data, it is much harder to state confidently that it represents a true deviation from the remaining data set. It is quite likely that this data point represents randomly distributed noise in the data. This is because the point ‘A’ seems to fit a pattern represented by other randomly distributed points. Therefore, throughout this book, the term “outlier” refers to a data point that could either be considered an abnormality or noise, whereas an “anomaly” refers to a special kind of outlier that is of interest to an analyst.

In the *unsupervised scenario*, where previous examples of interesting anomalies are not available, the noise represents the semantic boundary between normal data and true anomalies—noise is often modeled as a weak form of outliers that does not always meet the strong criteria necessary for a data point to be considered interesting or anomalous enough. For example, data points at the boundaries of clusters may often be considered noise. Typ-

ically, most outlier detection algorithms use some quantified measure of the *outlierness* of a data point, such as the sparsity of the underlying region, nearest neighbor based distance, or the fit to the underlying data distribution. Every data point lies on a continuous spectrum from normal data to noise, and finally to anomalies, as illustrated in Figure 1.2. The separation of the different regions of this spectrum is often not precisely defined, and is chosen on an *ad hoc* basis according to application-specific criteria. Furthermore, the separation between noise and anomalies is not pure, and many data points created by a noisy generative process may be deviant enough to be interpreted as anomalies on the basis of the outlier score. Thus, anomalies will *typically* have a much higher outlier score than noise, but this is not a distinguishing factor between the two as a matter of *definition*. Rather, it is the interest of the analyst that regulates the distinction between noise and an anomaly.

Some authors use the terms *weak outliers* and *strong outliers* in order to distinguish between noise and anomalies [4, 318]. The detection of noise in the data has numerous applications of its own. For example, the removal of noise creates a much cleaner data set, which can be utilized for other data mining algorithms. Although noise might not be interesting in its own right, its *removal and identification* continues to be an important problem for mining purposes. Therefore, both noise and anomaly detection problems are important enough to be addressed in this book. Throughout this book, methods specifically relevant to *either* anomaly detection or noise removal will be identified. However, the bulk of the outlier detection algorithms could be used for either problem, since the difference between them is really one of semantics.

Since the semantic distinction between noise and anomalies is based on analyst interest, the best way to find such anomalies and distinguish them from noise is to use the feedback from *previously known outlier examples of interest*. This is quite often the case in many applications, such as credit-card fraud detection, where previous examples of interesting anomalies may be available. These may be used in order to learn *a model that distinguishes the normal patterns from the abnormal data*. Supervised outlier detection techniques are typically much more effective in many application-specific scenarios, because the characteristics of the previous examples can be used to sharpen the search process towards more relevant outliers. This is important, because outliers can be defined in numerous ways in a given data set, most of which may not be interesting. For example, in Figures 1.1(a) and (b), previous examples may suggest that only records with unusually high values of both attributes should be considered anomalies. In such a case, the point ‘A’ in *both* figures should be regarded as noise, and the point ‘B’ in Figure 1.1(b) should be considered an anomaly instead! The crucial point to understand here is that anomalies need to be *unusual in an interesting way*, and the supervision process re-defines what one might find interesting. Generally, unsupervised methods can be used either for noise removal or anomaly detection, and supervised methods are designed for application-specific anomaly detection. Unsupervised methods are often used in an exploratory setting, where the discovered outliers are provided to the analyst for further examination of their application-specific importance.

Several levels of supervision are possible in practical scenarios. In the fully supervised scenario, examples of both normal and abnormal data are available that can be clearly distinguished. In some cases, examples of outliers are available, but the examples of “normal” data may also contain outliers in some (unknown) proportion. This is referred to as classification with positive and unlabeled data. In other semi-supervised scenarios, only examples of normal data or only examples of anomalous data may be available. Thus, the number of variations of the problem is rather large, each of which requires a related but dedicated set of techniques.

Finally, the data representation may vary widely across applications. For example, the data may be purely multidimensional with no relationships among points, or the data may be sequential with temporal ordering, or may be defined in the form of a network with arbitrary relationships among data points. Furthermore, the attributes in the data may be numerical, categorical, or mixed. Clearly, the outlier detection process needs to be sensitive to the nature of the attributes and relationships in the underlying data. In fact, the relationships themselves may often provide an outlier-detection criterion in the form of connections between entities that do not usually occur together. Such outliers are referred to as *contextual* outliers. A classical example of this is the concept of *linkage outliers* in social network analysis [17]. In this case, entities (nodes) in the graph that are normally not connected together may show *anomalous* connections with each other. Thus, the impact of data types on the anomaly detection process is significant and will be carefully addressed in this book.

This chapter is organized as follows. In section 1.2, the importance of data modeling in outlier analysis is discussed. In section 1.3, the basic outlier models for outlier detection are introduced. Outlier ensembles are introduced in section 1.4. Section 1.5 discusses the basic data types used for analysis. Section 1.6 introduces the concept of supervised modeling of outliers for data analysis. Methods for evaluating outlier detection algorithms are discussed in section 1.7. The conclusions are presented in section 1.8.

## 1.2 The Data Model is Everything

---

Virtually all outlier detection algorithms create a model of the normal patterns in the data, and then compute an outlier score of a given data point on the basis of the deviations from these patterns. For example, this data model may be a generative model such as a Gaussian-mixture model, a regression-based model, or a proximity-based model. All these models make different assumptions about the “normal” behavior of the data. The outlier score of a data point is then computed by evaluating the quality of the fit between the data point and the model. In many cases, the model may be algorithmically defined. For example, nearest neighbor-based outlier detection algorithms model the outlier tendency of a data point in terms of the distribution of its  $k$ -nearest neighbor distance. Thus, in this case, the assumption is that outliers are located at large distances from most of the data.

Clearly, the choice of the data model is crucial. An incorrect choice of data model may lead to poor results. For example, a fully generative model such as the Gaussian mixture model may not work well, if the data does not fit the generative assumptions of the model, or if a sufficient number of data points are not available to learn the parameters of the model. Similarly, a linear regression-based model may work poorly, if the underlying data is clustered arbitrarily. In such cases, data points may be incorrectly reported as outliers *because of poor fit to the erroneous assumptions of the model*. Unfortunately, outlier detection is largely an *unsupervised* problem in which examples of outliers are not available to learn<sup>1</sup> the best model (in an automated way) for a particular data set. This aspect of outlier detection tends to make it more challenging than many other *supervised* data mining problems like classification in which *labeled* examples are available. Therefore, in practice, the choice of the model is often dictated by the analyst’s understanding of the kinds of deviations relevant to an application. For example, in a spatial application measuring a behavioral attribute such as the location-specific temperature, it would be reasonable to assume that an unusual deviation of the temperature attribute in a spatial locality is an

---

<sup>1</sup>In supervised problems like classification, this process is referred to as *model selection*.

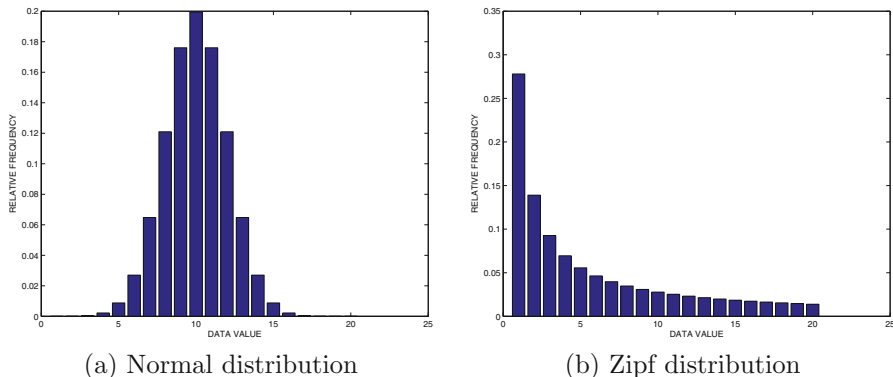


Figure 1.3: Applying  $Z$ -value test on the Normal and Zipf distributions

indicator of abnormality. On the other hand, for the case of high-dimensional data, even the definition of data locality may be ill-defined because of data sparsity. Thus, an effective model for a particular data domain may only be constructed after carefully evaluating the relevant modeling properties of that domain.

In order to understand the impact of the model, it is instructive to examine the use of a simple model known as the  $Z$ -value test for outlier analysis. Consider a set of 1-dimensional quantitative data observations, denoted by  $X_1 \dots X_N$ , with mean  $\mu$  and standard deviation  $\sigma$ . The  $Z$ -value for the data point  $X_i$  is denoted by  $Z_i$  and is defined as follows:

$$Z_i = \frac{|X_i - \mu|}{\sigma} \quad (1.1)$$

The  $Z$ -value test computes the number of standard deviations by which a data point is distant from the mean. This provides a good proxy for the outlier score of that point. An implicit assumption is that the data is modeled from a normal distribution, and therefore the  $Z$ -value is a random variable drawn from a *standard* normal distribution with zero mean and unit variance. In cases where the mean and standard deviation of the distribution can be accurately estimated, a good “rule-of-thumb” is to use  $Z_i \geq 3$  as a proxy for the anomaly. However, in scenarios in which very few samples are available, the mean and standard deviation of the underlying distribution cannot be estimated robustly. In such cases, the results from the  $Z$ -value test need to be interpreted more carefully with the use of the (related) *Student’s  $t$ -distribution* rather than a normal distribution. This issue will be discussed in Chapter 2.

It is often forgotten by practitioners during modeling that the  $Z$ -value test implicitly assumes a normal distribution for the underlying data. When such an approximation is poor, the results are harder to interpret. For example, consider the two data frequency histograms drawn on values between 1 and 20 in Figure 1.3. In the first case, the histogram is sampled from a normal distribution with  $(\mu, \sigma) = (10, 2)$ , and in the second case, it is sampled from a Zipf distribution  $1/i$ . It is evident that most of the data lies in the range  $[10 - 2*3, 10 + 2*3]$  for the normal distribution, and all data points lying outside this range can be truly considered anomalies. Thus, the  $Z$ -value test works very well in this case. In the second case with the Zipf distribution, the anomalies are not quite as clear, although the data points with very high values (such as 20) can probably be considered anomalies. In this case, the mean and standard deviation of the data are 5.24 and 5.56, respectively. As a result, the  $Z$ -value test does not declare *any* of the data points as anomalous (for a



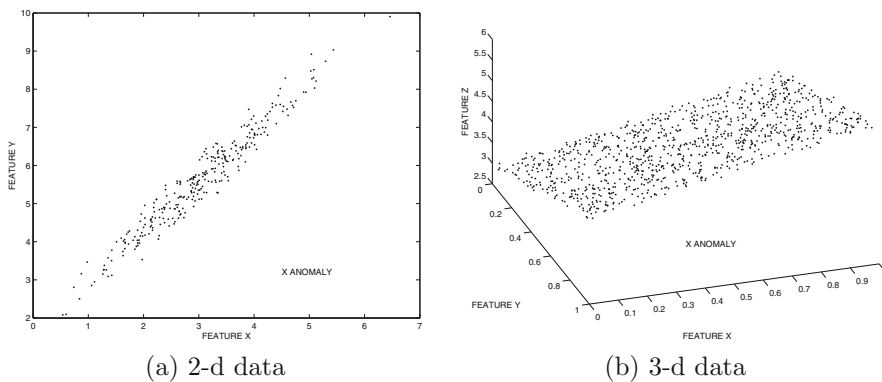


Figure 1.4: Linearly Correlated Data

threshold of 3), although it does come close. In any case, the significance of the  $Z$ -value from the Zipf-distribution is not very meaningful at least from the perspective of probabilistic interpretability. This suggests that if mistakes are made at the modeling stage, it can result in an incorrect understanding of the data. Such tests are often used as a *heuristic* to provide a rough idea of the outlier scores even for data sets that are far from normally distributed, and it is important to interpret such scores carefully.

An example in which the  $Z$ -value test would not work even as a heuristic, would be one in which it was applied to a data point that was an outlier only because of its relative position, rather than its extreme position. For example, if the  $Z$ -value test is applied to an individual dimension in Figure 1.1(a), the test would fail miserably, because point ‘A’ would be considered the most centrally located and normal data point. On the other hand, the test can still be reasonably applied to a set of *extracted* 1-dimensional values corresponding to the  $k$ -nearest neighbor distances of each point. Therefore, the effectiveness of a model depends both on the choice of the test used, and *how* it is applied.

The best choice of a model is often data-specific. This requires a good understanding of the data itself before choosing the model. For example, a regression-based model would be most suitable for finding the outliers in the data distributions of Figure 1.4, where most of the data is distributed along linear correlation planes. On the other hand, a clustering model would be more suitable for the cases illustrated in Figure 1.1. An poor choice of model for a given data set is likely to provide poor results. *Therefore, the core principle of discovering outliers is based on assumptions about the structure of the normal patterns in a given data set. Clearly, the choice of the “normal” model depends highly on the analyst’s understanding of the natural data patterns in that particular domain.* This implies that it is often useful for the analyst to have a semantic understanding of the data representation, although this is often not possible in real settings.

There are many trade-offs associated with model choice; a highly complex model with too many parameters will most likely overfit the data, and will also find a way to fit the outliers. A simple model, which is constructed with a good intuitive understanding of the data (and possibly also an understanding of what the analyst is looking for), is likely to lead to much better results. On the other hand, an oversimplified model, which fits the data poorly, is likely to declare normal patterns as outliers. The initial stage of selecting the data model is perhaps the most crucial one in outlier analysis. The theme about the impact of data models will be repeated throughout the book, with specific examples.

### 1.2.1 Connections with Supervised Models

One can view the outlier detection problem as a variant of the classification problem in which the class label (“normal” or “anomaly”) is unobserved. Therefore, by virtue of the fact that normal examples far outnumber the anomalous examples, one can “pretend” that the entire data set contains the normal class and create a (possibly noisy) model of the normal data. Deviations from the normal model are treated as outlier scores. This connection between classification and outlier detection is important because much of the theory and methods from classification generalize to outlier detection [32]. The unobserved nature of the labels (or outlier scores) is the reason that outlier detection methods are referred to as *unsupervised* whereas classification methods are referred to as *supervised*. In cases where the anomaly labels are observed, the problem simplifies to the imbalanced version of data classification, and it is discussed in detail in Chapter 7.

The model of normal data for unsupervised outlier detection may be considered a *one-class analog* of the multi-class setting in classification. However, the one-class setting is sometimes far more subtle from a modeling perspective, because it is much easier to distinguish between examples of two classes than to predict whether a particular instance matches examples of a single (normal) class. When at least two classes are available, the *distinguishing* characteristics between the two classes can be learned more easily in order to sharpen the accuracy of the model.

In many forms of predictive learning, such as classification and recommendation, there is a natural dichotomy between *instance-based learning methods* and *explicit generalization methods*. Since outlier detection methods require the design of a model of the normal data in order to make predictions, this dichotomy applies to the unsupervised domain as well. In instance-based methods, a training model is not constructed up front. Rather, for a given test instance, one computes the most relevant (i.e., closest) instances of the training data, and makes predictions on the test instance using these related instances. Instance-based methods are also referred to as *lazy learners* in the field of classification [33] and *memory-based methods* in the field of recommender systems [34].

A simple example of an instance-based learning method in outlier analysis is the use of the 1-nearest-neighbor distance of a data point as its outlier score. Note that this approach does not require the construction of a training model up front because all the work of determining the nearest neighbor is done after specifying the identity of the instance to be predicted (scored). The 1-nearest neighbor outlier detector can be considered the unsupervised analog of the 1-nearest neighbor classifier in the supervised domain. Instance-based models are extremely popular in the outlier analysis domain because of their simplicity, effectiveness, and intuitive nature. In fact, many of the most popular and successful methods for outlier detection, such as  $k$ -nearest neighbor detectors [58, 456] and Local Outlier Factor (LOF) [96] (cf. Chapter 4), are instance-based methods.

The popularity of instance-based methods is so great in the outlier analysis community that the vast array of one-class analogs of other supervised methods are often overlooked. In principle, almost any classification method can be re-designed to create a one-class analog. Most of these methods are *explicit generalization methods*, in which a summarized model needs to be created up front. Explicit generalization methods use a two-step process on the data set  $\mathcal{D}$ :

1. Create a one-class model of the normal data using the original data set  $\mathcal{D}$ . For example, one might learn a linear hyperplane describing the normal data in Figure 1.4(b). This hyperplane represents a *summarized model* of the entire data set and therefore represents an explicit generalization of the data set.

Table 1.1: Classification methods and their unsupervised analogs in outlier analysis

Supervised Model	Unsupervised Analog(s)	Type
$k$ -nearest neighbor	$k$ -NN distance, LOF, LOCI (Chapter 4)	Instance-based
Linear Regression	Principal Component Analysis (Chapter 3)	Explicit Generalization
Naive Bayes	Expectation-maximization (Chapter 2)	Explicit Generalization
Rocchio	Mahalanobis method (Chapter 3) Clustering (Chapter 4)	Explicit Generalization
Decision Trees Random Forests	Isolation Trees Isolation Forests (Chapters 5 and 6)	Explicit generalization
Rule-based	FP-Outlier (Chapter 8)	Explicit Generalization
Support-vector machines	One-class support-vector machines (Chapter 3)	Explicit generalization
Neural Networks	Replicator neural networks (Chapter 3)	Explicit generalization
Matrix factorization (incomplete data prediction)	Principal component analysis Matrix factorization (Chapter 3)	Explicit generalization

- Score each point in  $\mathcal{D}$  based on its deviation from this model of normal data. For example, if we learn a linear hyperplane using the data set of Figure 1.4(b) in the first step, then we might report the Euclidean distance from this hyperplane as the outlier score.

One problem with explicit generalization methods is that the same data set  $\mathcal{D}$  is used for both training and scoring. This is because it is hard to exclude a specific test point during the scoring process (like instance-based methods). Furthermore, unlike classification in which the presence or absence of ground-truth (labeling) naturally partitions the data into training and test portions, there is no labeling available in unsupervised problems. Therefore, one typically wants to use the entire data set  $\mathcal{D}$  for both training and testing in unsupervised problems, which causes overfitting. Nevertheless, the influence of individual points on overfitting is often small in real-world settings because explicit generalization methods tend to create a concise summary (i.e., *generalized* representation) of a much larger data set. Since the same data  $\mathcal{D}$  is used for training and testing, one can view outlier scores as the training data errors made in the assumption of “pretending” that all training data points belong to the normal class. Often an effective approach to reduce overfitting is to repeatedly partition the data into training and test sets in a randomized way and average the outlier scores of test points from the various models. Such methods will be discussed in later chapters.

Virtually all classification models can be generalized to outlier detection by using an appropriate one-class analog. Examples of such models include linear regression models, principal component analysis, probabilistic expectation-maximization models, clustering methods, one-class support vector machines, matrix factorization models, and one-class

neural networks. For the reader who has a familiarity with the classification problem, we have listed various classification models and their corresponding one-class analogs for outlier detection in Table 1.1. The table is not comprehensive and is intended to provide intuition about the connections between the supervised and unsupervised settings with representative examples. The connections between supervised and unsupervised learning are very deep; in section 7.7 of Chapter 7, we point out yet another useful connection between outlier detection and regression modeling. This particular connection has the merit that it enables the use of hundreds of off-the-shelf regression models for unsupervised outlier detection with an almost trivial implementation.

## 1.3 The Basic Outlier Detection Models

---

This section will present an overview of the most important models in the literature, and also provide some idea of the settings in which they might work well. A detailed discussion of these methods are provided in later chapters. Several factors influence the choice of an outlier model, including the data type, data size, availability of relevant outlier examples, and the need for interpretability in a model. The last of these criteria merits further explanation.

The *interpretability* of an outlier detection model is extremely important from the perspective of the analyst. It is often desirable to determine *why* a particular data point should be considered an outlier because it provides the analyst further hints about the diagnosis required in an application-specific scenario. This process is also referred to as that of discovering the *intensional knowledge* about the outliers [318] or that of *outlier detection and description* [44]. Different models have different levels of interpretability. Typically, models that work with the original attributes and use fewer transforms on the data (e.g., principal component analysis) have higher interpretability. The trade-off is that data transformations often enhance the contrast between the outliers and normal data points at the expense of interpretability. Therefore, it is critical to keep these factors in mind while choosing a specific model for outlier analysis.

### 1.3.1 Feature Selection in Outlier Detection

It is notoriously difficult to perform feature selection in outlier detection because of the unsupervised nature of the outlier detection problem. Unlike classification, in which labels can be used as guiding posts, it is difficult to learn how features relate to the (unobserved) ground truth in unsupervised outlier detection. Nevertheless, a common way of measuring the non-uniformity of a set of univariate points  $x_1 \dots x_N$  is the *Kurtosis measure*. The first step is to compute the mean  $\mu$  and standard deviation  $\sigma$  of this set of values and standardize the data to zero mean and unit variance as follows:

$$z_i = \frac{x_i - \mu}{\sigma} \quad (1.2)$$

Note that the mean value of the *squares* of  $z_i$  is always 1 because of how  $z_i$  is defined. The Kurtosis measure computes the mean value of the *fourth* power of  $z_i$ :

$$K(z_1 \dots z_N) = \frac{\sum_{i=1}^N z_i^4}{N} \quad (1.3)$$

Feature distributions that are very non-uniform show a high level of Kurtosis. For example, when the data contains a few extreme values, the Kurtosis measure will increase because

of the use of the fourth power. Kurtosis measures are often used [367] in the context of *subspace outlier detection methods* (see Chapter 5), in which outliers are explored in lower-dimensional projections of the data.

One problem with the Kurtosis measure is that it does not use the *interactions* between various attributes well, when it analyzes the features individually. It is also possible to use the Kurtosis measure on lower-dimensional distance distributions. For example, one can compute the Kurtosis measure on the set of  $N$  Mahalanobis distances of all data points to the centroid of the data after the data has been projected into a lower-dimensional subspace  $S$ . Such a computation provides the *multidimensional* Kurtosis of that subspace  $S$ , while taking into account the interactions between the various dimensions of  $S$ . The Mahalanobis distance is introduced in Chapter 2. One can combine this computation with a greedy method of iteratively adding features to a candidate subset  $S$  of features in order to construct a discriminative subset of dimensions with the highest multidimensional Kurtosis.

A second methodology for feature selection [429] is to use the connections of the outlier detection problem to supervised learning. The basic idea is that features that are uncorrelated with all other features should be considered irrelevant because outliers often correspond to violation of the model of normal data dependencies. Uncorrelated features cannot be used to model data dependencies. Therefore, if one uses a regression model to predict one of the features from the other features, and the *average* squared error is too large, then such a feature should be pruned. All features are standardized to unit variance and the root-mean squared error  $RMSE_k$  of predicting the  $k$ th feature from other features is computed. Note that if  $RMSE_k$  is larger than 1, then the error of prediction is greater than the feature variance and therefore the  $k$ th feature should be pruned. One can also use this approach to weight the features. Specifically, the weight of the  $k$ th feature is given by  $\max\{0, 1 - RMSE_k\}$ . Details of this model are discussed in section 7.7 of Chapter 7.

### 1.3.2 Extreme-Value Analysis

The most basic form of outlier detection is extreme-value analysis of 1-dimensional data. These are very specific types of outliers in which it is assumed that the values that are either too large or too small are outliers. Such special kinds of outliers are also important in many application-specific scenarios.

The key is to determine the *statistical tails of the underlying distribution*. As illustrated earlier in Figure 1.3, the nature of the tails may vary considerably depending upon the underlying data distribution. The normal distribution is the easiest to analyze, because most statistical tests (such as the  $Z$ -value test) can be interpreted directly in terms of probabilities of significance. Nevertheless, even for arbitrary distributions, such tests provide a good heuristic idea of the outlier scores of data points, even when they cannot be interpreted statistically. The problem of determining the tails of distributions has been widely studied in the statistics literature. Details of such methods will be discussed in Chapter 2.

Extreme-value statistics [437] is distinct from the traditional definition of outliers. The traditional definition of outliers, as provided by Hawkins, defines such objects by their *generative probabilities* rather than the extremity in their values. For example, in the data set  $\{1, 2, 2, 50, 98, 98, 99\}$  of 1-dimensional values, the values 1 and 99 could, very mildly, be considered extreme values. On the other hand, the value 50 is the average of the data set, and is most definitely not an extreme value. However, the value 50 is isolated from most of the other data values, which are grouped into small ranges such as  $\{1, 2, 2\}$  and  $\{98, 98, 99\}$ . Therefore, most probabilistic and density-based models would classify the value 50 as the strongest outlier in the data, and this result would also be consistent with Hawkins's gener-

ative definition of outliers. Confusions between extreme-value analysis and outlier analysis are common, especially in the context of multivariate data. This is quite often the case, since many extreme-value models also use probabilistic models in order to quantify the probability that a data point is an extreme value.

Although extreme-value analysis is naturally designed for univariate (one-dimensional) data, it is also possible to generalize it to multivariate data, by determining the points at the multidimensional *outskirts* of the data. It is important to understand that such outlier detection methods are tailored to determining *specific kinds of* outliers even in the multivariate case. For example, the point ‘A’ in both Figures 1.1(a) and (b) will not be deemed an extreme value by such methods, since it does not lie on the outer boundary of the data, even though it is quite clearly an outlier in Figure 1.1(a). On the other hand, the point ‘B’ in Figure 1.1(b) can be considered an extreme value, because it lies on the outskirts of the multidimensional data set.

Extreme-value modeling plays an important role in most outlier detection algorithms as a final step. *This is because most outlier modeling algorithms quantify the deviations of the data points from the normal patterns in the form of a numerical score.* Extreme-value analysis is usually required as a final step on these modeled deviations, since they are now represented as univariate values in which extreme values correspond to outliers. In many multi-criteria outlier detection algorithms, a vector of outlier scores may be obtained (such as extreme values of temperature and pressure in a meteorological application). In such cases, multivariate extreme-value methods can help in *unifying* these outlier scores into a single value, and also generate a binary label output. Therefore, even though the original data may not be in a form where extreme-value analysis is directly helpful, it remains an integral part of the outlier detection process. Furthermore, many real-world applications track statistical aggregates, in which extreme-value analysis provides useful insights about outliers.

Extreme-value analysis can also be extended to multivariate data with the use of distance- or depth-based methods [295, 343, 468]. However, these methods are applicable only to certain types of specialized scenarios in which outliers are known to be present at the boundaries of the data. Many forms of post-processing on multi-criteria outlier scores may use such methods. On the other hand, such methods are not very useful for *generic* outlier analysis, because of their inability to discover outliers in the sparse *interior* regions of a data set.

### 1.3.3 Probabilistic and Statistical Models

In probabilistic and statistical models, the data is modeled in the form of a closed-form probability distribution, and the parameters of this model are learned. Thus, the key assumption here is about the specific choice of the data distribution with which the modeling is performed. For example, a Gaussian mixture model assumes that the data is the output of a generative process in which each point belongs to one of  $k$  Gaussian clusters. The parameters of these Gaussian distributions are learned with the use of an *expectation-maximization (EM)* algorithm on the observed data so that the probability (or *likelihood*) of the process generating the data is as large as possible. A key output of this method is the membership probability of the data points to the different clusters, as well as the density-based fit to the modeled distribution. This provides a natural way to model the outliers, because data points that have very low fit values may be considered outliers. In practice, the logarithms of these fit values are used as the outlier scores because of the better propensity of the outliers to appear as extreme values with the use of log-fits. As discussed earlier, an extreme-value

test may be applied to these fit values to identify the outliers.

A major advantage of probabilistic models is that they can be easily applied to virtually any data type (or mixed data type), as long as an appropriate generative model is available for each mixture component. For example, if the data is categorical, then a discrete Bernoulli distribution may be used to model each component of the mixture. For a mixture of different types of attributes, a product of the attribute-specific generative components may be used. Since such models work with probabilities, the issues of data normalization are already accounted for by the generative assumptions. Thus, probabilistic models provide a generic EM-based framework, which is relatively easy to apply to any specific data type. This is not necessarily the case with many other models.

A drawback of probabilistic models is that they try to fit the data to a particular kind of distribution, which may sometimes not be appropriate. Furthermore, as the number of model parameters increases, over-fitting becomes more common. In such cases, the outliers may fit the underlying model of normal data. Many parametric models are also harder to interpret in terms of intensional knowledge, especially when the parameters of the model cannot be intuitively presented to an analyst in terms of underlying attributes. This can defeat one of the important purposes of anomaly detection, which is to provide diagnostic understanding of the abnormal data generative process. A detailed discussion of probabilistic methods, including the EM algorithm, is provided in Chapter 2.

### 1.3.4 Linear Models

These methods model the data along lower-dimensional subspaces with the use of linear correlations [467]. For example, in the case of Figure 1.4, the data is aligned along a 1-dimensional line in a 2-dimensional space. The optimal line that passes through these points is determined with the use of regression analysis. Typically, a least-squares fit is used to determine the optimal lower-dimensional hyperplane. The distances of the data points from this hyperplane are used to quantify the outlier scores because they quantify the deviations from the model of normal data. Extreme-value analysis can be applied on these scores in order to determine the outliers. For example, in the 2-dimensional example of Figure 1.4, a linear model of the data points  $\{(x_i, y_i), i \in \{1 \dots N\}\}$  in terms of two coefficients  $a$  and  $b$  may be created as follows:

$$y_i = a \cdot x_i + b + \epsilon_i \quad \forall i \in \{1 \dots N\} \quad (1.4)$$

Here,  $\epsilon_i$  represents the *residual*, which is the modeling error. The coefficients  $a$  and  $b$  need to be *learned from the data* to minimize the least-squares error, which is denoted by  $\sum_{i=1}^N \epsilon_i^2$ . This is a convex non-linear programming problem whose solution can be obtained in closed form. The squared residuals provide the outlier scores. One can use extreme-value analysis to identify the unusually large deviations, which should be considered outliers.

The concept of dimensionality reduction and principal component analysis (PCA) is quite similar [296], except that it uses a non-parametric approach in order to model the data correlations. PCA can be derived through multivariate regression analysis by determining the hyperplane that minimizes the least-squares error (i.e., distance) to the hyperplane. In other words, it provides a subspace of lower dimensionality with the least *reconstruction error* after projection. Outliers have large reconstruction errors because they do not conform to the aggregate subspace patterns in the data. Therefore, the reconstruction errors may be used as outlier scores. In addition, principal component analysis can be used for noise *correction* [21], in which the attributes of data points are modified to reduce noise. Outlier

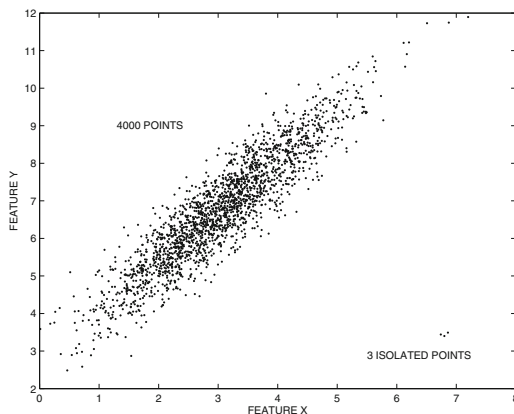


Figure 1.5: Small groups of anomalies can be a challenge to density-based methods

points are likely to be corrected more significantly than normal points. Methods like dimensionality reduction are special cases of the generic methodology of *matrix factorization*; the main advantage of the generic methodology is that it can even be used for incomplete data sets.

Dimensionality reduction and regression modeling are particularly hard to interpret in terms of the original attributes. This is because the subspace embedding is defined as a linear combination of attributes with positive or negative coefficients. This cannot easily be intuitively interpreted in terms specific properties of the data attributes. Nevertheless, some forms of dimensionality reduction, such as nonnegative matrix factorization, are highly interpretable. Dimensionality reduction, regression analysis, and matrix factorization methods for outlier detection are discussed in Chapter 3. Their natural nonlinear extensions such as kernel PCA, kernel SVMs, and neural networks, are also discussed in the same chapter. Furthermore, various forms of nonnegative matrix factorization are discussed in Chapters 8 and 12.

#### 1.3.4.1 Spectral Models

Many of the matrix decomposition methods such as PCA are also used in the context of graphs and networks. The main difference is in how the matrix is created for decomposition. Some variations of these methods, which are used in certain types of data such as graphs and networks, are also referred to as spectral models. Spectral methods are used commonly for clustering graph data sets, and are often used in order to identify anomalous changes in temporal sequences of graphs [280]. Spectral methods are closely related to matrix factorization, which can also be used in such settings [551]. Such models will be discussed in Chapters 3, 4, 5, and 12.

### 1.3.5 Proximity-Based Models

The idea in proximity-based methods is to model outliers as points that are isolated from the remaining data on the basis of similarity or distance functions. Proximity-based methods are among the most popular class of methods used in outlier analysis. Proximity-based methods may be applied in one of three ways, which are clustering methods, density-based methods



and nearest-neighbor methods. In clustering and other density-based methods, the dense regions in the data are found directly, and outliers are defined as those points that do not lie in these dense regions. Alternatively, one might define outliers as points that are located far away from the dense regions. The main difference between clustering and density-based methods is that clustering methods segment the data *points*, whereas the density-based methods such as histograms segment the data *space*. This is because the goal in the latter case is to estimate the density of test points in the data space, which is best achieved by space segmentation.

In nearest-neighbor methods [317, 456], the distance of each data point to its  $k$ th nearest neighbor is reported as its outlier score. By selecting a value of  $k > 1$ , small groups of tightly-knit points that are far away from the remaining data set can be identified and scored as outliers. It is reasonable to treat such sets of data points as outliers, because small related sets of points can often be generated by an anomalous process. For example, consider the case illustrated in Figure 1.5, which contains a large cluster containing 4000 data points, and a small set of isolated but three closely spaced and related anomalies. Such situations are quite common, because anomalies caused by the same (rare) process may result in small sets of data points that are almost identical. In this case, the points within an anomaly set are close to one another, and cannot be distinguished on the basis of the 1-nearest neighbor distance. Such anomalies are often hard to distinguish from noise by using certain types of density-based algorithms that are not sensitive to the global behavior of the data. On the other hand, the  $k$ -nearest neighbor approach can sometimes be effective. In the case of Figure 1.5, such sets of related anomalies may be identified by using  $k \geq 3$ . The  $k$ th nearest neighbor score provides an outlier score of the data set. This method can typically be computationally expensive, because it is required to determine the  $k$ th nearest neighbor of every point in the data set, which requires  $O(N^2)$  operations for a data set containing  $N$  points. However, in cases in which it is acceptable to report binary labels instead of scores, many of these distance computations can be pruned because some points can be shown to be non-outliers after a small number of distance computations. For example, if after computing the distances of a small fraction of the points to a particular point ‘A,’ the  $k$ -nearest neighbor distance of ‘A’ is lower than that of all the top- $r$  outliers found so far, then point ‘A’ is guaranteed to be a non-outlier. Therefore, no further distance computations to point ‘A’ need to be performed. As a result, the binary version of outlier detection often allows faster algorithms than the score-wise version of the problem. The latter is always quadratically related to the number of points in computational complexity. Quadratic computational complexity turns out to be surprisingly slow in real settings; it is often difficult to use these methods even for data sets containing a few hundred-thousand points, without leveraging some form of sampling.

In the case of clustering methods, the first step is to use a clustering algorithm in order to determine the dense regions of the data set. In the second step, some measure of the fit of the data points to the different clusters is used in order to compute an outlier score for the data point. For example, in the case of a  $k$ -means clustering algorithm, the distance of a data point to its nearest centroid may be used to measure its propensity to be an outlier. One needs to be careful with the use of clustering methods, because the specific data partitioning (and corresponding outlier scores) may vary significantly with the choice of clustering methodology. Therefore, it is usually advisable to cluster the data multiple times and average the scores obtained from the different runs [184, 406]. The results of such an approach are often surprisingly robust.

Density-based methods like histograms divide the data *space* into small regions, and the number of points in these regions are used to compute the outlier scores. Density-based

methods provide a high level of interpretability, when the sparse regions in the data can be presented in terms of combinations of the original attributes. For example, consider a sparse region constructed on the following subsets of attributes:

$$\text{Age} \leq 20, \text{Salary} \geq 100,000$$

Clearly, these constraints define a segment of the data space that is highly interpretable from a semantic point of view. It presents a clear description of *why* a data point should be considered an outlier. Some other methods such as kernel density estimation do not partition the data space, but are nevertheless focused on estimating the density of regions in the data space by replacing space segmentation with smoother kernel functions. Proximity-based methods for outlier detection are discussed in Chapter 4.

### 1.3.6 Information-Theoretic Models

Many of the aforementioned models for outlier analysis use various forms of data summarization such as generative probabilistic model parameters, clusters, or lower-dimensional hyperplanes of representation. These models implicitly generate a small *summary* of the data, and the *deviations* from this summary are flagged as outliers. Information-theoretic measures are also based on the same principle but in an indirect way. The idea is that outliers increase the minimum code length (i.e., minimum length of the *summary*) required to describe a data set because they represent *deviations* from natural attempts to summarize the data. For example, consider the following two strings:

```
ABABABABABABABABABABABABABABABAB
ABABACABABABABABABABABABABABABAB
```

The second string is of the same length as the first, and is different only at a single position containing the unique symbol ‘C.’ The first string can be described concisely as “AB 17 times.” However, the second string has a single position corresponding to the symbol ‘C.’ Therefore, the second string can no longer be described as concisely. In other words, the presence of the symbol ‘C’ somewhere in the string increases its minimum description length. It is also easy to see that this symbol corresponds to an outlier. Information-theoretic models are closely related to conventional models, because both use a concise representation of the data set as a baseline for comparison. For example, in the case of multidimensional data sets, both types of models use the following concise descriptions:

- A probabilistic model describes a data set in terms of generative model parameters, such as a mixture of Gaussian distributions or a mixture of exponential power distributions [92].
- A clustering or density-based summarization model describes a data set in terms of cluster descriptions, histograms, or other summarized representations, along with maximum error tolerances [284].
- A PCA model or spectral model describes the data in terms of lower dimensional subspaces of projection of multi-dimensional data or a condensed representation of a network [519], which is also referred to as its *latent representation*.
- A frequent pattern mining method describes the data in terms of an underlying code book of frequent patterns. These are among the most common methods used for information-theoretic anomaly detection [42, 151, 497].

All these models represent the data approximately in terms of individual condensed components representing aggregate trends. In general, outliers increase the length of the description in terms of these condensed components to achieve the same level of approximation. For example, a data set with outliers will require a larger number of mixture parameters, clusters, PCA-based subspace dimensionality, or frequent patterns in order to achieve *the same level of approximation*. Correspondingly, in information-theoretic methods, the key idea is to construct a *code book* in which to represent the data, and outliers are defined as points whose removal results in the *largest decrease* in description length [151], or the most accurate summary representation in the same description length after removal [284]. The term “code book” is rather loosely defined in outlier analysis and refers to the condensed aggregate components of the data in terms of which the data is described. The actual construction of the coding is often heuristic, and an effective choice is key to the success of the approach. In general, the determination of the minimum-length coding is a computationally intractable problem for a given data set, and therefore a variety of heuristic models (or code books) may be used for representation purposes [42, 151, 284, 497]. In many cases, these techniques can be related to conventional data summarization models for outlier analysis. In some cases, the coding is not explicitly constructed, but measures such as the entropy or Kolmogorov complexity are used as a surrogate in order to estimate the level of unevenness of a specific segment of the data [352, 312]. Segments with greater unevenness may be selectively explored to identify outliers. This represents a good use-case for information theoretic models because it is algorithmically simpler to quantify coding complexity than actually constructing the coding.

Conventional models look at this problem in a complementary way by *directly* defining outliers as points that are expressed in the *least precise way* by (or deviations from) from a *fixed* compression (e.g., clustering or factorization). On the other hand, information-theoretic models quantify the *differential impact on compression size* of removing an outlier point on a compression of fixed error (i.e., *aggregate* deviation). The two are clearly closely related, although the former is a more direct way of scoring points than the latter. Since information-theoretic methods largely differ from conventional models in terms of how the measure is defined, they often use similar methods as conventional techniques (e.g., probabilistic models [92], frequent pattern mining [42, 497], histograms [284], or PCA [519]) to create the coding representation. Therefore, most information-theoretic models cannot be considered a separate family from conventional models, and they will be discussed at various places in this book along with their conventional counterparts. It is noteworthy that the indirect approach used by information-theoretic models to score points can sometimes blunt the scores because of the impact of other points on the *aggregate* error. As a result, information-theoretic models often do not outperform their conventional counterparts. The best use-cases, therefore, arise in settings where quantifying the coding cost is algorithmically more convenient than directly measuring the deviations.

### 1.3.7 High-Dimensional Outlier Detection

The high-dimensional case is particularly challenging for outlier detection. The reason for this behavior is that many dimensions may be noisy and irrelevant for anomaly detection, which might also increase the propensity for pairwise distances to become more similar. The key point here is that *irrelevant* attributes have a dilution effect on the accuracy of distance computations and therefore the resulting outlier scores might also be inaccurate. When using distance-based algorithms to score outliers, one often observes the effect of weakly correlated and irrelevant attributes in the concentration of distances. In high-dimensional space, the

data becomes increasingly sparse, and all pairs of data points become almost equidistant from one another [25, 263]. As a result, the outlier scores become less distinguishable from one another.

In such cases, outliers are best emphasized in a lower-dimensional *local* subspace of relevant attributes. This approach is referred to as *subspace outlier detection* [4], which is an important class of algorithms in the field of outlier analysis. The assumption in subspace outlier detection is that *outliers are often hidden in the unusual local behavior of low-dimensional subspaces, and this deviant behavior is masked by full-dimensional analysis*. Therefore, it may often be fruitful to explicitly search for the subspaces in which the anomalous behavior of points is best emphasized. This approach is a generalization of both (full-dimensional) clustering and (full-data) regression analysis. It combines local data pattern analysis with subspace analysis in order to mine the significant outliers. This can be a huge challenge, because the simultaneous discovery of relevant data localities and subspaces in high dimensionality can be computationally very difficult. It is easy to make mistakes in selecting the “correct” subspace, and it has been suggested [31, 35] that such techniques can be meaningfully used *only* by identifying multiple *relevant* subspaces and combining the predictions from these different subspaces. This approach is closely related to the notion of *outlier ensembles* [31, 35], which will be discussed in the next section and also in Chapter 6.

Subspace methods are useful for interpreting outliers, especially when the subspaces are described in terms of the original attributes. In such cases, the outputs of the algorithms provide *specific combinations of attributes along with data locality* that are relevant to the anomalous characteristics. This type of interpretability is useful in cases where a small number of *explanatory* attributes need to be identified from a high-dimensional data set. Methods for high-dimensional outlier detection are discussed in Chapter 5.

## 1.4 Outlier Ensembles

---

In many data mining problems such as clustering and classification, a variety of meta-algorithms are used in order to improve the robustness of the underlying solutions. Such meta-algorithms combine the outputs of multiple algorithms and are referred to as *ensembles*. For example, common ensemble methods in classification include bagging, subsampling, boosting and stacking [11, 33, 176]. Similarly, ensemble methods are often used to improve the quality of the clustering [23]. Therefore, it is natural to ask whether such meta-algorithms also exist for outlier detection. The answer is in the affirmative, although the work on meta-algorithms for outlier detection is relatively recent as compared to other problems like classification and clustering in which it is well-established. A position paper that formalizes these issues may be found in [31] and a book on outlier ensembles may be found in [35]. In recent years, significant theoretical and algorithmic advancements have been made in the field of outlier ensembles [32]. This chapter will provide a broad overview of the field of outlier ensembles, and a more detailed discussion is provided in Chapter 6. There are two primary types of ensembles in outlier analysis:

- In *sequential ensembles*, a given algorithm or set of algorithms are applied sequentially, so that future applications of the algorithms are influenced by previous applications, in terms of either modifications of the base data for analysis or in terms of the specific choices of the algorithms. The final result is either a weighted combination of, or the final result of the last application of an outlier analysis algorithm. For example, in the context of the classification problem, boosting methods may be considered examples of sequential ensembles.

```

Algorithm SequentialEnsemble(Data Set:  $\mathcal{D}$ 
    Base Algorithms:  $\mathcal{A}_1 \dots \mathcal{A}_r$ )
begin
   $j = 1$ ;
  repeat
    Pick an algorithm  $\mathcal{A}_j$  based on results from
      past executions;
    Create a new data set  $f_j(\mathcal{D})$  from  $\mathcal{D}$  based
      on results from past executions;
    Apply  $\mathcal{A}_j$  to  $f_j(\mathcal{D})$ ;
     $j = j + 1$ ;
  until(termination);
  report outliers based on combinations of results
    from previous executions;
end

```

Figure 1.6: Sequential ensemble framework

- In *independent ensembles*, different algorithms, or different instantiations of the same algorithm are applied to either the complete data or portions of the data. The choices made about the data and algorithms applied are independent of the results obtained from these different algorithmic executions. The results from the different algorithm executions are combined together in order to obtain more robust outliers.

At a fundamental level, outlier ensembles are not very different from classification ensembles in terms of the underlying theoretical foundations [32]. Even though outlier detection is an unsupervised problem, the basic bias-variance theory from classification can be adapted to outlier detection by treating the underlying labels as unobserved [32]. As a result, many natural ensemble methods such as bagging and subsampling can be generalized easily to outlier detection with minor variations.

### 1.4.1 Sequential Ensembles

In sequential-ensembles, one or more outlier detection algorithms are applied sequentially to either all or portions of the data. The core principle of the approach is that each application of the algorithm enables a more refined execution with either a modified algorithm or data set. Thus, depending on the approach, either the data set or the algorithm may be changed in sequential executions. If desired, this approach can either be applied for a fixed number of times or executed to convergence. The broad framework of a sequential-ensemble algorithm is provided in Figure 1.6.

In each iteration, a successively refined algorithm is used on refined data based on results from previous executions. The function  $f_j(\cdot)$  is used to create a refinement of the data, which could correspond to data subset selection, attribute-subset selection, or generic data transformation methods. The description above is provided in a very general form, and many special cases can be instantiated from this framework. For example, in practice, only a single algorithm may be used on successive modifications of the data, as data is refined over time. The sequential ensemble may be applied for a fixed number of iterations

**Algorithm** IndependentEnsemble(Data Set:  $\mathcal{D}$   
 Base Algorithms:  $\mathcal{A}_1 \dots \mathcal{A}_r$ )  
**begin**  
 $j = 1$ ;  
**repeat**  
 Pick an algorithm  $\mathcal{A}_j$ ;  
 Create a new data set  $f_j(\mathcal{D})$  from  $\mathcal{D}$ ;  
 Apply  $\mathcal{A}_j$  to  $f_j(\mathcal{D})$ ;  
 $j = j + 1$ ;  
**until**(termination);  
**report** outliers based on combinations of results  
 from previous executions;  
**end**

Figure 1.7: Independent ensemble framework

or to convergence. The broad principle of sequential ensembles is that a greater knowledge of data with successive algorithmic execution helps focus on techniques and portions of the data that can provide fresh insights.

Sequential ensembles have not been sufficiently explored in the outlier analysis literature as general purpose meta-algorithms. However, many *specific* techniques in the outlier literature use methods that can be recognized as special cases of sequential ensembles. A classic example of this is the use of two-phase algorithms for building a model of the normal data. In the first-phase, an outlier detection algorithm is used in order to remove the obvious outliers. In the second phase, a *more robust* normal model is constructed after removing these obvious outliers. Thus, the outlier analysis in the second stage is more accurate because many of the training points that contaminate the model of normal data have been removed. Such approaches are commonly used for cluster-based outlier analysis (for constructing more robust clusters in later stages) [70], or for more robust histogram construction and density estimation (see Chapter 4).

## 1.4.2 Independent Ensembles

In independent ensembles, different instantiations of the algorithm or different portions of the data are used for outlier analysis. Alternatively, the same algorithm may be applied with a different initialization, parameter set, or random seed. The results from these different algorithm executions can be combined in order to obtain a more robust outlier score. Such algorithms comprise the vast majority of outlier ensemble methods in use today. A general-purpose description of independent-ensemble algorithms is provided in the pseudo-code description of Figure 1.7.

The broad principle of independent ensembles is that different algorithms might perform better on different parts of the data; therefore, a combination of the results from these algorithms might provide more robust results than any of the individual ensemble components. The resulting output is therefore no longer as dependent on the specific artifacts of a particular algorithm or data set. Independent ensembles are used frequently for high-dimensional outlier detection, because they enable the exploration of different subspaces of the data in which different types of deviants may be found. In fact, the area of subspace

outlier detection is deeply interconnected with outlier ensemble analysis (see Chapter 5).

There is a significant number of different ways in which different algorithms and training data sets may be leveraged for model combination. For example, the methods in [31, 32, 344, 367] sample subspaces from the underlying data in order to independently score outliers from each of these executions. Then, the scores from these different executions are unified into a single point-specific value. Similarly, methods such as bagging and subsampling, which combine the results from different training data sets in classification, have also been generalized to outlier detection [31, 32]. In some cases, randomized models are constructed by making randomized choices within an outlier scoring algorithm [368]. These methods will be discussed in Chapters 5 and 6.

## 1.5 The Basic Data Types for Analysis

---

Most of our aforementioned discussion is focused on multidimensional numerical data. Furthermore, it is assumed that the data records are independent of one another. However, in practice, the underlying data may be more complex both in attribute type and point-to-point dependence. Some examples of such real-world data types are discussed in this section.

### 1.5.1 Categorical, Text, and Mixed Attributes

Many data sets in real applications may contain categorical attributes that take on *discrete unordered* values. For example, demographic data might contain attributes such as race, gender, or ZIP code. Such attribute values are not ordered, and therefore require different analytical techniques. Mixed attribute data contain both numerical and categorical attributes. Most of the existing models can be extended to this case. In many cases, the major challenge is to construct a distance (or similarity) function that remains semantically meaningful for the case of discrete data.

Regression-based models can be used in a limited way over discrete attribute values, when the number of possible values of an attribute is not too large. The typical methodology is to convert the discrete data to binary data by creating one attribute for each categorical value. Regression models such as principal component analysis may then be applied to this binary data set. Such methods can be more easily extended to text, in which there is an inherent ordering among word frequencies. In such cases, the correlations among word occurrences can be used to create regression models. In fact, some of the most successful models for text de-noising are based on latent semantic analysis (LSA), which is a form of linear regression analysis [162]. Other common methods for text and categorical data include clustering [29], proximity-based methods [622], probabilistic models [578], and methods based on frequent pattern mining [42, 253, 497]. Methods for outlier detection in categorical, text, and mixed attribute data sets are discussed in Chapter 8.

### 1.5.2 When the Data Values have Dependencies

Most of the aforementioned discussion in this chapter is about the common multidimensional scenario, where it is assumed that the data records can be treated independently of one another. In practice, the different data values may be related to each other temporally, spatially, or through explicit network relationship links between the data items. The presence of such dependencies greatly changes the anomaly detection process even at the definition

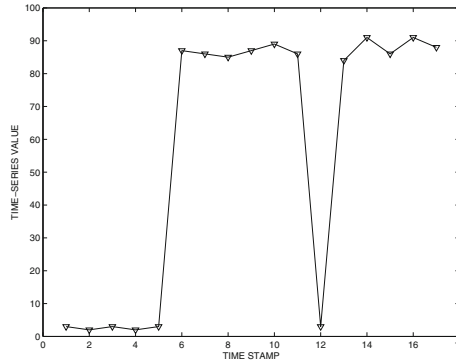


Figure 1.8: Example of Time Series

level. In such cases, the *expected values* of data items are influenced by their contextual dependencies, and therefore outliers are defined on the basis of such contextually modeled deviations. When a single data item (e.g., value from a time series) is declared as an anomaly because of its *relationship* to its related data items, it is referred to as a *contextual* outlier or anomaly. This is because such an outlier can only be understood in the *context* of its relationship with items in the temporal neighborhood. Such outliers are also sometimes referred to as *conditional anomalies* [503]. For example, a sudden spike in a time series is a contextual anomaly because it is very different from the values of its most recent time stamps; the key is to understand is that the most recent time stamps define the *expected* value of the series. Deviations from expected values represent outliers.

When a set of data items is declared anomalous *as a group of points*, it is referred to as a *collective* anomaly or outlier. For example, an unusual and rapid oscillation over time for a stock ticker value may be considered a collective anomaly, and it includes all the data items in the oscillation. Virtually, all anomalies in dependency-oriented data are *contextual* or *collective* anomalies, because they compute *expected* values based on relationships with adjacent data points in order to determine unexpected patterns. Furthermore, in such data sets, there are usually *multiple ways* to model anomalies, depending on what an analyst might be looking for. Some examples of such data domains are presented in this section.

### 1.5.2.1 Times-Series Data and Data Streams

Time series contains a set of values that are typically generated by continuous measurement over time. Therefore, the values in consecutive time stamps do not change very significantly, or change in a smooth way. In such cases, *sudden changes* in the underlying data records can be considered *anomalous events*. Therefore, the discovery of anomalous points in time series is closely related to the problem of anomalous *event detection*, and such events are often manifested as either *contextual or collective anomalies over related time stamps* [9, 19, 315]. The events are often created by sudden changes in the underlying system and may be of considerable interest to an analyst. For example, consider the following time series of values over consecutive time-stamps:

3, 2, 3, 2, 3, 87, 86, 85 87, 89, 86, 3, 84, 91, 86, 91, 88

The time series is illustrated in Figure 1.8. It is evident that there is a sudden change in the data value at time-stamp 6 from 3 to 87. This corresponds to an outlier. Subsequently,



the data stabilizes at this value, *and this becomes the new normal*. At time-stamp 12, the data value again dips to 3. *Even though this data value was encountered before*, it is still considered an outlier because of the sudden change in the consecutive data values. Thus, it is critical to understand that in this case, treating the data values as independent of one another is not helpful for anomaly detection, because the data values are highly influenced by the adjacent values of the data points. In other words, temporal *context* is important. Thus, the problem of outlier detection in time-series data is highly related to the problem of change detection because the normal models of data values are highly governed by adjacency in temporal ordering. When completely new data values are encountered, they are referred to as *novelties* [391, 392, 388], although outlier detection is relevant to any form of abrupt change, rather than only new data values.

It should be emphasized that change analysis and outlier detection (in temporal data) are closely related problems, but they are not necessarily identical. The change in a temporal data set could occur in one of two possible ways:

- The values and trends in the data stream change slowly over time, a phenomenon that is referred to as *concept drift* [390, 10]. In such cases, the concept drift can only be detected by careful analysis over a longer period of time, and is not immediately obvious in many circumstances.
- The values and trends in the data stream change *abruptly*, so as to *immediately arouse suspicion that the underlying data generation mechanism has somehow changed fundamentally*.

Of the two scenarios, only the second one can be used to identify outliers. It is also easy to see the parallels between the second scenario and Hawkins's definition of outliers [249], which was introduced at the very beginning of this chapter.

A common challenge in such scenarios is to perform the outlier detection *in real time*, as new data values arrive. Many scenarios of change analysis and anomaly detection in temporal data are too tightly integrated to be treated separately. In such settings, solutions for one can be used for the other and vice versa. On the other hand, the modeling formulations of anomaly detection in temporal data are very diverse, not all of which are directly related to change detection. Usually, online analysis is suited to change detection, whereas offline analysis may explore other unusual aspects of the data. Some examples are as follows:

- When the data is in the form of a time series (e.g., sensor data) large changes in *trends* may correspond to anomalies. These can be discovered as deviations from forecasted values using window-based analysis. In some cases, it may be desired to determine time-series subsequences of unusual shapes rather than change points in the data.
- For multidimensional data streams, changes in the aggregate distribution of the streaming data may correspond to unusual events. For example, network intrusion events may cause *aggregate* change points in a network stream. On the other hand, *individual* point novelties may or may not correspond to aggregate change points. The latter case is similar to multidimensional anomaly detection with an efficiency constraint for the streaming scenario.

Methods for anomaly detection in time-series data and multidimensional data streams are discussed in Chapter 9.

### 1.5.2.2 Discrete Sequences

Many discrete sequence-based applications such as intrusion-detection and fraud-detection are clearly temporal in nature. This scenario can be considered a categorical or discrete analog of time-series data in which individual positions contain categorical (symbolic) values. Discrete sequences may not necessarily be temporal in nature, but may be based on their relative *placement* with respect to one another. An example is the case of biological data in which the sequences are defined by their relative placement.

As in the case of autoregressive models of continuous data, it is possible to use (typically Markovian) *prediction-based* techniques to forecast the value of a single *position* in the sequence. Deviations from forecasted values are identified as *contextual* outliers. It is often desirable to perform the prediction in real time in these settings. In other cases, anomalous events can be identified only by variations from the normal *patterns* exhibited by the *subsequences* over multiple time stamps. This is analogous to the problem of unusual *shape detection* in time-series data, and it represents a set of *collective* outliers.

Therefore, discrete sequences are analogous to continuous sequences, except that the categorical values in the individual positions necessitate the use of different similarity functions, representation data structures, and predictive techniques. For example, discrete sequence forecasting requires (more complex) Markovian models as opposed to (simpler) autoregressive techniques. The problem formulations in the two cases are, however, similar at a conceptual level. The specific techniques used are different because numerical time-series values are *ordered* and comparable across a continuous spectrum, whereas discrete values are not. As a result of these differences, the case of discrete sequences has been addressed in a different chapter from time-series data.

Discrete data are common in many real applications. Most biological sequences are discrete, and therefore the value of each position is drawn from a set of categorical possibilities. Similarly, host-based intrusion applications typically lead to discrete data, because numerous diagnostic events are drawn from a discrete set of instances [126]. Methods for anomaly detection in discrete sequences are discussed in Chapter 10.

### 1.5.2.3 Spatial Data

In spatial data, many non-spatial attributes (e.g., temperature, pressure, image pixel color intensity) are measured at spatial locations. Unusual local changes in such values are reported as outliers. It should be pointed out that outlier detection in temporal data shares some resemblance to that in spatial data [523]. Both typically require the attribute of interest to exhibit a certain level of continuity. For example, consider the measurement of the temperature in which the measurement could be associated with a time-stamp and spatial coordinates. Just as it is expected that temperatures at consecutive time-stamps do not vary too much (temporal continuity), it is also expected that temperatures at spatially close locations do not vary too much (spatial continuity). In fact, such unusual spatial variations in sea-surface temperatures and pressures [523] are used in order to identify significant and anomalous spatiotemporal events in the underlying data (e.g., formation of cyclones). Spatiotemporal data is a generalization of both spatial and temporal data, and the methods used in either domain can often be generalized to such scenarios. Methods for finding outliers in spatial and spatiotemporal data are discussed in Chapter 11.

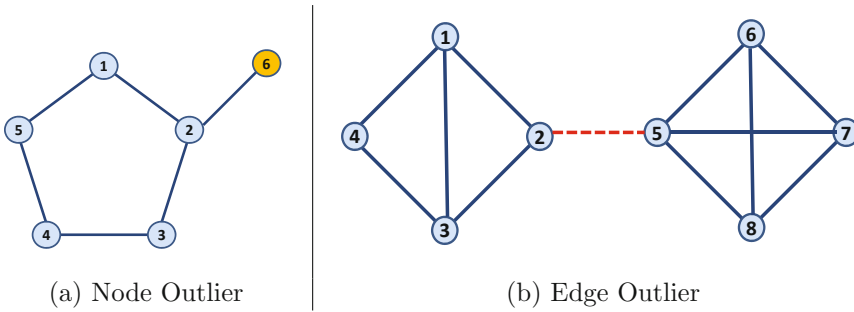


Figure 1.9: Examples of Node and Edge Outliers

#### 1.5.2.4 Network and Graph Data

In network or graph data, the data values may correspond to nodes in the network, and the relationships among the data values may correspond to the edges in the network. In such cases, outliers may be modeled in different ways depending on the irregularity of *either* the nodes in terms of their relationships to other nodes, or the edges themselves. For example, a node that shows irregularity in its structure within its locality may be considered an outlier [41]. Similarly, an edge that connects disparate communities of nodes may be considered a *relationship* or *community outlier* [17, 214]. In Figure 1.9, two examples of outliers in networks are illustrated. Figure 1.9(a) illustrates an example of a node outlier because the node 6 has an unusual locality structure that is significantly different from other nodes. On the other hand, the edge (2, 5) in Figure 1.9(b) may be considered a relationship outlier or community outlier, because it connects two distinct communities. Thus, there is greater complexity and flexibility in the definitions of outliers in complex data like graphs. There is also no unique way of defining the outliers and it is heavily dependent on the application domain at hand. In general, *the more complex the data is, the more the analyst has to make prior inferences of what is considered normal for modeling purposes.*

It is also possible to combine different types of dependencies for outlier modeling. For example, graphs may be temporal in nature. In such a case, the data may have both structural and temporal dependencies that change and influence each other over time [17]. Therefore, outliers may be defined in terms of significant changes in the underlying network community or distance structure. Such models combine network analysis and change detection to detect structural and temporal outliers. A detailed discussion of methods for temporal and non-temporal outlier detection in graphs is provided in Chapter 12. Relevant surveys are available in [14, 43, 457].

## 1.6 Supervised Outlier Detection

---

In many scenarios, previous examples of outliers may be available. A subset of the data may be labeled as anomalies, whereas the remaining data may be considered normal. In such cases, the anomaly identification process is referred to as *supervised outlier detection*, because labels are used in order to train a model that can determine *specific types* of anomalies. As a result, supervised models often provide *very different* results from the unsupervised case. For example, consider the following time-series:

3, 2, 3, 2, 3, 87, 2, 2, 3, 3, 3, 84, 91, 86, 91, 81

In this case, sudden changes in the data values (at 87 and 84) may be considered anomalies in the unsupervised scenario. However, in an application such as credit-card transaction levels, previous labeled examples of time-series may suggest that high consecutive values of the data should be considered anomalous. In such cases, the first occurrence of 87 should not be considered anomalous, whereas the occurrence of 84 along with its following values should be considered (collectively) anomalous.

As a general rule, one should always use supervision when labels are available because of its ability to discover application-specific anomalies of interest. Supervised outlier detection is a (difficult) special case of the classification problem. The main characteristic of this problem is that the labels are extremely unbalanced in terms of relative presence [132]. Since anomalies are far less common than normal points, it is possible for off-the-shelf classifiers to predict all test points as normal points and still achieve excellent accuracy. However, such results are not useful from a practical point of view. Therefore, the classifier is tuned, so that errors in classification of the anomalous class are penalized more heavily than the errors in classification of the majority class. The idea is that it is better to predict a negative class as an anomaly (false positive), rather than miss a true outlier (false negative). This leads to different trade-offs between false positives and false negatives than in other classification applications. These methods are referred to as *cost-sensitive learning*, because differential error costs are applied to different classes to regulate these trade-offs.

The supervised setting also supports several other variations of the classification problem that are quite challenging:

- A limited number of instances of the positive (outlier) class may be available, whereas the “normal” examples may contain an unknown proportion of outliers [183]. This is referred to as the Positive-Unlabeled Classification (PUC) problem in machine learning. This variation is still quite similar to the fully supervised rare-class scenario, except that the classification model needs to be more cognizant of the contaminants in the negative (unlabeled) class.
- Only instances of a subset of the normal and anomalous classes may be available, but some of the anomalous classes may be missing from the training data [388, 389, 538]. Such situations are quite common in scenarios such as intrusion detection in which some intrusions may be known, but other new types of intrusions are continually discovered over time. This is a *semi-supervised* setting for outlier detection. A combination of supervised and unsupervised methods may need to be used in such cases.
- In *active learning*, the problem of label acquisition is paired with the learning process [431]. The main assumption is that it is expensive to acquire examples of outliers, and therefore it is important to select the correct examples to label in order to perform accurate classification with the least number of labels.

Supervised methods for anomaly detection are discussed in Chapter 7.

## 1.7 Outlier Evaluation Techniques

---

A key question arises as to how the effectiveness of an outlier detection algorithm should be evaluated. Unfortunately, this is often a difficult task, because outliers, by definition, are rare. This means that the ground-truth labeling of data points as outliers or non-outliers is often not available. This is especially true for unsupervised algorithms, because if the

ground-truth were indeed available, it could have been used to create a more effective supervised algorithm. In the unsupervised scenario (without ground-truth), it is often difficult to judge the effectiveness of the underlying algorithms in a rigorous way. Therefore, much of the research literature uses case studies to provide an intuitive and qualitative evaluation of the underlying outliers in unsupervised scenarios.

In other unsupervised problems like data clustering, a common approach is to use *internal validity measures*, in which a model of “goodness” is used to measure the effectiveness of the algorithm. For example, a common measure of goodness in data clustering is the mean-squared radius of a cluster. The main problem with such measures is that they only provide an idea of how well the model of “goodness” *matches* the model of learning. After all, there is no way of knowing the “correct” model of goodness in unsupervised problems; the paradox is that if we knew this correct model then we should use it in the algorithm rather than for evaluation. In fact, it is relatively easy to game such internal validity models by choosing an algorithm that is related to the model of goodness; this problem is well-known in the clustering domain [33]. This is also referred to as the problem of *over-fitting* in internal evaluation. In outlier detection, this problem is far more severe because a small number of changes in the labels of the outliers can drastically affect the performance. For example, a distance-based internal measure would favor a distance-based algorithm over a linear (e.g., PCA-based) technique. Conversely, a linear model of internal validity would favor a PCA-based technique over a distance-based algorithm. Therefore, internal validity measures are rarely used for outlier detection, which seems to be a wiser approach than has been adopted by the data-clustering community.

In outlier detection, a more reasonable (albeit imperfect) approach is to use *external validity measures*. In some cases, the data sets may be adapted from imbalanced classification problems, and the rare labels may be used as surrogates for the ground-truth outliers. In such cases, a natural question arises as to how the ground-truth can be used to evaluate effectiveness. Most outlier-detection algorithms output an outlier score, and a threshold on this score is used to convert the scores into outlier labels. If the threshold is selected too restrictively to minimize the number of declared outliers, then the algorithm will miss true outlier points (false negatives). On the other hand, if the algorithm declares too many data points as outliers, then it will lead to too many false positives. This trade-off can be measured in terms of *precision* and *recall*, which are commonly used for measuring the effectiveness of set-based retrieval.

For any given threshold  $t$  on the outlier score, the declared outlier set is denoted by  $S(t)$ . As  $t$  changes, the size of  $S(t)$  changes as well.  $G$  represent the true set (ground-truth set) of outliers in the data set. Then, for any given threshold  $t$ , the *precision* is defined as the percentage of *reported* outliers that truly turn out to be outliers.

$$Precision(t) = 100 \cdot \frac{|S(t) \cap G|}{|S(t)|} \quad (1.5)$$

The value of  $Precision(t)$  is *not* necessarily monotonic in  $t$ , because both the numerator and denominator may change with  $t$  differently. The *recall* is correspondingly defined as the percentage of *ground-truth* outliers that have been reported as outliers at threshold  $t$ .

$$Recall(t) = 100 \cdot \frac{|S(t) \cap G|}{|G|} \quad (1.6)$$

By varying the parameter  $t$ , it is possible to plot a curve between the precision and the recall. This is referred to as the *Precision-Recall* curve. This curve is *not necessarily* monotonic.

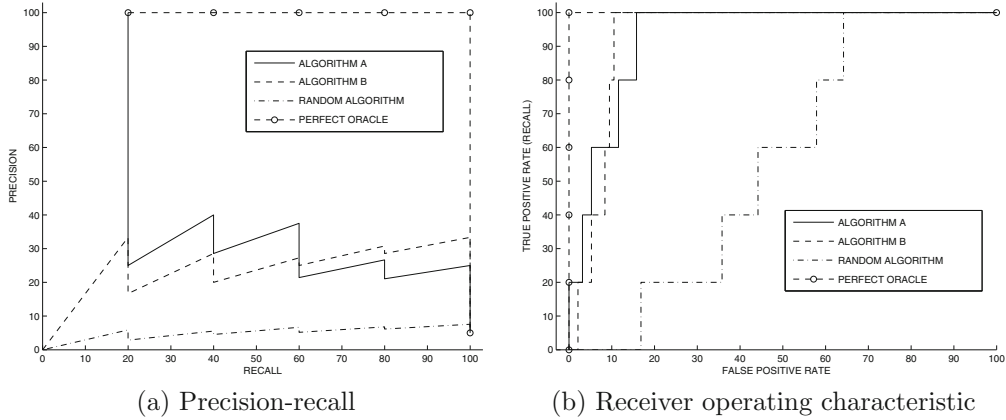


Figure 1.10: Precision-recall and receiver operating characteristic curves

Algorithm	Rank of Ground-truth Outliers
Algorithm A	1, 5, 8, 15, 20
Algorithm B	3, 7, 11, 13, 15
Random Algorithm	17, 36, 45, 59, 66
Perfect Oracle	1, 2, 3, 4, 5

Table 1.2: Rank of ground-truth outliers can be used to construct precision-recall curves

For more effective algorithms, high values of precision may often correspond to low values of recall and vice-versa. The precision-recall (PR) curve can also be generated by using thresholds on the *rank* of the data points, when sorted by outlier score. In the absence of ties in the outlier scores, a rank-based and score-based PR curve would be identical.

A *Receiver Operating Characteristic Curve (ROC)* is closely related to a Precision-Recall curve, but is sometimes visually more intuitive. In this case, the *True Positive Rate* is graphed against the *False Positive Rate*. The true positive rate  $TPR(t)$  is defined in the same way as the recall. The false positive rate  $FPR(t)$  is the percentage of the falsely reported positives out of the ground-truth negatives. In other words, the false-positive rate is a kind of “bad” recall, which reports the percentage of the negatives that are wrongly reported as outliers. Therefore, for a data set  $D$  with ground truth positives  $G$ , these definitions are as follows:

$$TPR(t) = Recall(t) = 100 \cdot \frac{|S(t) \cap G|}{|G|} \quad (1.7)$$

$$FPR(t) = BadRecall(t) = 100 \cdot \frac{|S(t) - G|}{|D - G|} \quad (1.8)$$

Therefore, the ROC curve plots the “bad” recall ( $FPR(t)$ ) on the X-axis, and the “good” recall ( $TPR(t)$ ) on the Y-axis. Note that both good and bad recall increase monotonically with the more relaxed values of the threshold  $t$  at which more outliers are reported. Therefore, the end points of the ROC curve are always at  $(0,0)$  and  $(100,100)$ , and a random method is expected to exhibit performance along the diagonal line connecting these points. The *lift* obtained above this diagonal line provides an idea of the additional accuracy of

the approach over a random method. The ROC curve is simply a different way to characterize the trade-offs than the precision-recall curve, although it has the advantage of being monotonic and more easily interpretable in terms of its lift characteristics.

In order to illustrate the insights gained from these different graphical representations, consider an example of a data set with 100 points, from which five points are outliers. Two algorithms  $A$  and  $B$  are applied to this data set, which rank all data points from 1 to 100, with a lower rank representing a greater propensity to be an outlier. Thus, the precision and recall values can be generated by determining the ranks of the 5 ground truth outlier points. In Table 1.2, some hypothetical ranks for the 5 ground truth outliers have been illustrated for the different algorithms. In addition, the ground truth ranks for a random algorithm have been indicated. This algorithm outputs a random outlier score for each point. Similarly, the ranks for a “perfect oracle” algorithm which ranks the correct top 5 points as outlier have also been illustrated in the table. The corresponding PR curve for this hypothetical output of outlier scores are illustrated in Figure 1.10(a). Other than the oracle algorithm, all the trade-off curves are non-monotonic. This is because the discovery of a new outlier at any particular relaxation in rank threshold results in a spike in the precision, which becomes less pronounced at higher values of the recall. The corresponding ROC curve is illustrated in Figure 1.10(b). Unlike the PR curve, this curve is clearly monotonic.

What do these curves really tell us? For cases in which one curve strictly dominates another the relative superiority between the two algorithms is unambiguous. For example, it is immediately evident that the oracle algorithm is superior to all algorithms, and the random algorithm is inferior to all the other algorithms. On the other hand, the algorithms  $A$  and  $B$  exhibit better performance at different parts of the ROC curve. In such cases, it is hard to say that one algorithm is strictly superior. From Table 1.2, it is clear that Algorithm  $A$  ranks three of the correct ground-truth outliers very highly, but the remaining two outliers are ranked poorly. In the case of Algorithm  $B$ , the highest ranked outliers are not as well ranked as the case of Algorithm  $A$ , although all five outliers are determined much earlier in terms of rank threshold. Correspondingly, Algorithm  $A$  dominates on the earlier part of the PR (or ROC) curve, whereas Algorithm  $B$  dominates on the later part. Some practitioners use the area under the ROC curve as a proxy for the overall effectiveness of the algorithm. A trapezoidal rule is used to compute the area, wherein the staircase-like ROC curve is replaced with a more convex approximation.

### 1.7.1 Interpreting the ROC AUC

The ROC AUC has the following simple probabilistic interpretation [246]:

**Theorem 1.7.1** *Given a ranking or scoring of a set of points in order of their propensity to be outliers (with higher ranks/scores indicating greater outlieriness), the ROC AUC is equal to the probability that a randomly selected outlier-inlier pair is ranked correctly (or scored in the correct order).*

In other words, one can also define the ROC AUC by computing the following mean over all outlier-inlier pairs in the data set:

$$\text{ROC AUC} = \text{MEAN}_{\overline{X}_i \in G, \overline{X}_j \in D-G} \begin{cases} 1 & \overline{X}_i \text{ ranked/scored higher than } \overline{X}_j \\ 0.5 & \overline{X}_i \text{ ranked/scored equal to } \overline{X}_j \\ 0 & \overline{X}_i \text{ ranked/scored lower than } \overline{X}_j \end{cases} \quad (1.9)$$

A nice characteristic of this definition is that it is easy to intuitively understand why a random algorithm would provide an AUC of around 0.5. This definition is also related to

the Kendall rank-correlation coefficient, in which a similar computation is performed over *all* pairs of objects rather than only outlier-inlier pairs, and the range<sup>2</sup> of the reward function is drawn from  $\{-1, 0, +1\}$  rather than  $\{0, 0.5, 1\}$ .

A measure, such as the AUC, should be used very carefully, because all parts of the ROC curve may not be equally important for different applications. This is because the initial part of the ROC curve is usually far more important. For example, for a data set containing 1000 points, it makes little difference whether an outlier data point is ranked at the 501<sup>st</sup> or 601<sup>st</sup> position. On the other hand, it makes a lot of difference whether an outlier data point is ranked at the 1<sup>st</sup> or 101<sup>st</sup> position. The AUC of the ROC makes little distinction between these two types of errors. In this sense, measures like precision can sometimes be more realistic in many settings. Other top-heavy measures such as the normalized discounted cumulative gain (NDCG) can also be adapted from the information retrieval and recommendation literature to outlier detection [34]. This measure computes the utility of a ranked list by given greater credit for outliers that are ranked at the top of the list.

## 1.7.2 Common Mistakes in Benchmarking

A common mistake made during the benchmarking of outlier detection algorithms occurs in cases where the algorithm is dependent on one or more user-defined parameters. For example, a  $k$ -nearest neighbor outlier detector scores a data point based on its  $k$ -nearest neighbor distance, where  $k$  is a user-defined parameter. It is common to run the algorithm repeatedly in order to select the best parameter at which the ROC AUC is optimized. Such an approach is not acceptable in outlier detection because one has effectively used knowledge of the outlier labels in selecting the parameters. In other words, the algorithm is no longer unsupervised. In problems like outlier detection, the only proper way to benchmark between a pair of algorithms is to run both over a “reasonable” range of parameters and compare the two algorithms using some central estimator of the performance over the resulting runs. For example, one can compare the median AUC performance or *box-plot performance*<sup>3</sup> of the two algorithms over the various parameter choices. Furthermore, different detectors might require the identification of a completely different range of parameters, which creates further challenges for comparison purposes. For example, a one-class support-vector machine might have a completely different choice of parameters than a nearest-neighbor detector, which further complicates a proper understanding of their relative performance. Unlike supervised settings, in which cross-validation is possible for selecting an approximately optimum parameter value for each classifier, the reasonable range for each outlier detector is often set on the basis of simple meta-characteristics of the data set such as its size and dimensionality. Clearly, these choices require some understanding and experience on the part of the analyst with various algorithms. There is only limited guidance available in the research literature on proper parameter choices for various algorithms.

A natural question arises as to whether one can use the *best* choice of parameters for *each* of the candidate algorithms to compare them. After all, such an approach does not *seem to* favor any particular algorithm over another since all algorithms are seeded with similar

---

<sup>2</sup>In the case of the Kendall rank-correlation coefficient, agreements are rewarded with +1, whereas disagreements are penalized with -1. Neutrals are credited values of 0. An outlier pair or an inlier pair would always belong to the neutral category. As a result, the absolute magnitude of the Kendall coefficient is more sensitive to the proportion of outliers in the data.

<sup>3</sup>See Figure 6.1 of Chapter 6 for an example. Box-plots are formally introduced in section 2.2.2.3 of Chapter 2.



knowledge. There are two problematic issues in doing so. First, the best choice of parameters cannot be known in unsupervised problems and therefore the results do not reflect the experience of an analyst in real-world settings. Second, such an evaluation will favor the more unstable algorithm, which is prone to overfitting into specific parameter choices. This will provide a skewed view of the algorithm to the analyst if the stable algorithm performs better than the unstable algorithm most of the time but the unstable algorithm performs extremely well at very specific parameter choices. After all, in an unsupervised setting, the likelihood of being correctly able to guess such parameter choices is similar to stumbling over a needle in a haystack. A second way of avoiding parametrization bias in comparing a pair of detectors [32] is to create an ensemble average of the execution of the algorithm over different settings, and compare the ensemble AUC of the different detectors. This truly provides an idea of the relative performance of the best possible avatars of various algorithms. At the end of the day, comparing two algorithms is somewhat of an art-form in the unsupervised setting, and one must rely at least a little on the experience and good judgement of the analyst in making proper experimental design choices.

## 1.8 Conclusions and Summary

---

The problem of outlier detection finds applications in numerous domains, where it is desirable to determine interesting and unusual events in the underlying generating process. The core of all outlier detection methods is the creation of a probabilistic, statistical or algorithmic model that characterizes the normal data. The deviations from this model are used to identify the outliers. A good domain-specific knowledge of the underlying data is often crucial designing simple and accurate models that do not overfit the underlying data. The problem of outlier detection becomes especially challenging, when significant relationships exist among the different data points. This is the case for time-series and network data in which the patterns in the relationships among the data points (whether temporal or structural) play the key role in defining the outliers. Outlier analysis has tremendous scope for further research, especially in the area of structural and temporal analysis.

## 1.9 Bibliographic Survey

---

A number of books and surveys have been written on the problem of outlier analysis. The classic books [74, 249, 467] in this area have mostly been written from the perspective of the statistics community. Most of these books were written before the wider adoption of database technology, and are therefore not written from a computational perspective. More recently, this problem has been studied quite extensively by the computer science community. These works consider practical aspects of outlier detection, corresponding to the cases where the data may be very large, or may have very high dimensionality. Numerous surveys have also been written that discuss the concept of outliers from different points of view, methodologies, or data types [38, 77, 125, 126, 313, 388, 389]. These surveys study outlier detection from different points of view such as the neural network setting [388, 389] or the one-class setting [313]. Among these, the survey by Chandola *et al.* [125] is the most recent and arguably the most comprehensive. This excellent review covers outlier detection quite broadly from the perspective of multiple communities. Detailed experimental comparisons of various outlier detection algorithms may be found in [35, 114, 184, 221, 419].

The basic models discussed in this chapter have also been researched extensively, and have been studied widely in the literature. Details of these methods (along with the corre-

sponding bibliographic notes) will be provided in later chapters. Here, only the most important works in each area are covered. The key statistical techniques on regression-based modeling are covered in [467]. The  $Z$ -value test discussed in section 1.2 is used commonly in the statistical literature, and many variants for limited sample sizes such as the Grubb's test [225] and  $t$ -value test are also available. The basic EM-algorithm for unsupervised modeling of data sets was first proposed in [164] and leveraged for outlier detection in [578]. The non-parametric technique of principal component analysis (PCA) discussed in section 1.2 is described well in [296]. The core technique of PCA was extended to text (with some minor variations) as Latent Semantic Indexing [162]. A variety of distance-based methods for outlier detection are proposed in [317, 456, 533], and density-based methods for outlier detection were proposed in [96]. Methods for interpreting distance-based outliers were first proposed in [318]. A variety of information-theoretic methods for outlier detection are discussed in [42, 57, 92, 122, 151, 256, 257, 352, 497].

The issues of poor behavior of high-dimensional applications (such as clustering and nearest-neighbor search) have been observed in several prior works in the literature [5, 7, 8, 25, 263]. The problem of high-dimensional outlier detection was first proposed in [4]. Subspace approaches for outlier detection were proposed in this paper, and a number of other recent methods have followed a similar line of work [308, 327, 402, 403, 404, 406, 604, 605, 606, 607, 619].

Outliers have been studied extensively in the context of different data domains. While numeric data is the most commonly studied case, numerous methods have also been proposed for categorical and mixed data [38, 578]. Methods for unsupervised outlier detection in text corpora are proposed in [240]. The problem of detecting outliers with dependencies has also been studied extensively in the literature. Methods for detecting outliers and changes in time series and streams were proposed in [9, 17, 19, 29, 310, 311, 312, 315]. Novelty detection [388] is an area that is closely related to outlier analysis, and it is often studied in the context of supervised models, where novel classes from a data stream are detected in real time [391, 392] with the use of learning methods. However, novelty detection is also studied often in the unsupervised scenario, particularly in the context of *first story detection* in topic detection and tracking in text streams [622]. Spatial outliers [2, 324, 376, 487, 488, 489, 490] are closely related to the problem of finding outliers in temporal data, since such data also show spatial continuity, just as temporal data show temporal continuity. Some forms of spatial data also have a temporal component to them, which requires the determination of spatiotemporal outliers [141, 142]. Outlier detection in discrete sequences is related to the problem of temporal outlier detection in continuous sequences. For discrete sequences, an excellent survey may be found in [126]. General surveys on anomaly detection in various types of temporal data may be found in [231, 232].

Methods for finding node outliers with unusual neighborhood behavior in graphs were proposed in [41], and techniques for finding relationship outliers, subgraph outliers and community outliers were proposed in [17, 214, 416, 452]. The primary ideas in all these methods is that outlier regions in a network are caused by unusual relationships in the form of edges, subgraphs, and communities. The problem of evolutionary network analysis in temporal networks is studied in [20, 233, 234, 519]. Surveys on anomaly detection in static and dynamic networks may be found in [14, 43, 457].

Recently, methods for *outlier ensembles* have been proposed. The work in [344] designs methods for using different subsets of features in outlier detection methods, and combining them in order to provide more effective results. The work in [402, 403, 404] shows how to combine the scores from different subspaces found by outlier detection algorithms in order to provide a unified and more robust result. The work in [367] proposes the notion of

*isolation forests* that are analogs of the successful notion of random forests in classification. Recently, the field has been formalized by positioning the existing (informal) work in the field of outlier ensembles, and also establishing theoretical foundations [31, 32, 35].

The supervised version of the outlier detection problem has been studied extensively in the form of *rare class detection*. For the supervised case, readers are referred to a general book on classification [176], since this problem is essentially a cost-sensitive variation [132, 182] on the standard classification problem, in which the class distributions are very imbalanced. In particular, the readers are referred to [132, 182] for a thorough discussion on the foundations of cost-sensitive learning from imbalanced data sets. A number of methods for classification from positive and unlabeled data are discussed in [183], and a good review of the previous work in this area may also be found from the references in this paper. The work in [431, 618, 619] first showed how human supervision could be used to significantly improve the effectiveness of outlier detection. Finally, the *semi-supervised* scenario of novelty detection has been discussed extensively in [388, 389, 538].

Evaluation methods in outlier analysis are identical to the techniques used in information retrieval, recommender systems, and (supervised) rare-class learning. In fact, most of the evaluation methods discussed in the Chapter 7 of a recent recommender systems book [34] can also be used for outlier analysis. A detailed discussion of ROC curves may be found in [192]. While the ROC and PR curves are the traditional methods for outlier evaluation, it has recently been noted [402] that these methods may not necessarily provide all the insights needed for different types of analysis. Therefore, the work in [402] has proposed a coefficient based on the Spearman correlation between the best possible ranking and the ranking determined by the algorithm.

## 1.10 Exercises

---

1. Which of the following points from each of the following sets of points below is an outlier? Why?
  - (1-dimensional) { 1, 3, 2, 1, 3, 2, 75, 1, 3, 2, 2, 1, 2, 3, 2, 1 }
  - (1-dimensional) { 1, 2, 3, 4, 2, 19, 9, 21, 20, 22 }
  - (2-dimensional) { (1, 9), (2, 9), (3, 9), (10, 10), (10, 3), (9, 1), (10, 2) }
2. Use MATLAB or any other mathematical software to create a histogram of the data distribution along each of the dimensions in the different cases of Exercise 1. Can you see the outliers visually? Which ones? In which case are the outliers not clear and why?
3. For the 2-dimensional case of Exercise 1, plot the data points on a 2-dimensional plane. Can you see the outliers visually? Which ones?
4. Apply the  $Z$ -value test to each of the cases in Exercise 1. For the 2-dimensional case, apply the  $Z$ -value test to the individual dimensions. Do you discover the correct outliers?
5. For the 2-dimensional case in Exercise 1, construct the function  $f(x_1, x_2) = |x_1 - x_2|$ . Apply the  $Z$ -value test to  $f(x_1, x_2)$  over each of the data points. Do you obtain the correct outliers, as suggested by your visual analysis in Exercise 3? Why?

6. Determine the nearest neighbor of each data point for the cases in Exercise 1. Which data points have the largest value of the nearest neighbor distance? Are they the correct outliers?
7. Apply a  $k$ -means clustering algorithm to each of the cases in Exercise 1, while setting  $k = 2$ . Which data points lie furthest from the two means thus found? Are these the correct outliers?
- 8 Consider the following time-series:
  - 1, 2, 3, 3, 2, 1, 73, 1, 2, 3, 5
  - 1, 2, 3, 4, 3, 2, 1, 3, 73, 72, 74, 73, 74, 1, 2, 3, 4, 2
  - 1, 2, 3, 5, 6, 19, 11, 15, 17, 2, 17, 19, 17, 18

Which data points would you consider outliers? How does the temporal component influence your choice of outliers? Now examine the points at which the time series changes significantly? How do these points relate to the outliers?

9. Consider the undirected network  $G = (N, A)$  of 8 nodes in  $N$  indexed from 1 through 8. Let the edge set  $A$  be  $\{ (1, 2), (1, 3), (1, 4), (1, 5), (1, 6), (1, 7), (1, 8) \}$ . Draw the network on paper to visualize it. Is there any node, which you would consider an outlier? Why?
  - Now delete the edge  $(1, 7)$ . Does this change the set of nodes you would consider outliers? Why?
10. Consider the undirected network  $G = (N, A)$  of 8 nodes in  $N$  indexed from 1 through 8. Let the edge set  $A$  be  $\{ (1, 2), (1, 3), (1, 4), (2, 3), (2, 4), (5, 7), (4, 7), (5, 6), (6, 8), (5, 8), (6, 7) \}$ . Draw the network on paper to visualize it. Is there any edge, which you would consider an outlier? Why?
11. Consider three algorithms  $A$ ,  $B$  and  $C$ , which are run on a data set with 100 points and 5 outliers. The rank of the outliers by score for the three algorithms are as follows:
  - $A$ : 1, 3, 5, 8, 11
  - $B$ : 2, 5, 6, 7, 9
  - $C$ : 2, 4, 6, 10, 13
 Draw the PR curves for each of the algorithms. Would you consider any of the algorithms strictly superior to any of the others? Why?

---

---

## Chapter 2

# Probabilistic and Statistical Models for Outlier Detection

---

---

“With four parameters, I can fit an elephant, and with five, I can make him wiggle his trunk.” – John von Neumann

## 2.1 Introduction

---

The earliest methods for outlier detection were rooted in probabilistic and statistical models and date back to the nineteenth century [180]. These methods were proposed well before the advent and popularization of computer technology and were therefore designed without much focus on practical issues such as data representation or computational efficiency. Nevertheless, the underlying mathematical models are extremely useful and have eventually been adapted to a variety of computational scenarios.

A popular form of statistical modeling in outlier analysis is that of detecting *extreme univariate values*. In such cases, it is desirable to determine data values at the tails of a univariate distribution along with a corresponding level of statistical significance. Although extreme univariate values belong to a very specific category of outliers, they have numerous applications. For example, virtually all outlier detection algorithms use numerical scores to measure the anomalousness of data points, and the final step in these algorithms is to determine the extreme values from these scores. The identification of statistically significant extreme values helps in the conversion of outlier scores into binary labels. Some examples of outlier scoring mechanisms, which are used by different classes of algorithms, are as follows:

- In probabilistic modeling, the likelihood fit of a data point to a generative model is the outlier score.
- In proximity-based modeling, the  $k$ -nearest neighbor distance, distance to closest cluster centroid, or local density value is the outlier score.
- In linear modeling, the residual distance of a data point to a lower-dimensional representation of the data is the outlier score.

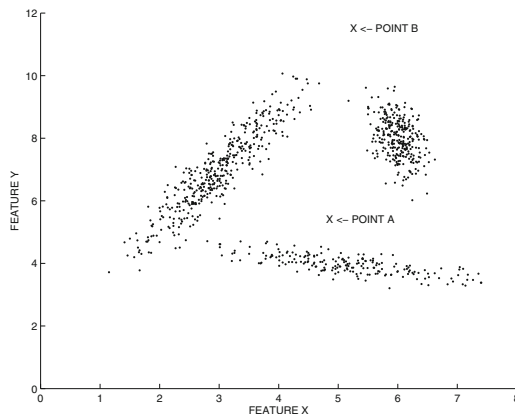


Figure 2.1: Example of the distinction between multivariate extreme values and outliers

- In temporal modeling, the deviation of a data point from its forecasted value is used to create the outlier score.

Thus, even when extreme-value modeling cannot be performed on the original data, the ability to determine the extreme values effectively from a set of outlier scores forms the cornerstone of all outlier detection algorithms as a final step. Therefore, the issue of extreme-value modeling will be studied extensively in this chapter.

Extreme-value modeling can also be easily extended to multivariate data. Data points that lie on the *pareto-extremes* of the data are referred to as multivariate extreme values. For example in Figure 2.1, data point ‘B’ is a multivariate extreme value. On the other hand, data point ‘A’ is an outlier but not a multivariate extreme value. Multivariate extreme-value analysis methods are sometimes also used for general outlier analysis. These techniques can *sometimes* perform surprisingly well in real-world outlier analysis applications, although they are not intended to be general outlier analysis methods. The reasons for this behavior are mostly rooted in the fact that real-world feature extraction methods sometimes create representations in which outliers are caused by extremes in values. For example, in a credit-card fraud detection application, it is common to extract features corresponding to the size and frequency of transactions. Unusually large or frequent transactions often correspond to outliers. Even if a subset of features is extracted in this way, it can greatly increase the effectiveness of multivariate extreme-value analysis methods for outlier detection. The drawback of using such methods in the general case is that data points like ‘A’ in Figure 2.1 are missed by such methods. Nevertheless, such methods should not be ignored in real applications in spite of this obvious drawback. In many cases, such techniques can be added as one or more components of an ensemble method (see Chapter 6) to enhance its accuracy.

It is also possible to use probabilistic modeling for finding general outliers beyond extreme values. For example, in Figure 2.1, one can model the data set as a *mixture* of three Gaussian components and therefore discover both outliers ‘A’ and ‘B.’ Mixture models can be considered probabilistic versions of clustering algorithms that discover the outliers as a side-product. A significant advantage of these methods is that they are easy to generalize to different data formats or even mixed attribute types, once a generative model for the data has been defined. Most probabilistic models assume a particular form to the underlying distribution for each mixture component (e.g., Gaussian) to model the normal patterns of data points. Subsequently, the parameters of this model are learned so that the observed

data has the maximum likelihood of being generated by the model [164]. This model is, therefore, a *generative* model for the data, and the probability of a particular data point being generated can be estimated from this model. Data points that have an unusually low probability of being generated by the model are identified as outliers. Mixture models are natural generalizations of multivariate extreme-value analysis; for example, if we modeled the mixture to contain a single Gaussian component, the approach specializes to one of the most well-known multivariate extreme-value analysis methods (see *Mahalanobis method* in section 2.3.4).

This chapter is organized as follows. The next section discusses statistical models for univariate extreme-value analysis. Methods for extreme-value analysis in multivariate data are discussed in section 2.3. Section 2.4 discusses methods for probabilistic modeling of outliers. Section 2.5 discusses the limitations of probabilistic models for outlier analysis. Section 2.6 presents the conclusions and summary.

## 2.2 Statistical Methods for Extreme-Value Analysis

---

In this section, we will present probabilistic and statistical methods for extreme-value analysis in univariate data distributions. The extreme values in a probability distribution are collectively referred to as the distribution *tail*. Statistical methods for extreme-value analysis quantify the probabilities in the tails of distributions. Clearly, a very low probability value of a tail indicates that a data value inside it should be considered anomalous. A number of tail inequalities bound these probabilities in cases where the actual distribution is not available.

### 2.2.1 Probabilistic Tail Inequalities

Tail inequalities can be used in order to bound the probability that a value in the tail of a probability distribution should be considered anomalous. The strength of a tail inequality depends on the number of assumptions made about the underlying random variable. Fewer assumptions lead to weaker inequalities but such inequalities apply to larger classes of random variables. For example, the *Markov* and *Chebyshev* inequalities are weak inequalities but they apply to very large classes of random variables. On the other hand, the Chernoff bound and Hoeffding inequality are both stronger inequalities but they apply to restricted classes of random variables.

The *Markov inequality* is one of the most fundamental tail inequalities, and it is defined for distributions that take on only non-negative values. Let  $X$  be a random variable, with probability distribution  $f_X(x)$ , a mean of  $E[X]$ , and a variance of  $Var[X]$ .

**Theorem 2.2.1 (Markov Inequality)** *Let  $X$  be a random variable that takes on only non-negative random values. Then, for any constant  $\alpha$  satisfying  $E[X] < \alpha$ , the following is true:*

$$P(X > \alpha) \leq E[X]/\alpha \tag{2.1}$$

**Proof:** Let  $f_X(x)$  represent the density function for the random variable  $X$ . Then, we have:

$$\begin{aligned} E[X] &= \int_x x \cdot f_X(x) \cdot dx = \int_{0 \leq x \leq \alpha} x \cdot f_X(x) \cdot dx + \int_{x > \alpha} x \cdot f_X(x) \cdot dx \\ &\geq \int_{x > \alpha} x \cdot f_X(x) \cdot dx \geq \int_{x > \alpha} \alpha \cdot f_X(x) \cdot dx \end{aligned}$$

The first inequality follows from the non-negativity of  $x$ , and the second follows from the fact that the integral is defined only over the cases in which  $x > \alpha$ . Furthermore, the term on the right-hand side of the last equation is exactly equal to  $\alpha \cdot P(X > \alpha)$ . Therefore, the following is true:

$$E[X] \geq \alpha \cdot P(X > \alpha) \quad (2.2)$$

The aforementioned inequality can be re-arranged in order to obtain the final result. ■

The Markov inequality is defined only for probability distributions of non-negative values and provides a bound only on the upper tail. In practice, it is often desired to bound both tails of arbitrary distributions. Consider the case where  $X$  is an arbitrary random variable, which is not necessarily non-negative. In such cases, the Markov inequality cannot be used directly. However, the (related) *Chebychev inequality* is very useful in such cases. The Chebychev inequality is a direct application of the Markov inequality to a non-negative derivative of random variable  $X$ :

**Theorem 2.2.2 (Chebychev Inequality)** *Let  $X$  be an arbitrary random variable. Then, for any constant  $\alpha$ , the following is true:*

$$P(|X - E[X]| > \alpha) \leq \text{Var}[X]/\alpha^2 \quad (2.3)$$

**Proof:** The inequality  $|X - E[X]| > \alpha$  is true if and only if  $(X - E[X])^2 > \alpha^2$ . By defining  $Y = (X - E[X])^2$  as a (non-negative) derivative random variable from  $X$ , it is easy to see that  $E[Y] = \text{Var}[X]$ . Then, the expression on the left hand side of the theorem statement is the same as determining the probability  $P(Y > \alpha^2)$ . By applying the Markov inequality to the random variable  $Y$ , one can obtain the desired result. ■

The main trick used in the aforementioned proof was to apply the Markov inequality to a non-negative function of the random variable. This technique can generally be very useful for proving other types of bounds, when the distribution of  $X$  has a specific form (such as the sum of Bernoulli random variables). In such cases, a parameterized function of the random variable can be used in order to obtain a parameterized bound. The underlying parameters can then be optimized for the tightest possible bound. Several well-known bounds such as the Chernoff bound and the Hoeffding inequality are derived with the use of this approach.

The Markov and Chebychev inequalities are relatively weak inequalities and often do not provide tight enough bounds to be useful in many practical scenarios. This is because these inequalities do not make any assumptions on the nature of the random variable  $X$ . Many practical scenarios can however be captured, when stronger assumptions are used on the random variable. In such cases, much tighter bounds on tail distributions are possible. A particular case is one in which a random variable  $X$  may be expressed as a sum of other independent bounded random variables.

### 2.2.1.1 Sum of Bounded Random Variables

Many practical observations, which are defined in the form of *aggregates*, can be expressed as sums of bounded random variables. Some examples of such scenarios are as follows:

**Example 2.2.1 (Sports Statistics)** *The National Basketball Association (NBA) draft teams have access to college basketball statistics for the different candidate players. For each player and each game, a set of quantitative values describe their various scoring statistics over different games. For example, these quantitative values could correspond to the number*



of dunks, assists, rebounds, and so on. For a particular statistic, the aggregate performance of any player can be expressed as the sum of their statistics over  $N$  different games:

$$X = \sum_{i=1}^N X_i$$

All values of  $X_i$  lie in the range  $[l, u]$ . The performances of a player over different games are assumed to be independent of one another. The long-term global mean of the statistic represented by  $X_i$  over all players is known to be  $\mu$ . The NBA draft teams would like to identify the anomalous players on the basis of each statistic.

In this example, the aggregate statistic is represented as a sum of bounded random variables. The corresponding tail bounds can be quantified with the use of the *Hoeffding inequality*.

In many cases, the individual random variable components in the aggregation are not only bounded, but also binary. Thus, the aggregate statistic can be expressed as a sum of Bernoulli random variables.

**Example 2.2.2 (Grocery Shopping)** *A grocery store keeps track of the number of customers (from its frequent purchaser program), who have frequented the store on a particular day. The long term probability of any customer  $i$  attending the store on a given day is known to be  $p_i$ . The behavior of the different customers is also known to be independent of one another. For a given day, evaluate the probability that the store receives more than  $\eta$  (frequent purchase program) customers.*

In the second example, the number of customers can be expressed as a sum of independent Bernoulli random variables. The corresponding tail distributions can be expressed in terms of the *Chernoff bound*. Finally, we provide a very common application of anomaly detection from aggregates, which is that of fault diagnosis in manufacturing.

**Example 2.2.3 (Manufacturing Quality Control)** *A company uses a manufacturing assembly line to produce a product, which may have faults in it with a pre-defined (low) probability  $p$ . The quality-control process samples  $N$  products from the assembly line, and examines them closely to count the number of products with defects. For a given count of faulty products, evaluate the probability that the assembly line is behaving anomalously.*

The sample size  $N$  is typically large, and, therefore, it is possible to use the *Central Limit Theorem* to assume that the samples are normally distributed. According to this theorem, the sum of a large number of independent and identical normal distributions converges to a normal distribution.

The different types of bounds and approximations will be formally introduced in this section. The Chernoff bounds and the Hoeffding inequality will be discussed first. Since the expressions for the lower tail and upper tails are slightly different, they will be addressed separately. The lower-tail Chernoff bound is introduced below.

**Theorem 2.2.3 (Lower-Tail Chernoff Bound)** *Let  $X$  be random variable that can be expressed as the sum of  $N$  independent binary (Bernoulli) random variables, each of which takes on the value of 1 with probability  $p_i$ .*

$$X = \sum_{i=1}^N X_i$$

Then, for any  $\delta \in (0, 1)$ , we can show the following:

$$P(X < (1 - \delta) \cdot E[X]) < e^{-E[X] \cdot \delta^2 / 2} \quad (2.4)$$

where  $e$  is the base of the natural logarithm.

**Proof:** The first step is to show the following inequality:

$$P(X < (1 - \delta) \cdot E[X]) < \left( \frac{e^{-\delta}}{(1 - \delta)^{(1 - \delta)}} \right)^{E[X]} \quad (2.5)$$

The *unknown* parameter  $t > 0$  is introduced in order to create a parameterized bound. The lower-tail inequality of  $X$  is converted into an upper-tail inequality on the exponentiated expression  $e^{-t \cdot X}$ . This random expression can be bounded by the Markov inequality, and it provides a bound as a function of  $t$ . This function of  $t$  can be optimized, in order to obtain the tightest possible bound. By using the Markov inequality on the exponentiated form, the following can be derived:

$$P(X < (1 - \delta) \cdot E[X]) \leq \frac{E[e^{-t \cdot X}]}{e^{-t \cdot (1 - \delta) \cdot E[X]}}$$

By expanding  $X = \sum_{i=1}^N X_i$  in the exponent, the following can be obtained:

$$P(X < (1 - \delta) \cdot E[X]) \leq \frac{\prod_i E[e^{-t \cdot X_i}]}{e^{-t \cdot (1 - \delta) \cdot E[X]}} \quad (2.6)$$

The aforementioned simplification uses the fact that the expectation of the product of independent variables is equal to the product of the expectations. Since each  $X_i$  is Bernoulli, the following can be shown:

$$E[e^{-t \cdot X_i}] = 1 + E[X_i] \cdot (e^{-t} - 1) < e^{E[X_i] \cdot (e^{-t} - 1)}$$

The second inequality follows from polynomial expansion of  $e^{E[X_i] \cdot (e^{-t} - 1)}$ . By substituting this inequality back into Equation 2.6, and using  $E[X] = \sum_i E[X_i]$ , the following may be obtained:

$$P(X < (1 - \delta) \cdot E[X]) \leq \frac{e^{E[X] \cdot (e^{-t} - 1)}}{e^{-t \cdot (1 - \delta) \cdot E[X]}}$$

The expression on the right is true for any value of  $t > 0$ . It is desired to determine the value of  $t$  that provides the *tightest possible* bound. Such a value of  $t$  may be obtained by computing the derivative of the expression with respect to  $t$  and setting it to 0. It can be shown that the resulting value of  $t = t^*$  from this optimization process is as follows:

$$t^* = \ln(1/(1 - \delta)) \quad (2.7)$$

By using this value of  $t^*$  in the aforementioned inequality, it can be shown to be equivalent to Equation 2.5. This completes the first part of the proof.

The first two terms of the Taylor expansion of the logarithmic term in  $(1 - \delta) \cdot \ln(1 - \delta)$  can be expanded to show that  $(1 - \delta)^{(1 - \delta)} > e^{-\delta + \delta^2 / 2}$ . By substituting this inequality in the denominator of Equation 2.5, the desired result is obtained. ■

A similar result for the upper-tail Chernoff bound may be obtained, albeit in a slightly different form.

**Theorem 2.2.4 (Upper-Tail Chernoff Bound)** *Let  $X$  be random variable, which is expressed as the sum of  $N$  independent binary (Bernoulli) random variables, each of which takes on the value of 1 with probability  $p_i$ .*

$$X = \sum_{i=1}^N X_i$$

Then, for any  $\delta \in (0, 2 \cdot e - 1)$ , the following is true:

$$P(X > (1 + \delta) \cdot E[X]) < e^{-E[X] \cdot \delta^2 / 4} \quad (2.8)$$

where  $e$  is the base of the natural logarithm.

**Proof:** The first step is to show the following inequality:

$$P(X > (1 + \delta) \cdot E[X]) < \left( \frac{e^\delta}{(1 + \delta)^{(1 + \delta)}} \right)^{E[X]} \quad (2.9)$$

As before, this can be done by introducing the unknown parameter  $t > 0$ , and converting the upper-tail inequality on  $X$  into that on  $e^{t \cdot X}$ . This can be bounded by the Markov Inequality as a function of  $t$ . This function of  $t$  can be optimized, in order to obtain the tightest possible bound.

It can be further shown by algebraic simplification that the inequality in Equation 2.9 provides the desired result for all values of  $\delta \in (0, 2 \cdot e - 1)$ . ■

Next, the Hoeffding inequality will be introduced. The Hoeffding inequality is a more general tail inequality than the Chernoff bound, because it does not require the underlying data values to be drawn from a Bernoulli distribution. In this case, the  $i$ th data value needs to be drawn from *the bounded interval*  $[l_i, u_i]$ . The corresponding probability bound is expressed in terms of the parameters  $l_i$  and  $u_i$ . Thus, the scenario for the Chernoff bound is a special case of that for the Hoeffding inequality. We state the Hoeffding inequality below, for which both the upper- and lower-tail inequalities are identical.

**Theorem 2.2.5 (Hoeffding Inequality)** *Let  $X$  be a random variable that can be expressed as the sum of  $N$  independent random variables, each of which is bounded in the range  $[l_i, u_i]$ .*

$$X = \sum_{i=1}^N X_i$$

Then, for any  $\theta > 0$ , the following can be shown:

$$P(X - E[X] > \theta) \leq e^{-\frac{2 \cdot \theta^2}{\sum_{i=1}^N (u_i - l_i)^2}} \quad (2.10)$$

$$P(E[X] - X > \theta) \leq e^{-\frac{2 \cdot \theta^2}{\sum_{i=1}^N (u_i - l_i)^2}} \quad (2.11)$$

**Proof:** The proof of the upper-tail portion will be briefly described here. The proof of the lower-tail inequality is identical. For any choice parameter  $t \geq 0$ , the following is true:

$$P(X - E[X] > \theta) = P(e^{t \cdot (X - E[X])} > e^{t \cdot \theta}) \quad (2.12)$$

The Markov inequality can be used to show that the right-hand probability is at most  $E[e^{(X - E[X])}] \cdot e^{-t \cdot \theta}$ . The expression within  $E[e^{(X - E[X])}]$  can be expanded in terms of the

Table 2.1: Comparison of different methods used to bound tail probabilities

Result	Scenario	Strength
Chebychev	Any random variable	Weak
Markov	Nonnegative random variable	Weak
Hoeffding	Sum of independent bounded random variables	Strong (Exponentially reduces with samples)
Chernoff	Sum of i.i.d. Bernoulli random variables	Strong (Exponentially reduces with samples)
CLT	Sum of <b>many</b> i.i.d. variables	Almost exact
Generalized CLT	Sum of <b>many</b> independent and bounded variables	Almost exact

individual components  $X_i$ . Since the expectation of the product is equal to the product of the expectations of independent random variables, the following can be shown:

$$P(X - E[X] > \theta) \leq e^{-t \cdot \theta} \cdot \prod_i E[e^{t \cdot (X_i - E[X_i])}] \quad (2.13)$$

The key is to show that the value of  $E[e^{t \cdot (X_i - E[X_i])}]$  is at most equal to  $e^{t^2 \cdot (u_i - l_i)^2 / 8}$ . This can be shown with the use of an argument that uses the convexity of the exponential function  $e^{t \cdot (X_i - E[X_i])}$  in conjunction with Taylor's theorem (see Exercise 12).

Therefore, the following is true:

$$P(X - E[X] > \theta) \leq e^{-t \cdot \theta} \cdot \prod_i e^{t^2 \cdot (u_i - l_i)^2 / 8} \quad (2.14)$$

This inequality holds for any positive value of  $t$ . Therefore, in order to find the tightest bound, the value of  $t$  that minimizes the right-hand side of the above equation needs to be determined. The optimal value of  $t = t^*$  can be shown to be the following:

$$t^* = \frac{4 \cdot \theta}{\sum_{i=1}^N (u_i - l_i)^2} \quad (2.15)$$

By substituting the value of  $t = t^*$ , the desired result may be obtained. The lower-tail bound may be derived by applying the aforementioned steps to  $P(E[X] - X > \theta)$  rather than  $P(X - E[X] > \theta)$ . ■

Thus, the different inequalities may apply to scenarios of different generality, and may also have different levels of strength. These different scenarios are presented in Table 2.1.

An interesting observation is that the Hoeffding tail bounds decay exponentially with  $\theta^2$ , which is exactly how the normal distribution behaves. This is not very surprising, because the sum of a large number of independent bounded random variables converges to the normal distribution according to the *Central Limit Theorem (CLT)*. Such a convergence is useful, because the bounds provided by an exact distribution (or a close approximation) are much tighter than any of the aforementioned tail inequalities.

**Theorem 2.2.6 (Central Limit Theorem)** *The sum of a large number  $N$  of independent and identically distributed random variables with mean  $\mu$  and standard deviation  $\sigma$  converges to a normal distribution with mean  $\mu \cdot N$  and standard deviation  $\sigma \cdot \sqrt{N}$ .*

A more generalized form of the CLT can also be applied to sums of independent variables (not necessarily identical), in which the variables are sufficiently bounded in terms of underlying moment measures. An example of such a generalization of the CLT is the *Lyapunov* CLT [88]. The basic idea is that the means and variances of a sum of a large number of independent (but not identically distributed) random variables can be approximated by the corresponding sums of the means and variances, respectively. Some weak assumptions on the underlying distributions are also imposed for the condition to hold. Refer to the bibliographic notes.

### 2.2.2 Statistical-Tail Confidence Tests

The normal distribution has numerous applications such as statistical-tail confidence testing. In statistical-tail confidence tests, the extreme values from a set of data values distributed according to a normal distribution are identified. The assumption of a normal distribution is rather ubiquitous in real domains. This is true not just for variables that are expressed as sums of random samples (as discussed in the previous section), but many variables that are generated by different random processes. The density function  $f_X(x)$  for the normal distribution with mean  $\mu$  and standard deviation  $\sigma$  is defined as follows:

$$f_X(x) = \frac{1}{\sigma \cdot \sqrt{2 \cdot \pi}} \cdot e^{-\frac{(x-\mu)^2}{2 \cdot \sigma^2}} \quad (2.16)$$

In some settings, it is appropriate to assume that the mean  $\mu$  and standard deviation  $\sigma$  of the modeling distribution are known. This is the case, when a very large number of samples of the data are available for accurate estimation of  $\mu$  and  $\sigma$ . In other cases,  $\mu$  and  $\sigma$  might be available from domain knowledge. Then, the  $Z$ -value  $z_i$  of an observed value  $x_i$  can be computed as follows:

$$z_i = (x_i - \mu) / \sigma \quad (2.17)$$

Since the normal distribution can be directly expressed as a function of the  $Z$ -value (and no other parameters), it follows that the tail probability of point  $x_i$  can also be expressed as a function of  $z_i$ . In fact, the  $Z$ -value corresponds to a scaled and translated normal random variable, which is also known as the *standard* normal distribution with mean 0 and variance 1. Therefore, the cumulative standard normal distribution can be used directly in order to determine the exact value of the tail probability at that value of  $z_i$ . From a practical perspective, since this distribution is not available in closed form, normal distribution tables are used in order to map the different values of  $z_i$  to probabilities. This provides a statistical level of significance, which can be interpreted directly as a probability of the data point being an outlier. The underlying assumption is that the data was generated by a normal distribution.

#### 2.2.2.1 $t$ -Value Test

The aforementioned discussion assumes that the mean and standard deviation of the modeling distribution can be estimated very accurately from a large number of samples. However, in practice, the available data sets might be small. For example, for a sample with 20 data points, it is much harder to model the mean and standard deviations accurately. How do we accurately perform statistical-significance tests in such cases?

The *Student's  $t$ -distribution* provides an effective way to model anomalies in such scenarios. This distribution is defined by a parameter known as the *number of degrees of freedom*  $\nu$ , which is closely defined by the available sample size. The  $t$ -distribution approximates the

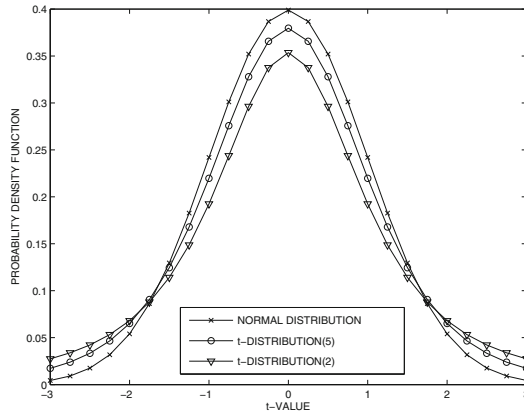


Figure 2.2: The  $t$ -distributions for different numbers of degrees of freedom (corresponding to different sample sizes)

normal distribution extremely well for larger degrees of freedom ( $> 1000$ ), and converges to the normal distribution in the limit where it goes to  $\infty$ . For fewer degrees of freedom (or sample size), the  $t$ -distribution has a similar bell-shaped curve as the normal distribution, except that it has heavier tails. This is quite intuitive, because the heavier tail accounts for the loss in statistical significance from the inability to accurately estimate the mean and standard deviation of the modeling (normal) distribution from fewer samples.

The  $t$ -distribution is expressed as a function of *several independent identically-distributed standard normal distributions*. It has a single parameter  $\nu$  that corresponds to the number of *degrees of freedom*. This regulates the *number* of such normal distributions, in terms of which it is expressed. The parameter  $\nu$  is set to  $N - 1$ , where  $N$  is the total number of available samples. Let  $U_0 \dots U_\nu$  be  $\nu + 1$ , independent and identically distributed normal distributions with zero mean and unit standard deviation. Such a normal distribution is also referred to as the *standard normal distribution*. Then, the  $t$ -distribution is defined as follows:

$$T(\nu) = \frac{U_0}{\sqrt{(\sum_{i=1}^{\nu} U_i^2)/\nu}} \quad (2.18)$$

The intuition for using the  $t$ -distribution is that the denominator explicitly models the randomness of estimating the standard deviation of the underlying normal distribution with the use of only a *small* number of independent samples. The term  $\sum_{i=1}^{\nu} U_i^2$  in the denominator is a  $\chi^2$  distribution with parameter  $\nu$ , and the entire (scaled) denominator converges to 1, when  $\nu \Rightarrow \infty$ . Therefore, in the limiting case, when a large number of samples are available, the randomness contributed by the denominator disappears, and the  $t$ -distribution converges to the normal distribution. For smaller values of  $\nu$  (or sample sizes), this distribution has a heavier tail. Examples of the  $t$ -distribution for different values of  $\nu$  are provided in Figure 2.2. It is evident that  $t$ -distributions with fewer degrees of freedom have heavier tails.

The process of extreme-value detection with a small number of samples  $x_1 \dots x_N$  proceeds as follows. First, the mean and standard deviation of the sample are estimated. This is then used to compute the  $t$ -value of each data point directly from the sample. The  $t$ -value is computed in an identical way as the  $Z$ -value. The tail probability of each data point is computed from the cumulative density function of the  $t$ -distribution with  $(N - 1)$ -degrees of

freedom. As in the case of the normal distribution, standardized tables are available for this purpose. From a practical perspective, if more than 1000 samples are available, then the  $t$ -distribution (with at least 1000 degrees of freedom) is so close to the normal distribution, that it is possible to use the normal distribution as a very good approximation.

### 2.2.2.2 Sum of Squares of Deviations

A common situation in outlier detection is the need to unify the deviations along independent criteria into a single outlier score. Each of these deviations is typically modeled as a  $Z$ -value from an independent and identically distributed standard normal distribution. The aggregate deviation measure is then computed as the sum of the squares of these values. For a  $d$ -dimensional data set, this is a  $\chi^2$ -distribution with  $d$  degrees of freedom. A  $\chi^2$ -distribution with  $d$  degrees of freedom is defined as the sum of the squares of  $d$  independent standard normal random variables. In other words, consider the variable  $V$ , which is expressed as the squared sum of independent and identically distributed standard normal random variables  $Z_i \sim N(0, 1)$ :

$$V = \sum_{i=1}^d Z_i^2$$

Then,  $V$  is a random variable drawn from a  $\chi^2$ -distribution with  $d$  degrees of freedom.

$$V \sim \chi^2(d)$$

Although a detailed discussion of the characteristics of the  $\chi^2$ -distribution is skipped here, its cumulative distribution is not available in closed form, but it needs to be computationally evaluated. From a practical standpoint, cumulative probability tables are typically available for modeling purposes. The cumulative probability tables of the  $\chi^2$ -distribution can then be used in order to determine the probabilistic level of significance for that aggregate deviation value. This approach is particularly useful when the deviations are modeled to be statistically independent of one another. As we will see in Chapter 3, such situations could arise in models such as principal component analysis, where the errors along the different components are often modeled as independent normal random variables.

### 2.2.2.3 Visualizing Extreme Values with Box Plots

An interesting approach to visualize univariate extreme values is the use of *box plots* or *box and whisker diagrams*. Such an approach is particularly useful in the context of visualizing outlier scores. In a box-plot, the statistics of a univariate distribution are summarized in terms of five quantities. These five quantities are the “minimum/maximum” (whiskers), the upper and lower quartiles (boxes), and the median (line in middle of box). We have enclosed quotations around two of these quantities because they are defined in a non-standard way. The distance between the upper and lower quartiles is referred to as the *inter-quartile range (IQR)*. The “minimum” and “maximum” are defined in a (non-standard) trimmed way in order to define the location of the whiskers. If there are no points more than 1.5 IQR above the top quartile value (upper end of the box), then the upper whisker is the true maximum. Otherwise, the upper whisker is set at 1.5 times the IQR from the upper end of the box. An exactly analogous rule holds true for the lower whisker, which is set at 1.5 IQR from the lower end of the box. In the special case of normally distributed data, a value of 1.5 IQR more than the top quartile corresponds to a distance of 2.7 times the standard deviation

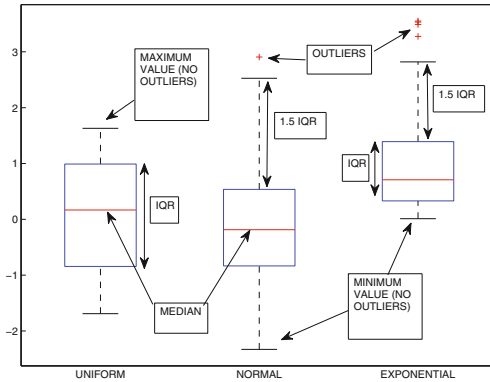


Figure 2.3: Visualizing univariate extreme values with box plots

(from the mean). Therefore, the whiskers are roughly placed at locations similar to the  $3 \cdot \sigma$  cut-off points in a normal distribution.

An example of a box plot is illustrated in Figure 2.3. In this case we have shown 100 data points corresponding to each of the (i) uniform distribution with zero mean and unit variance, (ii) standard normal distribution, and (iii) an exponential distribution with unit mean. Note that the first two distributions are symmetric about the mean, whereas the last is not. The corresponding box plots are shown in Figure 2.3. In each case, the upper and lower ends of the box represent<sup>1</sup> the upper and lower quartiles. In the case of the uniform distribution, there are no outliers, and therefore, the upper and lower whiskers represent the true maximum and minimum values. On the other hand, there are outliers at the upper end in the case of the normal and exponential distributions. Therefore, the whiskers are placed at  $1.5 \text{ IQR}$  above the upper ends of the boxes in each of the cases.

Many other conventions exist on the placement of whiskers, such as the use of the actual minimum/maximum or the use of particular percentiles of the data distribution. The specific convention used in this book is referred to as the *Tukey box-plot*. Aside from visualizing extreme values, this type of diagram is useful for visualizing the performance of a randomized outlier detection algorithm and is often used in outlier ensemble analysis. We will revisit this issue in section 6.4 of Chapter 6.

## 2.3 Extreme-Value Analysis in Multivariate Data

Extreme-value analysis can also be applied to multivariate data in a variety of ways. Some of these definitions try to model the underlying distribution explicitly, whereas others are based on more general statistical analysis, which does not assume any particular statistical distribution of the underlying data. In this section, we will discuss four different classes of methods that are designed to find data points at the *boundaries of multivariate data*. The first of these classes of methods (depth-based) is not a statistical or probabilistic approach. Rather, it is based on convex hull analysis of the point geometry. However, we have included it in this chapter, because it naturally fits with the other multivariate extreme-value methods in terms of the *types* of outliers it finds.

<sup>1</sup>It is possible to change the percentile levels of the boxes, although the use of quartiles is ubiquitous.



```

Algorithm FindDepthOutliers(Data Set:  $\mathcal{D}$ , Score Threshold:  $r$ );
begin
   $k = 1$ ;
  repeat
    Find set  $S$  of corners of convex hull of  $\mathcal{D}$ ;
    Assign depth  $k$  to points in  $S$ ;
     $\mathcal{D} = \mathcal{D} - S$ ;
     $k = k + 1$ ;
  until ( $\mathcal{D}$  is empty);
  Report points with depth at most  $r$  as outliers;
end

```

Figure 2.4: Pseudocode for finding depth-based outliers

While the methods discussed in this section are effective in finding outliers at the *outer boundaries* of a data space, they are not good at finding outliers within the inner regions of the data space. Such methods can effectively find outliers for the case illustrated in Figure 2.7, but not the outlier ‘A’ illustrated in Figure 2.1. Nevertheless, the determination of such outliers can be useful in many specialized scenarios. For example, in cases where multiple deviation values may be associated with records, multivariate extreme-value analysis may be useful. Consider a weather application in which multiple attributes such as temperature and pressure are measured at different spatial locations, and the local spatial deviations from the expected values are computed as an intermediate step. These deviations from expected values on different attributes may need to be transformed into a single meaningful outlier score. An example is illustrated in section 11.2.1.3 of Chapter 11, where deviations are computed on the different measured values of spatial data. In general, such methods are useful for post-processing a multidimensional vector of outlier scores, in which each outlier score is derived using a different and possibly independent criterion. As discussed in Chapter 1, it is particularly common to confuse methods for extreme-value analysis with general outlier analysis methods that are defined in terms of generative probabilities. However, it is important to distinguish between the two, since the application-specific scenarios in which the two kinds of methods are used are quite different.

### 2.3.1 Depth-Based Methods

In depth-based methods, convex hull analysis is used in order to find outliers. The idea is that the points in the outer boundaries of the data lie at the corners of the convex hull. Such points are more likely to be outliers. A depth-based algorithm proceeds in an iterative fashion. In the  $k$ th iteration, all points at the corners of the convex hull of the data set are removed from the data set. These points are assigned a depth of  $k$ . These steps are repeated until the data set is empty. All points with depth at most  $r$  are reported as the outliers. Alternatively, the depth of a data point may be directly reported as the outlier score. The steps of the depth-based approach are illustrated in Figure 2.4.

The algorithm is also pictorially illustrated on a sample data set in Figure 2.5. A number of efficient methods for finding depth-based outliers have been discussed in [295, 468]. The computational complexity of convex-hull methods increases exponentially with dimensionality. Furthermore, with increasing dimensionality, a larger proportion of data points lie at

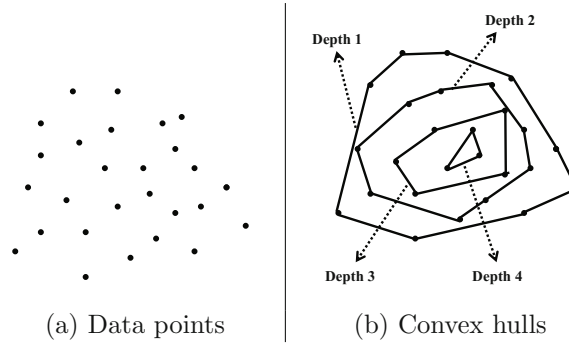


Figure 2.5: Depth-based outlier detection

the corners of a convex hull. This is because the number of points at the corners of a convex hull can be exponentially related to the data dimensionality. Therefore, such methods are not only computationally impractical, but also increasingly ineffectual in higher dimensionality because of increasing ties in outlier scores. Depth-based methods are generally quite different from most of the probabilistic and statistical models discussed in this chapter. In fact, they cannot really be considered probabilistic or statistical methods. However, they are presented here because of their relationship to other multivariate extreme-value methods. Such methods share many characteristics in common, in spite of being methodologically different. For example, they work well only in scenarios where outliers lie at the boundaries of data space, rather than as isolated points in the interior of the data.

### 2.3.2 Deviation-Based Methods

Deviation-based methods measure the impact of outliers on the data variance. For example, the method proposed in [62] tries to measure how much the variance in the underlying data is reduced, when a particular data point is removed. Since the basic assumption is that the outliers lie at the boundary of the data, it is expected that the removal of such data points will significantly reduce the variance. This is essentially an *information-theoretic* method, since it examines the reduction in complexity, when a data point is removed. Correspondingly, the *smoothing factor* for a set of data points  $R$  is defined as follows:

**Definition 2.3.1** *The smoothing factor  $SF(R)$  for a set  $R$  is the reduction in the data set variance, when the set of points in  $R$  are removed from the data.*

Outliers are defined as exception sets  $E$  such that their removal causes the maximum reduction in variance of the data. In other words, for *any* subset of data points  $R$ , it must be the case that:

$$SF(E) \geq SF(R)$$

If more than one set have the same reduction in variance, then the smaller set is preferred. This follows the standard information theoretic principle of finding the sets that increase the description length of the data as much as possible, in as little space. The determination of the optimal set  $E$  is a very difficult problem, because  $2^N$  possibilities exist for a data set containing  $N$  points. The work in [62] uses a number of heuristics such as best-first search and random sampling. One good aspect of this approach is that it is distribution-independent, and can be applied to any kind of data set, as long as an appropriate definition

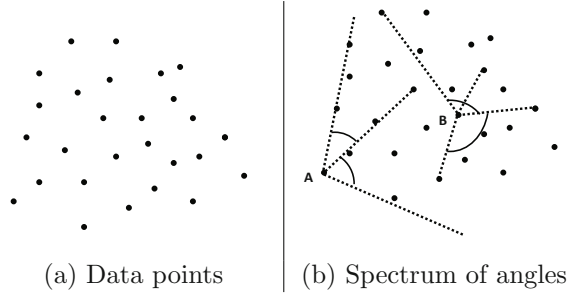


Figure 2.6: Angle-based outlier detection

of the smoothing factor can be constructed. In the original work in [62], this approach has been applied to the case of sequence data.

### 2.3.3 Angle-Based Outlier Detection

This method was originally proposed as a general outlier analysis method, although this book has reclassified it to a multivariate extreme-value analysis method. The idea in angle-based methods is that data points at the boundaries of the data are likely to enclose the entire data within a smaller angle, whereas points in the interior are likely to have data points around them at different angles. For example, consider the two data points ‘A’ and ‘B’ in Figure 2.6, in which point ‘A’ is an outlier, and point ‘B’ lies in the interior of the data. It is clear that all data points lie within a limited angle centered at ‘A.’ On the other hand, this is not the case for data point ‘B,’ which lies within the interior of the data. In this case, the angles between different pairs of points can vary widely. In fact, the more isolated a data point is from the remaining points, the smaller the underlying angle is likely to be. Thus, data points with a smaller angle spectrum are outliers, whereas those with a larger angle spectrum are not outliers.

Consider three data points  $\bar{X}$ ,  $\bar{Y}$ , and  $\bar{Z}$ . Then, the angle between the vectors  $\bar{Y} - \bar{X}$  and the  $\bar{Z} - \bar{X}$ , will not vary much for different values of  $\bar{Y}$  and  $\bar{Z}$ , when  $\bar{X}$  is an outlier. Furthermore, the angle is inversely weighted by the distance between the points. The corresponding angle (weighted cosine) is defined as follows:

$$WCos(\bar{Y} - \bar{X}, \bar{Z} - \bar{X}) = \frac{\langle \bar{Y} - \bar{X}, \bar{Z} - \bar{X} \rangle}{\|\bar{Y} - \bar{X}\|_2 \cdot \|\bar{Z} - \bar{X}\|_2}$$

Here,  $\|\cdot\|_2$  represents the  $L_2$ -norm, and  $\langle \cdot \rangle$  represents the scalar product. Note that this is a weighted cosine, since the denominator contains the squares of the  $L_2$ -norms. The inverse weighting by the distance further reduces the weighted angles for outlier points, which also has an impact on the spectrum of angles. Then, the *variance in the spectrum* of this angle is measured by varying the data points  $\bar{Y}$  and  $\bar{Z}$ , while keeping the value of  $\bar{X}$  fixed. Correspondingly, the *angle-based outlier factor (ABOF)* of the data point  $\bar{X} \in \mathcal{D}$  is defined as follows:

$$ABOF(\bar{X}) = Var_{\{Y, Z \in \mathcal{D}\}} WCos(\bar{Y} - \bar{X}, \bar{Z} - \bar{X})$$

Data points that are outliers will have a smaller spectrum of angles, and will therefore have lower values of the angle-based outlier factor  $ABOF(\bar{X})$ .

The angle-based outlier factor of the different data points may be computed in a number of ways. The naive approach is to pick all possible triples of data points and compute the  $O(N^3)$  angles between the different vectors. The ABOF values can be explicitly computed from these values. However, such an approach can be impractical for very large data sets. A number of efficiency-based optimizations have therefore been proposed.

In order to speed up the approach, a natural possibility is to use sampling in order to approximate this value of the angle-based outlier factor. A sample of  $k$  data points can be used in order to approximate the ABOF of a data point  $\bar{X}$ . One possibility is to use an unbiased sample. However, since the angle-based outlier factor is inversely weighted by distances, it follows that the nearest neighbors of a data point have the largest contribution to the angle-based outlier factor. Therefore, the  $k$ -nearest neighbors of  $\bar{X}$  can be used to approximate the outlier factor much more effectively than an unbiased sample of the all the data points. It has also been shown in [325] that many data points can be filtered out on the basis of approximate computation, since their approximate values of the ABOF are too high, and they cannot possibly be outliers. The exact values of the ABOF are computed only for a small set of points, and the points with the lowest values of the ABOF are reported as outliers. We refer the reader to [325] for the details of these efficiency optimizations.

Because of the inverse weighting by distances, the angle-based outlier analysis method can be considered a hybrid between distance-based and angle-based methods. As discussed earlier with the use of the illustrative example, the latter factor is primarily optimized to finding multivariate extreme values in the data. The precise impact of each of these factors<sup>2</sup> does not seem to be easily quantifiable in a statistically robust way. In most data sets such as in Figure 2.1, outliers lie not just on the boundaries of the data, but also in the interior of the data. Unlike extreme values, outliers are defined by generative probabilities. While the distance factor can provide some impact for the outliers in the interior, the work is primarily focused on the advantage of angular measures, and it is stated in [325] that the degree of impact of distance factors is minor compared to the angular factors. This implies that outliers on the boundaries of the data will be highly favored in terms of the overall score, because of the lower spectrum of angles. Therefore, the angle-based method treats outliers with similar generative probabilities in the interior and the boundaries of the data in a differential way, which is not statistically desirable for general outlier analysis. Specifically, the outliers at the boundaries of the data are more likely to be favored in terms of the outlier score. Such methods can effectively find outliers for the case illustrated in Figure 2.7, but the outlier ‘A’ illustrated in Figure 2.1 will be favored less. Therefore, while this approach was originally presented as a general outlier analysis method, it has been classified in the section on multivariate extreme-value analysis methods in this book.

It has been claimed in [325] that the approach is more suitable for high-dimensional data because of its use of angles, as opposed to distances. However, it has been shown in earlier work [455], that angle-based measures are not immune to the dimensionality curse, because of concentration effects in the cosine measure. Such concentration effects would also impact the spectrum of the angles, even when they are combined with distances. The variation in the angle spectrum in Figure 2.6 is easy to show visually in 2-dimensional data, but the sparsity effects will also impact the spectrum of angles in higher dimensions. If the main problem, as suggested in [325], is the lack of contrast between pairwise distances, then this is not resolved with the use of angles instead of distances. In a setting where all pairs of distances are similar, all triangles will be equilateral, and therefore all (cosines of) angles

---

<sup>2</sup>When a random variable is scaled by a factor of  $a$ , its variance is scaled by a factor of  $a^2$ . However, the scaling here is not by a constant factor.

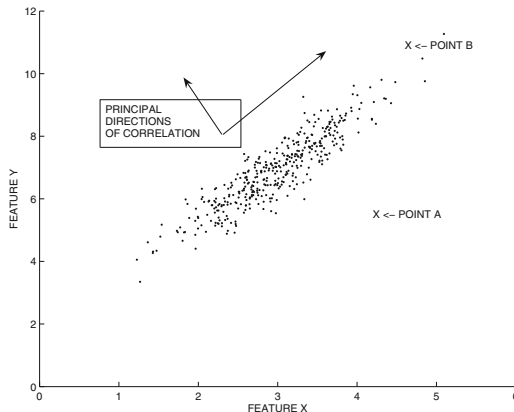


Figure 2.7: Extreme-value analysis in multivariate data with Mahalanobis distance

will converge to 0.5. In fact, the cosine can be shown to be a direct function of Euclidean pairwise distances:

$$\text{Cosine}(\bar{X}, \bar{Y}) = \frac{\|X - 0\|^2 + \|Y - 0\|^2 - \|X - Y\|^2}{2 \cdot \|X - 0\| \cdot \|Y - 0\|} \quad (2.19)$$

If distances retain little information about the relative contrasts, there is little reason to believe that an *indirect* function of the distances (like the cosine spectrum) will do any better. A clear explanation of why the spectrum of angles should be more robust to high dimensionality than distances has not<sup>3</sup> been provided in [325]. More importantly, such methods do not address the issue of locally irrelevant attributes [4], which are the primary impediment to effective outlier analysis methods with increasing dimensionality. Another important point to note is that multivariate extreme-value analysis is much simpler than general outlier analysis in high dimensionality, because the parts of the data to explore are approximately known, and therefore the analysis is global rather than local. The evidence over different dimensions can be accumulated with the use of a very simple classical distance-distribution method [343, 493]. The approach, discussed in the next section, is also suitable for high-dimensional extreme-value analysis, because it implicitly weights globally relevant and irrelevant directions in the data in a different way, and is statistically sound in terms of probabilistic interpretability of the extreme values.

### 2.3.4 Distance Distribution-based Techniques: The Mahalanobis Method

A *distribution-dependent* approach is to model the entire data set to be normally distributed about its mean in the form of a multivariate Gaussian distribution. Let  $\bar{\mu}$  be the

<sup>3</sup>The use of the cosine function in some high-dimensional domains such as text has been cited as an example in a later work [326]. In domains with small and varying non-zero attributes, the cosine is preferred because of important normalization properties, and not because of greater dimensionality resistance. By substituting  $\|\bar{X}\| = \|\bar{Y}\| = 1$  in Equation 2.19, it is evident that the cosine is equivalent to the Euclidean distance if all points are normalized to lie on a unit ball. The cosine function is not immune to the dimensionality curse even for the unique structure of text [455]. An increasing fraction of non-zero attributes, towards more general distributions, directly impacts the data hubness.

$d$ -dimensional mean (row) vector of a  $d$ -dimensional data set, and  $\Sigma$  be its  $d \times d$  covariance matrix. In this case, the  $(i, j)$ th entry of the covariance matrix is equal to the covariance between the dimensions  $i$  and  $j$ . Then, the probability distribution  $f(\bar{X})$  for a  $d$ -dimensional (row vector) data point  $\bar{X}$  can be defined as follows:

$$f(\bar{X}) = \frac{1}{\sqrt{|\Sigma|} \cdot (2 \cdot \pi)^{(d/2)}} \cdot \exp \left[ -\frac{1}{2} \cdot (\bar{X} - \bar{\mu})\Sigma^{-1}(\bar{X} - \bar{\mu})^T \right] \quad (2.20)$$

The value of  $|\Sigma|$  denotes the determinant of the covariance matrix. We note that the term in the exponent is (half) the squared *Mahalanobis distance* of the data point  $\bar{X}$  to the centroid  $\bar{\mu}$  of the data. This term is used as the outlier score and may be directly computed as follows:

$$\text{Mahalanobis}(\bar{X}, \bar{\mu}, \Sigma) = \sqrt{(\bar{X} - \bar{\mu})\Sigma^{-1}(\bar{X} - \bar{\mu})^T} \quad (2.21)$$

The computation of the Mahalanobis distance requires the inversion of the covariance matrix  $\Sigma$ . In cases where the matrix  $\Sigma$  is not invertible, it is possible to use *regularization* with a  $d \times d$  identity matrix  $I$ . The basic idea is to replace  $\Sigma$  with  $\Sigma + \lambda I$  for some small value of  $\lambda > 0$  in Equation 2.21. Here,  $\lambda > 0$  represents the regularization parameter.

The Mahalanobis distance of a point is similar to its Euclidean distance from the centroid of the data, except that it normalizes the data on the basis of the inter-attribute correlations. For example, if the axis system of the data were to be rotated to the principal directions (shown in Figure 2.7), then the data would have no inter-attribute correlations. As we will see in section 3.3 of Chapter 3, it is actually possible to determine such directions of correlations generally in  $d$ -dimensional data sets with the use of principal component analysis (PCA). The Mahalanobis distance is simply equal to the Euclidean distance between  $\bar{X}$  and  $\bar{\mu}$  in such a transformed (axes-rotated) data set *after* dividing each of the transformed coordinate values by the standard-deviation of that direction. Therefore, principal component analysis can also be used in order to compute the Mahalanobis distance (see section 3.3.1 of Chapter 3).

This approach recognizes the fact that the different directions of correlation have different variance, and the data should be treated in a statistically normalized way along these directions. For example, in the case of Figure 2.7, the data point ‘A’ can be more reasonably considered an outlier than data point ‘B,’ on the basis of the natural correlations in the data. On the other hand, the data point ‘A’ is closer to the centroid of the data (than data point ‘B’) on the basis of *Euclidean distance*, but not on the basis of the Mahalanobis distance. Interestingly, data point ‘A’ also seems to have a much higher spectrum of angles than data point ‘B,’ at least from an average sampling perspective. This implies that, at least on the basis of the primary criterion of angles, the angle-based method would incorrectly favor data point ‘B.’ This is because it is unable to account for the relative relevance of the different directions, an issue that becomes more prominent with increasing dimensionality. The Mahalanobis method is robust to increasing dimensionality, because it uses the covariance matrix in order to summarize the high dimensional deviations in a statistically effective way. It is noteworthy that the Mahalanobis method should *not* be merely considered an extreme-value method. In fact, as section 3.3.1 shows, its correlation-sensitive characteristics are more powerful than its extreme-value characteristics.

We further note that each of the distances along the principal correlation directions can be modeled as a 1-dimensional standard normal distribution, which is approximately independent from the other orthogonal directions of correlation. As discussed earlier in this chapter, the sum of the squares of  $d$  variables drawn independently from a standard normal distribution, will result in a variable drawn from a  $\chi^2$ -distribution with  $d$  degrees of freedom.

Therefore, the cumulative probability distribution tables of the  $\chi^2$  distribution can be used in order to determine the outliers with the appropriate level of significance.

### 2.3.4.1 Strengths of the Mahalanobis Method

Although the Mahalanobis method seems simplistic at first sight, it is easy to overlook the fact that the Mahalanobis method accounts for the inter-attribute dependencies in a graceful way, which become particularly important in high-dimensional data sets. This simple approach turns out to have several surprising advantages over more complex distance-based methods in terms of accuracy, computational complexity, and parametrization:

1. It is short-sighted to view the Mahalanobis method only as a multivariate extreme-value analysis method because most of its power resides in its use of inter-attribute correlations. The use of the covariance matrix ensures that inter-attribute dependencies are accounted for in the outlier detection process. In fact, as discussed in Chapter 3, one can view the Mahalanobis method as a soft version of PCA. Although it is not immediately obvious, the merits of some of the sophisticated linear models such as one-class support-vector machines (SVMs)<sup>4</sup> and matrix factorization are inherently built into the approach. In this sense, the Mahalanobis method uses a more powerful model than a typical multivariate extreme-value analysis method. A detailed discussion of the connections of PCA with the Mahalanobis method and its nonlinear extension is provided in Chapter 3. Aside from its PCA-based interpretation, it also has a natural probabilistic interpretation as a special case of the EM-method discussed in the next section.
2. The Mahalanobis method is *parameter-free*. This is important in unsupervised problems like outlier detection, in which there is no meaningful way of setting the parameters by testing its performance on the data set. This is because ground-truth is not available for parameter tuning.
3. The features in real data sets are often extracted in such a way that extremes in values expose the outliers, which is an easy case for the Mahalanobis method. If the analyst has an intuitive understanding that extremes in (many of the) feature values are indicative of outliers, then the Mahalanobis method can sometimes be used confidently. Even in cases where all features do not show this characteristic, the natural aggregation effects in the Mahalanobis distance are able to expose the outliers. At the very least, one can leverage the Mahalanobis method as one of the components of an ensemble method (cf. Chapter 6) to exploit the subset of features that are friendly to extreme-value analysis. A simple combination of a nearest-neighbor detector and the Mahalanobis method can perform surprisingly robustly as compared to a variety of other complex detectors. The addition of a distance-based component to an ensemble method also ensures that outliers like data point ‘A’ in Figure 2.1 are not missed completely.
4. As discussed later in Chapter 4, most distance-based methods require  $O(N^2)$  time for a data set containing  $N$  points. Even for data sets containing a few hundred thousand points, it often becomes computationally challenging to compute the outlier

---

<sup>4</sup>Some one-class SVMs [538, 539] learn a circular separator wrapped around the centroid of the data, albeit in a transformed kernel space. As discussed in the next chapter, it is also possible to kernelize the Mahalanobis method because of its PCA-based interpretation. Furthermore, the solutions in the two cases can be shown to be closely related (cf. section 3.4.3).

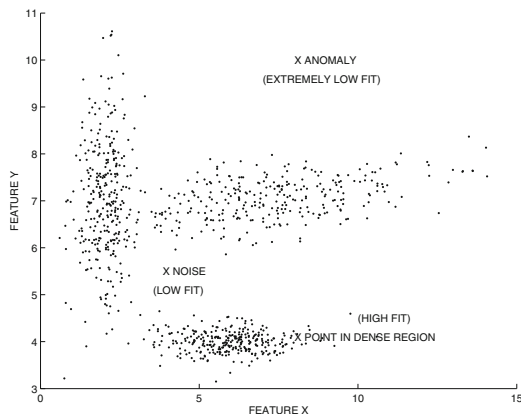


Figure 2.8: Relating fit probabilities to anomalous behavior

scores. On the other hand, the Mahalanobis method is *linear* in the number of data points, although it does require at least quadratic time and space in terms of the data *dimensionality*. Nevertheless, since the number of points is typically orders of magnitude greater than the number of dimensions, the Mahalanobis method has a significant advantage in terms of computational time in most real-world settings.

Therefore, the Mahalanobis method can often be used as an *additive component* of ensemble methods, even when it is not desirable to use it on a stand-alone basis.

## 2.4 Probabilistic Mixture Modeling for Outlier Analysis

---

The previous section was focused on the problem of extreme-value analysis for outlier modeling. The simple Mahalanobis method is effective for the example of Figure 2.7, because the entire data set is distributed in one large cluster about the mean. For cases in which the data may have many different clusters with different orientations, such an extreme-value approach may not be effective. An example of such a data set is illustrated in Figure 2.1. For such cases, more general distribution-based modeling algorithms are needed.

The key idea in this generalization is to use probabilistic mixture modeling of the data points. Such models are typically *generative* models, where for each data point, we can estimate the generative probability (or the fit probability) to the model. First, we assume a specific form of the generative model (e.g., a mixture of Gaussians), and then estimate the parameters of this model with the use of the expectation-maximization (EM) algorithm. The parameters are estimated so that the observed data has a *maximum likelihood fit* to the generative model. Given this model, we then estimate the generative probabilities (or fit probabilities) of the underlying data points. Data points that fit the distribution will have high fit probabilities, whereas anomalies will have very low fit probabilities. Some examples of how different types of data points relate to the fit probability in such a model are illustrated in Figure 2.8.

The broad principle of a mixture-based generative model is to *assume* that the data was generated from a mixture of  $k$  distributions with the probability distributions  $\mathcal{G}_1 \dots \mathcal{G}_k$  with the repeated application of the following stochastic process:



- Select the  $r$ th probability distribution with probability  $\alpha_r$ , where  $r \in \{1 \dots k\}$ .
- Generate a data point from  $\mathcal{G}_r$ .

We denote this generative model by  $\mathcal{M}$ . The value  $\alpha_r$  indicates the prior probability, and intuitively represents the fraction of the data generated from mixture component  $r$ . We note that the different values of  $\alpha_r$ , and the parameters of the different distributions  $\mathcal{G}_r$  are not known in advance, and they need to be learned in a data-driven manner. In some simplified settings, the values of the prior probabilities  $\alpha_r$  may be fixed to  $1/k$ , although these values also need to be learned from the observed data in the most general case. The most typical form of the distribution  $\mathcal{G}_r$  is the Gaussian distribution. The parameters of the distribution  $\mathcal{G}_r$  and the prior probabilities  $\alpha_r$  need to be *estimated* from the data, so that the data has the maximum likelihood fit of being generated. Therefore, we first need to define the concept of the fit of the data set to a particular component of the mixture. Let us assume that the density function of  $\mathcal{G}_r$  is given by  $f^r(\cdot)$ . The probability (density function) of the data point  $\bar{X}_j$  being generated by the model is given by the following:

$$f^{point}(\bar{X}_j|\mathcal{M}) = \sum_{i=1}^k \alpha_i \cdot f^i(\bar{X}_j) \quad (2.22)$$

Then, for a data set  $\mathcal{D}$  containing  $N$  records denoted by  $\bar{X}_1 \dots \bar{X}_N$ , the probability of the data set being generated by the model  $\mathcal{M}$  is the product of the corresponding individual point-wise probabilities (or probability *densities*):

$$f^{data}(\mathcal{D}|\mathcal{M}) = \prod_{j=1}^N f^{point}(\bar{X}_j|\mathcal{M}) \quad (2.23)$$

The log-likelihood fit  $\mathcal{L}(\mathcal{D}|\mathcal{M})$  of the data set  $\mathcal{D}$  with respect to  $\mathcal{M}$  is the logarithm of the aforementioned expression and can be (more conveniently) represented as a sum of values over the different data points.

$$\mathcal{L}(\mathcal{D}|\mathcal{M}) = \log \left[ \prod_{j=1}^N f^{point}(\bar{X}_j|\mathcal{M}) \right] = \sum_{j=1}^N \log \left[ \sum_{i=1}^k \alpha_i \cdot f^i(\bar{X}_j) \right] \quad (2.24)$$

This log-likelihood fit needs to be optimized to determine the model parameters, and therefore maximize the fit of the data points to the generative model. The log-likelihood fit is preferable to the likelihood fit because of its additive nature across different data points, and its numerical convenience.

It is noteworthy that it is much easier to determine the optimal model parameters separately for each component of the mixture, if we knew (at least probabilistically), which data point was generated by which component of the mixture. At the same time, the probability of generation of these different data points from different components is dependent on these optimal model parameters. This circularity in dependence naturally suggests an iterative EM-algorithm in which the model parameters and probabilistic data point assignments to components are iteratively refined and estimated from one another. Let  $\Theta$  be a vector representing the *entire set* of parameters describing all components of the mixture model. For example, in the case of the Gaussian mixture model,  $\Theta$  would contain all the component mixture means, variances, co-variances, and the parameters  $\alpha_1 \dots \alpha_k$ . Then, the EM-algorithm starts off with an initial set of values of  $\Theta$  (possibly corresponding to random assignments of data points to mixture components), and proceeds as follows:

- **(E-step):** Given current value of the parameters in  $\Theta$ , determine the *posterior* probability  $P(\overline{X}_j | \mathcal{G}_r, \Theta)$  that the point  $\overline{X}_j$  was generated by the  $r$ th mixture component. This computation is performed for all point-component pairs  $(\overline{X}_j, \mathcal{G}_r)$ .
- **(M-step):** Given current probabilities of assignments of data points to clusters, use maximum likelihood approach to determine the value of all the parameters  $\Theta$ , which maximizes the log-likelihood fit on the basis of current assignments. Therefore, in the Gaussian setting, all cluster means, covariance matrices, and prior probabilities  $\alpha_1 \dots \alpha_k$  need to be estimated.

It now remains to explain the details of the E-step and the M-step. The E-step simply computes the probability density of the data point  $\overline{X}_j$  being generated by each component of the mixture, and then computes the fractional value for each component. This is defined by the Bayes posterior probability that the data point  $\overline{X}_j$  was generated by component  $r$  (with model parameters fixed to the current set of the parameters  $\Theta$ ). Therefore, we have:

$$P(\mathcal{G}_r | \overline{X}_j, \Theta) = \frac{\alpha_r \cdot f^{r, \Theta}(\overline{X}_j)}{\sum_{i=1}^k \alpha_i \cdot f^{i, \Theta}(\overline{X}_j)} \quad (2.25)$$

With some abuse of notation, a superscript  $\Theta$  has been added to the probability density functions in order to denote the fact that they are evaluated at the current set of model parameters  $\Theta$ .

Next, we describe the parameter estimation of the *M-step*, which maximizes the likelihood fit. In order to optimize the fit, we need to compute the partial derivative of the log-likelihood fit with respect to corresponding model parameters, and set them to 0 in order to determine the optimal value. The values of  $\alpha_r$  are easy to estimate and are equal to the expected fraction of the points assigned to each cluster, based on the current values of  $P(\mathcal{G}_r | \overline{X}_j, \Theta)$ . In practice, in order to obtain more robust results for smaller data sets, the expected number of data points belonging to each cluster in the numerator is augmented by 1, and the total number of points in the denominator is  $N + k$ . Therefore, the estimated value of  $\alpha_r$  is  $(1 + \sum_{j=1}^N P(\mathcal{G}_r | \overline{X}_j, \Theta)) / (k + N)$ . This approach is a form of regularization, and it is also referred to as *Laplacian smoothing*. For example, if  $N$  is extremely small, such an approach pushes the assignment probability towards  $1/k$ . This represents a natural *prior* assumption about the distribution of points in clusters.

In order to determine the other parameters specific to a particular component  $r$  of the mixture, we simply treat each value of  $P(\mathcal{G}_r | \overline{X}_j, \Theta)$  as a weight of that data point in that component, and then perform maximum likelihood estimation of the parameters *of that component*. This is generally a much simpler process than having to deal with all components of the mixture at one time. The precise estimation process depends on the probability distribution at hand. For example, consider a setting in which the  $r$ th Gaussian mixture component in  $d$  dimensions is represented by the following distribution:

$$f^{r, \Theta}(\overline{X}_j) = \frac{1}{\sqrt{|\Sigma_r|} \cdot (2 \cdot \pi)^{(d/2)}} \cdot \exp \left[ -\frac{1}{2} \cdot (\overline{X}_j - \overline{\mu}_r) \Sigma_r^{-1} (\overline{X}_j - \overline{\mu}_r)^T \right] \quad (2.26)$$

Here,  $\overline{\mu}_r$  is the  $d$ -dimensional mean vector and  $\Sigma_r$  is the  $d \times d$  co-variance matrix of the generalized Gaussian distribution of the  $r$ th component. The value of  $|\Sigma_r|$  denotes the determinant of the covariance matrix. When the number of mixture components is large, the non-diagonal entries are often set to 0 in order to reduce the number of estimated parameters. In such cases, the determinant of  $\Sigma_r$  simplifies to the product of the variances along the individual dimensions.

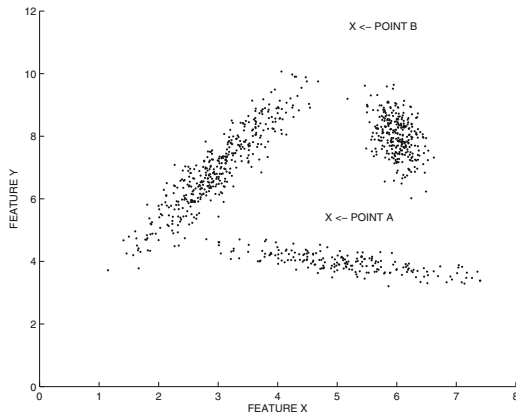


Figure 2.9: EM-Algorithm can determine clusters with arbitrary correlations (Revisiting Figure 2.1)

It can be shown that the maximum-likelihood estimation of  $\overline{\mu}_r$  and  $[\Sigma_r]_{ij}$  are equal to the (probabilistically weighted) means and co-variances of the data points in that component. Recall that these probabilistic weights were derived from the assignment probabilities in the E-step. Thus, the E-step and the M-step depend on each other and can be probabilistically executed to convergence in order to determine the optimum parameter values  $\Theta$ .

At the end of the process, we have a probabilistic model that describes the entire data set as the observed output of a generative process. This model also provides a probabilistic fit value for each data point in the form of Equation 2.22. This value provides the outlier score. Thus, we can use this fit in order to rank all the data points, and determine the most anomalous ones. The idea is that points that are far away from the dense regions in the data (such as the one shown in the upper region of Figure 2.8) will have very low fit values. These points are the anomalies in the data. If desired, statistical hypothesis testing can be applied for identification of outliers with unusually low fit values. However, for statistical testing, the logarithm function should be applied to the fit values (i.e., log-likelihood fits should be used) to reduce the relative variance of inliers (large fit values), so that points with very low fit values will pass an extreme-value test.

The approach requires the number of mixture components as an input parameter. In some cases, domain-specific insights about the data can be used to make meaningful choices. In cases where such insights are not available, an ensemble of mixture models with different parameter settings is useful [184]. In particular, the work in [184] averages the point-wise log-likelihood scores obtained on models with different numbers of mixture components. Furthermore, these models are built on different samples of the data set. Excellent results have been reported using this approach.

### 2.4.1 Relationship with Clustering Methods

Probabilistic mixture modeling is a stochastic version of clustering methods, which can also be used for outlier detection (cf. Chapter 4). It is noteworthy that the fit values in a Gaussian mixture model use the distances of points from cluster centroids in the exponent of the Gaussian. Therefore, the log-likelihood fit of a single Gaussian is the Mahalanobis distance, although the additive fits from multiple Gaussians cannot be simplified in this

manner. Nevertheless, the effect of the nearest cluster is often predominant in the fit values. In the clustering models of Chapter 4, only the distance to the *nearest* cluster centroid is used directly as the outlier score. Therefore, the clustering techniques of Chapter 4 can be viewed as hard versions of the EM-algorithm in which a specific (nearest) cluster is used for scoring the points rather than using the combined fit values from all the clusters in a soft probabilistic combination.

The EM-algorithm can identify arbitrarily oriented clusters in the data, when the clusters have elongated shapes in different directions of correlation. This can help in better identification of outliers. For example, in the case of Figure 2.9, the fit of point ‘A’ would be lower than that for point ‘B,’ even though point ‘B’ is closer to a cluster on the basis of absolute distances. This is because the Mahalanobis distance in the exponent of the Gaussian normalizes for the distances along the different directions of correlation in the data. Indeed, data point ‘A’ is more obviously an outlier.

## 2.4.2 The Special Case of a Single Mixture Component

Interestingly, the special case in which the mixture distribution contains a single Gaussian component (cf. Equation 2.26) works surprisingly well in real settings. This is in part because using a single Gaussian component corresponds to the Mahalanobis method of section 2.3.4. The specific merits of this method are discussed in section 2.3.4.1. As we will see in Chapter 3, this leads to a soft version of Principal Component Analysis (PCA), which is known to be effective because of its ability to identify data points that violate interattribute dependencies. The Mahalanobis method can therefore be explained both from the point of view of probabilistic methods and linear models.

Although the use of a single mixture component seems to miss true outliers (like the outlier ‘A’ of Figure 2.9), it also has the advantage that none of the mixture components can overfit a small but tightly-knit cluster of outliers. When a larger number of mixture components are used, one of the components might correspond to a small tightly knit group of outliers like the outlier cluster illustrated in Figure 2.10. The Mahalanobis method will correctly label the points in this cluster as outliers, whereas a mixture model (with a larger number of components) runs the risk of modeling this small cluster as a legitimate mixture component. Interesting anomalies often occur in small clusters because they might have been caused by similar underlying causes (e.g., a specific disease or type of credit-card fraud). The Mahalanobis method is able to identify such clusters as outliers because they are often inconsistent with the global mean and covariance structure of the data. Because of these typical characteristics of real data sets, very simple methods like the Mahalanobis method sometimes outperform significantly more complex models.

As discussed in section 3.3.8 of the next chapter, one can also combine the Mahalanobis method with kernel methods to model more general distributions. For example, some variations of these methods can correctly identify the outlier ‘A’ in Figure 2.9. An ensemble-centric version of this approach has been shown to provide high-quality results [35].

## 2.4.3 Other Ways of Leveraging the EM Model

The EM model discussed in the previous section quantifies the outlier score as the fit of the point to any of the mixture components. Therefore, all mixture components are assumed to be instances of the normal class. A different approach is one in which some domain-specific insights are available about the differences in distribution of the normal classes and the anomalous class. In such a case, different probability distributions are used to model the

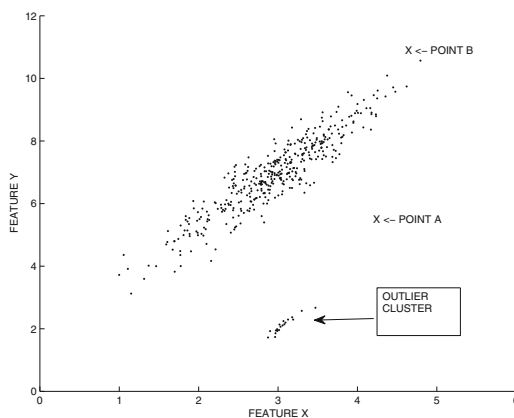


Figure 2.10: The use of a single mixture component is robust to the presence of small outlier clusters

normal and anomalous classes. The outlier score of a point is quantified as its fit to the anomalous class and larger scores are indicative of anomalies. This approach is generally difficult to use in the absence of specific insights about the differences in distribution of the normal and anomalous classes. For example, such an approach has been used for the identification of particular types of outliers such as noise [110]. In the next section, we will provide another example of a setting in which it is possible to model normal and outlier classes with realistic (and different) distributions.

#### 2.4.4 An Application of EM for Converting Scores to Probabilities

Interestingly, EM algorithms can also be used as a final step after many such outlier detection algorithms for converting the scores into probabilities [213]. Note that the fit value returned by the EM algorithm of the previous section (cf. Equation 2.22) is a probability *density* value, and cannot be interpreted as a numerical probability. The ability to characterize an outlier in terms of numerical probabilities is a very useful step for intuition and interpretability.

The idea is that the distribution of the scores can be treated as a univariate data set, which can then be fit to a probabilistic generative model. In this case, the outlier points are explicitly assumed to belong to a component of the mixture model (rather than simply treating them as points with low fit values). Note that one can differentiate the outlier and non-outlier classes in this setting only if some additional insight is available about the natural distributions of the outlier and non-outlier classes. Therefore, different types of distributions can be used to model the outlier and non-outlier components. The work in [213] uses a bimodal mixture of exponential and Gaussian functions. The assumption is that the non-outlier points are distributed according to the exponential distribution, whereas the outlier points are distributed according to the Gaussian distribution. This assumption is made on the basis of the “typical” behavior of outlier scores in real applications. The EM algorithm is used to compute the parameters of each component of the mixture distribution, and the corresponding prior probabilities of assignment. These can be used in order to convert the outlier scores into probabilities with the use of the Bayes rule, since it is now

possible to compute the posterior probability (see Equation 2.25) that the data point belongs to the outlier component. We note that the assignment of a component of the mixture to the outlier class is critical in being able to estimate the probability that a data point is an outlier, which is facilitated by the difference in the distributions (Gaussian versus exponential) of outliers scores in the two classes.

## 2.5 Limitations of Probabilistic Modeling

---

Parametric methods are very susceptible to noise and overfitting in the underlying data. Mixture models always assume a specific distribution of the data, and then try to learn the parameters of this distribution. A natural trade-off exists between the generality of this distribution and the number of parameters that need to be learned. If this trade-off is not calibrated carefully, then one of the following two scenarios could occur:

- When the particular assumptions of the model are inaccurate (e.g., inappropriate use of Gaussian distribution), the data is unlikely to fit the model well. As a result, a lot of spurious data points may be reported as outliers.
- When the model is too general, the number of parameters to describe the model increases. For example, when one uses an inappropriately large number of mixture components, a small but tightly-knit outlier cluster may fit one of the mixture components. An example of such a small cluster is illustrated in Figure 2.10. In fact, the technique in the previous section of converting scores to probabilities leverages the fact that univariate outlier scores often cluster together in a Gaussian distribution. Unfortunately, there is no way of generalizing this approach easily to multidimensional data sets. As a result, when reporting points of *low* fit as outliers (rather than a specially modeled outlier class), it is always possible to miss the true outliers as a result of the overfitting caused by small clusters of outliers. One possibility for reducing overfitting is to fix the prior probabilities to  $1/k$ , although such assumptions might sometimes result in under-fitting.

The proper selection of simplifying assumptions is always tricky. For example, the clusters in the data may be of arbitrary shape or orientation, and may not fit a simplified Gaussian assumption in which the data values along different dimensions are independent of one another. This corresponds to setting the non-diagonal entries of  $\Sigma_r$  to 0 in the Gaussian case. In real data sets, significant correlations may exist among the different dimensions. In such cases, one cannot assume that the matrix  $\Sigma_r$  is diagonal, which would necessitate the learning of  $O(d^2)$  parameters for *each cluster*. This can cause overfitting problems when the number of points in the data set is small. On the other hand, efficiency remains a concern in the case of larger data sets, especially if a larger number of parameters are estimated. This is because these methods use the iterative EM algorithm, which needs to scan the entire data step in each iteration of the E- and M-steps. However, these methods are still more efficient than many point-to-point distance-based methods, which require  $O(N^2)$  time for a data set containing  $N$  points. These methods will be discussed in Chapter 4.

Finally, the issue of *interpretability* remains a concern for many parametric methods. For example, consider the generalized Gaussian model, which tries to learn clusters with non-zero covariances. In such a case, it is difficult to intuitively interpret the clusters with the use of these parameters. Correspondingly, it is also difficult to define simple and intuitive rules that provide critical ideas about the underlying outliers. We note that this issue

may not necessarily be a problem for *all* parametric methods. If the parameters are chosen carefully enough, then the final model can be described simply and intuitively. For example, simplified versions of the Gaussian model without co-variances may sometimes be described simply and intuitively in terms of the original features of the data. On the other hand, such simplifications might cause under-fitting and other qualitative challenges. These trade-offs are, however, endemic to almost all outlier detection methods and not just probabilistic models.

## 2.6 Conclusions and Summary

---

In this chapter, a number of fundamental probabilistic and statistical methods for outlier analysis were introduced. Such techniques are very useful for confidence testing and extreme-value analysis. A number of tail inequalities for extreme-value analysis were also introduced. These methods can also be generalized to the multivariate scenario. Extreme-value analysis has immense utility as a final step in converting the scores from many outlier analysis algorithms into binary labels. In many specific applications, such techniques turn out to be very useful even for general outlier analysis. The EM approach for probabilistic mixture modeling of outliers can be viewed as a generalization of the Mahalanobis method. This technique can also be viewed as one of the clustering methods that are commonly used for outlier detection.

## 2.7 Bibliographic Survey

---

The classical inequalities (e.g., Markov, Chebychev, Chernoff, and Hoeffding) are widely used in probability and statistics for bounding the accuracy of aggregation-based statistics. A detailed discussion of these different methods may be found in [407]. A generalization of the Hoeffding's inequality is the McDiarmid's inequality [393], which can be applied to a more general function of the different values of  $X_i$  (beyond a linearly separable sum). The main restriction on this function is that if the  $i$ th argument of the function (i.e., the value of  $X_i$ ) is changed to any other value, the function cannot change by more than  $c_i$ .

The central limit theorem has been studied extensively in probability and statistics [88]. Originally, the theorem was proposed for the case of sums of independent and identically distributed variables. Subsequently, it was extended by Aleksandr Lyapunov to cases where the variables are not necessarily identically distributed [88], but do need to be independent. A weak condition is imposed on these distributions, ensuring that the sum is not dominated by a few of the components. In such a case, the sum of the variables converges to the normal distribution as well. Thus, this is a generalized version of the Central Limit Theorem.

Statistical hypothesis testing has been used widely in the literature in order to determine statistical levels of significance for the tails of distributions [74, 462]. A significant literature exists on hypothesis testing, where the anomalous properties of not just individual data points, but also the collective behavior of groups of data points can be tested. Such techniques are also used in online analytical processing scenarios where the data is organized in the form of data cubes. It is often useful to determine outliers in different portions of a data cube with the use of hypothesis testing [474].

The statistical method for deviation detection with variance reduction was first proposed in [62]. Angle-based methods for extreme-value analysis in multivariate data were proposed in [325]. The multivariate method for extreme-value analysis with the use of the

Mahalanobis distance was proposed in [343, 493]. This technique does not work well when the outliers lie in sparse regions between clusters. A number of depth-based methods have been proposed in [295, 468]. These methods compute the convex hull of a set of data points, and progressively peel off the points at the corners of this hull. The depth of a data point is defined as the order of convex hull that is peeled. These techniques have not found much popularity because they suffer the same drawback as the method of [343] for finding internally located outliers. Furthermore, convex hull computation is extremely expensive with increasing dimensionality. Furthermore, with increasing dimensionality an increasing proportion of the points will lie on the outermost convex hull. Therefore, such methods can only be applied to 2- or 3-dimensional data sets in practice.

It should be noted that the use of probabilistic methods for outlier detection is distinct from the problem of outlier detection in probabilistic or uncertain data [26, 290, 559]. In the former case, the data is uncertain, but the methods are probabilistic. In the latter case, the data itself is probabilistic. The seminal discussion on the EM-algorithm is provided in [164]. This algorithm has a particularly simple form, when the components of the mixture are drawn from the exponential family of distributions. The work in [578] proposed an *online* mixture learning algorithm, which can handle *both* categorical and numerical variables. An interesting variation of the EM-algorithm treats one component of the mixture model specially as an anomaly component [187]. Correspondingly, this component is drawn from a uniform distribution [187], and is also assigned a low a priori probability. Therefore, instead of determining the anomalous points that do not fit any mixture component well, this approach tries to determine the points which fit this special component of the mixture. Such an approach would generally be more effective at modeling noise rather than anomalies, because the special component in the mixture model is likely to model the noise patterns. Finally, a Gaussian mixture model has also been used recently in order to create a global probabilistic model for outlier detection [583].

The EM-algorithm has also been used for clutter removal from data sets [110]. In this case, noise is removed from the data set by modeling the derived data as a mixture of Poisson distributions. We note that the approach in [110] is designed for noise detection, rather than the identification of true anomalies. It was shown in [110] that the improvement in data quality after removal of the clutter (noise) was significant enough to greatly ease the identification of relevant features in the data. The approach of using a special component of the mixture in order to convert the distribution of outlier scores into probabilities has been used in [213]. In addition to the approach discussed in section 2.4.4, a different modeling approach with the use of the logistic sigmoid function is discussed in [213]. Methods for converting outlier scores into probabilities in the supervised scenario have been discussed in [599].

An implicit assumption made by EM methods is that the attributes are conditionally independent of one another once the mixture component has been selected. A probabilistic approach that makes stronger assumptions on attribute interdependence is the *Bayesian Network*. The Bayesian network approach for outlier detection [66] models dependencies among attributes with an off-the-shelf network and uses these dependencies to score points as outliers based on the violations of these dependencies.

## 2.8 Exercises

---

1. [**Upper-Tail Chernoff Bound**] The chapter provides a proof sketch of the upper-tail Chernoff bound, but not the full proof. Work out the full proof of bound on the upper



tail using the lower-tail proof as a guide. Where do you use the fact that  $\delta < 2 \cdot e - 1$ ?

2. Suppose you flip an “unbiased” coin 100 times. You would like to investigate whether the coin is showing anomalous behavior (in terms of not being “unbiased” as claimed). Determine the mean and standard deviation of the random variable representing the number of “tails”, under the assumption of an unbiased coin. Provide a bound on the probability that you obtain more than 90 tails with the use of the (i) Markov Inequality (ii) Chebychev Inequality (iii) Chernoff Upper Tail Bound, (iv) Chernoff Lower Tail Bound and (v) Hoeffding Inequality. [Hint: Either the upper-tail or lower-tail Chernoff bound can be used, depending on which random variable you look at.]
3. Repeat Exercise 2, when you know that the coin is rigged to show “tails” every eight out of nine flips.
4. Use the central limit theorem to approximate the number of tails by a normal distribution. Use the cumulative normal distribution to approximate the probability that the number of “tails” should be more than 90 for both the cases of Exercises 2 and 3.
5. A manufacturing process produces widgets, each of which is 100 feet long, and has a standard deviation of 1 foot. Under normal operation, these lengths are independent of one another.
  - Use the normal distribution assumption to compute the probability that something anomalous is going on in the manufacturing process, if a sampled widget is 101.2 feet long?
  - How would your answer change, if the sampled widget was 96.3 feet long?
6. In the example above, consider the case where 10,000 widgets from the assembly line were sampled, and found to have an average length of 100.05. What is the probability that something anomalous is going on in the manufacturing process?
7. Use MATLAB or any other mathematical software to plot the  $t$ -distribution with 100 degrees of freedom. Superimpose a standard normal distribution on this plot. Can you visually see the difference? What does this tell you?
8. Work out the steps of the EM-algorithm, when all non-diagonal elements of the covariance matrix  $\Sigma$  are set to zero, and each diagonal element in a given component has the same value. Furthermore, the prior probabilities of assignment are equal to  $1/k$ , where  $k$  is the number of mixture components. Now perform the following modifications:
  - Change the E-step, so that each data point is deterministically assigned to the cluster with the highest probability (hard assignment), rather than a soft probabilistic assignment. Under what distance-based conditions does a data point get assigned to a cluster?
  - How does this algorithm relate to the  $k$ -means algorithm?
  - How would your answers change, if all components were constrained to have the same cluster variance?
9. Using the insights gained from Exercise 8, work out how the EM-algorithm with a Gaussian mixture model with a complete set of covariance matrices  $\Sigma_r$ , and a fixed set of priors, relates to a generalized  $k$ -means algorithm. [Hint: Consider the concept of Mahalanobis distance computations for assignments in  $k$ -means. How should the prior probabilities be defined?]

10. Download the KDD Cup 1999 data set from the UCI Machine Learning Repository [203]. Extract the quantitative attributes from the data set. Apply the EM-algorithm with 20 mixture components, when non-diagonal elements are set to 0.
  - Determine the fit of each data point to the learned distribution. Determine the top 10 points with the least fit. Do these data points correspond to intrusion attacks or normal data?
  - Repeat the process while allowing non-zero non-diagonal elements. How does your answer change?
  - Randomly sample 990 points from the data set, and then add the 10 points found in the first case above. Repeat the procedure on this smaller data set. Do you find significant anomalies in terms of fit probabilities? Do the lowest fit probabilities correspond to the same data points as in the first case above?
  - Repeat the same procedure with the second case above.
11. Repeat the first two portions of Exercise 10 on the Ionosphere data set from the UCI Machine Learning Repository. Note that the Ionosphere data set has much higher dimensionality (of quantitative attributes) and smaller number of records. Do you determine the same top-10 anomalies in the two cases? What are the absolute fit probabilities? What does this tell you about applying such algorithms to small and high dimensional data sets?
12. Let  $Z$  be a random variable satisfying  $E[Z] = 0$ , and  $Z \in [a, b]$ .
  - Show that  $E[e^{t \cdot Z}] \leq e^{t^2 \cdot (b-a)^2 / 8}$ .
  - Use the aforementioned result to complete the proof of the Hoeffding inequality.

---

---

## Chapter 3

# Linear Models for Outlier Detection

---

---

“My nature is to be linear, and when I’m not, I feel really proud of myself.” – Cynthia Weil

### 3.1 Introduction

---

The attributes in real data are usually highly correlated. Such dependencies provide the ability to predict attributes from one another. The notions of prediction and anomaly detection are intimately related. Outliers are, after all, values that deviate from expected (or *predicted*) values on the basis of a particular model. Linear models focus on the use of inter-attribute dependencies to achieve this goal. In the classical statistics literature, this process is referred to as *regression modeling*.

Regression modeling is a parametric form of correlation analysis. Some forms of correlation analysis attempt to predict dependent variables from other independent variables, whereas other forms summarize the entire data in the form of latent variables. An example of the latter is the method of *principal component analysis*. Both forms of modeling can be very useful in different scenarios of outlier analysis. The former is more useful for complex data types such as time-series (see Chapters 9 and 11), whereas the latter is more useful for the conventional multidimensional data type. A unified discussion of these two forms of linear modeling also lays the foundations for some of the discussions in later chapters.

The main assumption in linear models is that the (normal) data is embedded in a lower-dimensional subspace. Data points that do not naturally fit this embedding model are, therefore, regarded as outliers. In the case of proximity-based methods, which will be discussed in the next chapter, the goal is to determine specific *regions of the space* in which outlier points behave very differently from other points. On the other hand, in linear methods, the goal is to find *lower-dimensional subspaces*, in which the outlier points behave very differently from other points. This can be viewed as an orthogonal point of view to clustering- or nearest-neighbor methods, which try to summarize the data *horizontally* (i.e., on the rows or data values), rather than *vertically* (i.e., on the columns or dimensions). As will be discussed in the chapter on high-dimensional outlier detection, it is, in principle,

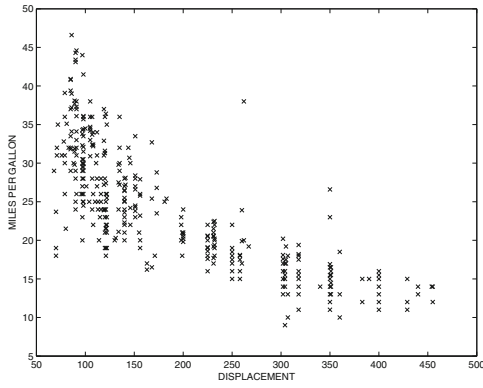
possible to combine these methods to obtain more general local subspace models, which can identify outliers by integrating the horizontal and vertical criteria in a holistic way.

The assumption of linear correlations is a critical leap of faith used by linear models. This may or may not be true for a particular data set, which will have a crucial impact on modeling effectiveness. In order to explain this point, we will use the *Autompg* and *Arrythmia* data sets from the *UCI Machine Learning Repository* [203]. The first data set contains features describing various car measurements and the corresponding mileage (mpg). The second data set contains features derived from electrocardiogram (ECG) readings of human patients.

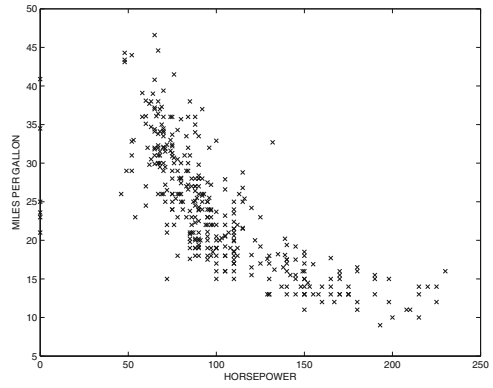
In Figures 3.1(a) and (b), the dependence of the *Miles per Gallon* attribute has been shown on each of the *displacement* and *horsepower* attributes, respectively, for the *Autompg* data set. It is evident that these attributes are highly correlated. Although a significant level of noise is also present in this particular data set, the linear dependence between the attributes is readily apparent. In fact, it can be shown for this data set, that with increasing dimensionality (by selecting more attributes from the data set), the data can be aligned along much lower-dimensional planes. This is also evident in the 3-dimensional plot of Figure 3.1(e). On the other hand, when various views along three of the measured dimensions of the *Arrythmia* data set (Figures 3.1(c), (d) and (f)) are examined, it is evident that the data separates out into two clusters, one of which is slightly larger than the other. Furthermore, it is rather hard to embed this type of data distribution into a lower-dimensional subspace. This data set is more suitable for proximity-based analysis, which will be presented in Chapter 4. The reason for introducing this example is to revisit the point made in the first chapter about the impact of the choices made during the crucial phase of selecting the correct model of normal data. In this particular case, it is evident that the linear model is more appropriate for data sets in which the data naturally aligns along lower-dimensional hyperplanes.

For unsupervised problems like outlier detection, it is hard to guarantee that a specific model of normal data will be effective. For example, in some data sets, different subsets of attributes may be suitable for different models. Such data sets are best addressed with the use of subspace methods discussed in Chapter 5, which can combine the power of row and column selection for outlier analysis. However, in many cases, simplified models such as linear models or proximity-based models are sufficient, without incurring the complexity of subspace methods. From a model-selection perspective, exploratory and visual analysis of the data is rather critical in the first phase of outlier detection in order to find out whether a particular data model is suitable for a particular data set. This is particularly true in the case of unsupervised problems like outlier detection in which ground-truth is not available in order to test the effectiveness of various models.

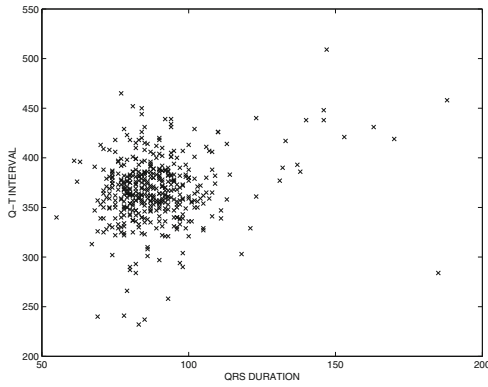
In this chapter, two main classes of linear models will be studied. The first class of models uses statistical regression modeling between dependent and independent variables in order to determine *specific types of dependencies* in the data. Such forms of modeling are more useful when some of the attributes are naturally predictive of others. For example, it is natural to predict the *last* value of a time-series based on a window of previous history of values in the time-series. In such cases, dependency-oriented regression modeling is leveraged by creating a derived data set of dependent and independent variables and quantifying deviations in the observed value of the dependent variable from its predicted value. Even for multidimensional data, we will show in section 7.7 of Chapter 7 that dependency-oriented models for regression can be used to decompose the unsupervised outlier detection problem into  $d$  different regression modeling problems for  $d$ -dimensional data. The second class of models uses principal component analysis to determine the lower-dimensional subspaces



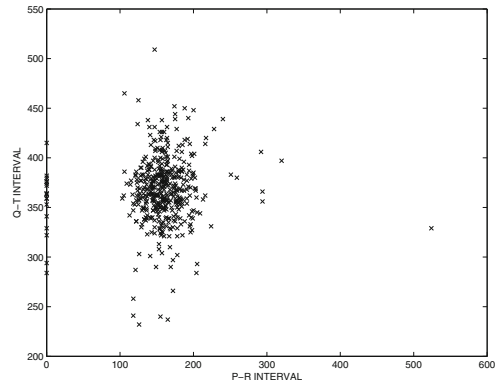
(a) View 1 (*Autmpg*)



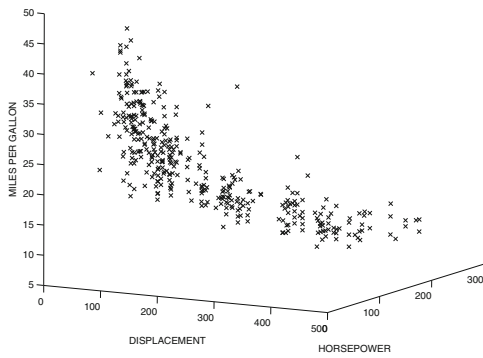
(b) View 2 (*Autmpg*)



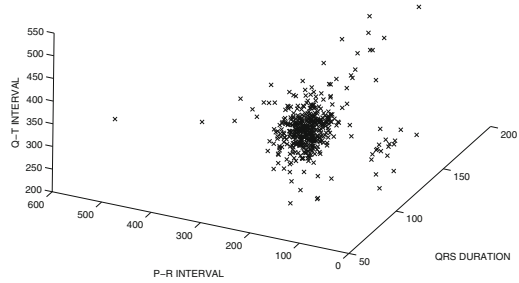
(c) View 1 (*Arrhythmia*)



(d) View 2 (*Arrhythmia*)



(e) 3-d View (*Autmpg*)



(f) 3-d View (*Arrhythmia*)

Figure 3.1: Effectiveness of linear assumption is data-set dependent

of projection. These models are useful for traditional multidimensional data because all attributes are treated in a homogeneous way with no distinction between dependent and independent variables. At a technical and mathematical level, both forms of modeling are quite similar, and use very similar methods in order to derive the optimal lower-dimensional representations. The main difference is in how the objective function of the two models is formulated. Some of the dependency-oriented models discussed in this chapter will also be useful in formulating outlier-detection models in the dependency-oriented data types discussed in later chapters.

It should be emphasized that regression-analysis is used extensively to detect anomalies in time-series data, and many of the basic techniques discussed in this chapter are applicable to that scenario as well. This techniques will be discussed in more detail in Chapter 9. However, since the time-series aspect of the problem is also based on dependencies of *temporally adjacent* data values, there are a number of subtle differences in the modeling process for multidimensional data and that for time-series data. Therefore, in this chapter, the much simpler case of multidimensional outlier analysis will be addressed. At the same time, the discussion will be general enough, so that the technical machinery required for applying regression analysis to the time-series scenario (Chapter 9) is introduced.

This chapter is organized as follows. In section 3.2, the basic linear regression models for outlier analysis will be introduced. In section 3.3, the principal component method for outlier analysis will be discussed. This can be considered an important special case of linear regression models, which is used frequently in outlier analysis, and it uses a similar optimization model. Therefore it is given a dedicated treatment in its own section. We discuss both the hard and soft versions of principal component analysis and show that the latter is equivalent to the Mahalanobis method discussed in the previous chapter. Furthermore, this technique can be easily extended to the nonlinear case with kernel methods. The related problem of one-class support-vector machines is discussed in section 3.4. A matrix-factorization view of linear models is discussed in section 3.5. Neural networks are introduced in section 3.6. Section 3.7 will study the limitations of linear models for outlier analysis. Section 3.8 contains the conclusions and summary.

## 3.2 Linear Regression Models

---

In linear regression, the observed values in the data are modeled using a linear system of equations. Specifically, the different dimensions in the data are related to one another using a set of linear equations, in which the coefficients need to be learned in a data-driven manner. Since the number of observed values is typically much larger than the dimensionality of the data, this system of equations is an *over-determined* one and cannot be solved exactly (i.e., zero error). Therefore, these models learn the coefficients that minimize the squared error of the deviations of data points from values predicted by the linear model. The exact choice of the error function determines whether a particular variable is treated specially (i.e., error of predicted variable value), or whether variables are treated homogeneously (i.e., error distance from estimated lower-dimensional plane). These different choices of the error function do *not* lead to the same model. In fact, the discussion in this chapter will show that the models can be qualitatively very different *especially in the presence of outliers*.

Regression analysis is generally considered an important application in its own right in the field of statistics. In classical instantiations of this application, it is desired to learn the value of a specific dependent variable from a set of independent variables. This is a common scenario in time-series analysis, which will be discussed in detail in Chapter 9.

Thus, a specific variable is treated *especially* from the other variables. Independent variables are also referred to as *explanatory variables*. This is a common theme with *contextual data types*, in which some attributes (e.g., time, spatial location, or adjacent series values) are treated as independent, whereas others (e.g., temperature or environmental measurements) are treated as dependent. Therefore, much of the framework in this section will also set the stage for the analysis of dependency-oriented data types in later chapters. However, for the straightforward multidimensional data type, all dimensions are treated in a homogeneous way, and the best-fitting linear relation among *all* the attributes is estimated. Therefore, there are subtle differences in the modeling (i.e., optimization formulation) of these two different settings.

Consider a domain such as temporal and spatial data, in which the attributes are partitioned into *contextual* and *behavioral* attributes. In such cases, a particular behavioral attribute value is often predicted as a linear function of the behavioral attributes in its *contextual* neighborhood in order to determine deviations from expected values. This is achieved by constructing a multidimensional data set from the temporal or spatial data in which a particular behavioral attribute value (e.g., temperature at current time) is treated as the dependent variable, and its contextual neighborhood behavioral values (e.g., temperatures in previous window) are treated as the independent variables. Therefore, the importance of predicting the dependent variable is paramount in estimating deviations and thereby quantifying outlier scores. In such cases, outliers are defined on the basis of the error of predicting the dependent variable, and anomalies within the interrelationships of independent variables are considered less important. Therefore, the focus of the optimization process is on minimizing the predicted error of the *dependent* variable(s) in order to create the model of normal data. Deviations from this model are flagged as outliers.

The special case of regression analysis with dependent variables will be studied first. The discussion of this case also sets the stage for a more detailed discussion for the cases of time-series data in Chapter 9 and spatial data in Chapter 11. Furthermore, the use of this technique for decomposition of the multidimensional outlier detection problem into a set of regression modeling problems will be studied in section 7.7 of Chapter 7. The identification of outliers in such cases is also very useful for *noise reduction* in regression modeling [467], which is an important problem in its own right.

In a later section, we will introduce a more general version of linear models in which no distinction is made between dependent and independent variables. The basic idea here is that the (normal) data are assumed to lie on a lower-dimensional hyperplane in the feature space. The normalized deviation of a point from this lower-dimensional hyperplane (in a direction perpendicular to the hyperplane) is used to compute the outlier score. As we will show later, these simple linear models can be greatly enriched with the use of *data transformations* that are able to map these linear models to more complex nonlinear models. Interestingly, certain types of transformations even allow the use of these linear models for outlier detection in more complex data types such as time-series data, discrete sequence data, and graph data. Furthermore, many other linear and nonlinear models such as one-class support-vector machines and neural networks can be viewed as variants of these models.

### 3.2.1 Modeling with Dependent Variables

A variable  $y$  can be modeled as a linear function of  $d$  dependent variables (or dimensions) as follows:

$$y = \sum_{i=1}^d w_i \cdot x_i + w_{d+1}$$

The variable  $y$  is the response variable or the dependent variable, and the variables  $x_1 \dots x_d$  are the independent or the explanatory variables. The coefficients  $w_1 \dots w_{d+1}$  need to be learned from the training data. The data set might contain  $N$  different instances denoted by  $\overline{X}_1 \dots \overline{X}_N$ . The  $d$  dimensions in the  $j$ th point  $\overline{X}_j$  are denoted by  $(x_{j1} \dots x_{jd})$ . The  $j$ th response variable is denoted by  $y_j$ . These  $N$  instance-pairs  $(\overline{X}_j, y_j)$  provide examples of how the dependent variable is related to the  $d$  independent variables with a linear function. The parameters of the linear function are learned with an optimization model that minimizes the aggregate squared error of predicting the dependent variable. The key assumption here is that the dependent variable is central to the application at hand, and therefore errors are defined only with respect to this variable. Outliers are defined as those data points in which the linear function learned from the  $N$  instances provides an unusually large error. Therefore, the  $j$ th instance of the response variable is related to the explanatory variables as follows:

$$y_j = \sum_{i=1}^d w_i \cdot x_{ji} + w_{d+1} + \epsilon_j \quad \forall j \in \{1, \dots, N\} \quad (3.1)$$

Here,  $\epsilon_j$  represents the error in modeling the  $j$ th instance, and it provides the *outlier score* of the pair  $(\overline{X}_j, y_j)$ . In *least-squares regression*, the goal is to determine the regression coefficients  $w_1 \dots w_{d+1}$  that minimize the error  $\sum_{j=1}^N \epsilon_j^2$ .

We will first introduce the notations needed to convert the system of  $N$  equations in Equation 3.1 into matrix form. The  $N \times (d+1)$ -matrix whose  $j$ th row is  $(x_{j1} \dots x_{jd}, 1)$  is denoted by  $D$ , and the  $N \times 1$  matrix (column vector) of the different values of  $y_1 \dots y_N$  is denoted by  $\overline{y}$ . The value of 1 is included as the final dimension in each row of  $D$  in order to address the effect of the constant term  $w_{d+1}$ . Thus, the first  $d$  dimensions of  $D$  can be considered a  $d$ -dimensional data set containing the  $N$  instances of the independent variables, and  $\overline{y}$  is a corresponding  $N$ -dimensional column-vector of response variables. The  $(d+1)$ -dimensional row-vector of coefficients  $w_1 \dots w_{d+1}$  is denoted by  $\overline{W}$ . This creates an *over-determined* system of equations corresponding to the rows of the following matrix-wise representation of Equation 3.1:

$$\overline{y} \approx D\overline{W}^T \quad (3.2)$$

Note the use of “ $\approx$ ” because we have not included the error term  $\epsilon_j$  of Equation 3.1. The least-squares error  $\sum_{j=1}^N \epsilon_j^2$  of predicting the response variable is optimized by minimizing the squared error  $\|D\overline{W}^T - \overline{y}\|^2$  in order to learn the coefficient vector  $\overline{W}$ . By applying differential calculus to the objective function, we obtain the following condition:

$$2D^T D\overline{W}^T - 2D^T \overline{y} = 0 \quad (3.3)$$

Therefore, the optimal coefficients for this minimization problem are provided by the following equation:

$$\overline{W}^T = (D^T D)^{-1} D^T \overline{y} \quad (3.4)$$

Note that  $D^T D$  is a  $(d+1) \times (d+1)$  matrix, which needs to be inverted in order to solve this system of equations. In cases where the matrix  $D$  is of rank less than  $(d+1)$ , the matrix



$D^T D$  will not be invertible. In such cases, Equation 3.2 represents an *under-determined* system, which has infinitely many solutions with zero error (outlier scores). In such cases, it is impossible to score the in-sample data points in a reasonable way because of paucity of data. Nevertheless, it is still possible to score out-of-sample data points by using the learned coefficients. To minimize overfitting, regularization is used. Given a regularization parameter  $\alpha > 0$ , we add the term  $\alpha \|W\|^2$  to the objective function. Therefore, the new objective function  $J$  is as follows:

$$J = \|D\bar{W}^T - \bar{y}\|^2 + \alpha \|W\|^2 \quad (3.5)$$

This optimization problem can be solved in a similar way by setting the gradient of  $J$  with respect to  $\bar{W}$  to 0.

$$2(D^T D + \alpha I)\bar{W}^T - 2D^T \bar{y} = 0 \quad (3.6)$$

Here,  $I$  represents a  $(d+1) \times (d+1)$  identity matrix. The regularized solution is as follows:

$$\bar{W}^T = (D^T D + \alpha I)^{-1} D^T \bar{y} \quad (3.7)$$

The closed-form solution to this problem is particularly convenient, and is one of the cornerstones of regression analysis in classical statistics. The outlier scores then correspond to the absolute values of the elements in the  $N$ -dimensional error vector  $\bar{\epsilon} = \bar{y} - D\bar{W}^T$ . However, it is important to note that using different attributes as the dependent variable will provide different outlier scores.

To understand this point, it is useful to examine the special case of 2-dimensional data:

$$Y = w_1 \cdot X + w_2 \quad (3.8)$$

In this case, the estimation of the coefficient  $w_1$  has a particularly simple form, and it can be shown that the best estimate for  $w_1$  is as follows:

$$w_1 = \frac{Cov(X, Y)}{Var(X)}$$

Here,  $Var(\cdot)$  and  $Cov(\cdot)$  correspond to the variance and covariance of the underlying random variables. The value  $w_2$  can further be easily estimated, by plugging in the means of  $X$  and  $Y$  into the linear dependence, once  $w_1$  has been estimated. In general, if  $X$  is regressed on  $Y$  instead of the other way around, one would have obtained  $w_1 = \frac{Cov(X, Y)}{Var(Y)}$ . Note that the regression dependencies would have been different for these cases.

The set of coefficients  $w_1 \dots w_{d+1}$  define a lower-dimensional hyperplane which fits the data as well as possible in order to optimize the error in the dependent variable. This hyperplane may be different for the same data set, depending upon which variable is chosen as the dependent variable. In order to explain this point, let us examine the behavior of two attributes from the *Auto-Mpg* data set of the *UCI Machine Learning Repository* [203].

Specifically, the second and the third attributes of the *Auto-Mpg* data set correspond to the *Displacement* and *Horsepower* attributes in a set of records corresponding to cars. The scatter plot for this pair of attributes is illustrated in Figure 3.2. Three regression planes have been shown in this figure, which are as follows:

- One regression plane is drawn for the case, when the *Horsepower* ( $Y$ -axis) is dependent on the *Displacement* ( $X$ -axis). The residual in this case is the error of prediction of the *Horsepower* attribute. The sum of squares of this residual over various data points is optimized.

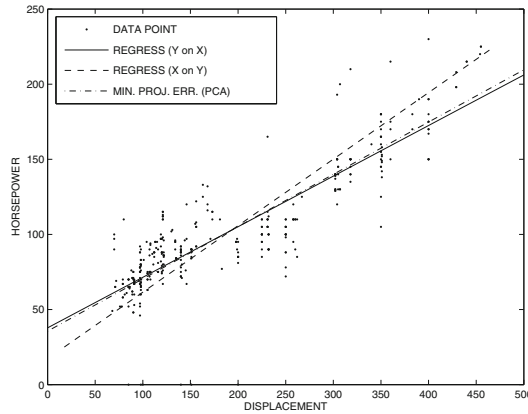


Figure 3.2: Optimal regression plane depends upon the choice of residual which is optimized

- The second regression plane is drawn for the case, when the *Displacement* ( $X$ -axis) is dependent on the *Horsepower* ( $Y$ -axis). The residual in this case is the error in prediction of the *Displacement* attribute.
- In the last case, the goal is to estimate a hyperplane that optimizes the aggregate mean-squared distance (i.e., residual error) of the data points from it. Thus, the residual in this case is the distance of each point to the hyperplane in a normal direction to the hyperplane, and it provides an outlier score. The computation of such a hyperplane will be discussed in a later section on principal component analysis (PCA).

It is evident from Figure 3.2 that the optimal hyperplanes in these different cases are quite different. While the optimization of the mean square projection distance produces a hyperplane which is somewhat similar to the case of  $Y$ -on- $X$  regression, the two are not the same. This is because these different cases correspond to different choices of errors on the residuals that are optimized, and therefore correspond to different best fitting hyperplanes. It is also noteworthy that the three projection planes are collinear and pass through the mean of the data set.

When the data fits the linear assumption well, all these hyperplanes are likely to be similar. However, the presence of noise and outliers can sometimes result in drastic negative effects on the modeling process. In order to illustrate this point, a variation of an example from [467] is used. In Figure 3.3, the different regression planes for two sets of five data points have been presented corresponding to different dependent variables. The two sets of five data points in Figures 3.3(a) and (b) are different by only one point, in which the  $Y$ -coordinate is distorted during data collection. As a result, this point does not fit the remaining data very well.

The original data set in Figure 3.3(a) fits the linear assumption very well. Therefore, all the three regression planes tend to be very similar to one another. However, after the perturbation of a single data point, the resulting projection planes are drastically perturbed. In particular, the  $X$  on  $Y$ -regression plane is significantly perturbed so as to no longer represent the real trends in the underlying data set. It is also noteworthy that the optimal projection plane is closer to the more stable of the two regression models. This is a general property of optimal projection planes, since they optimize their orientation in a stable way

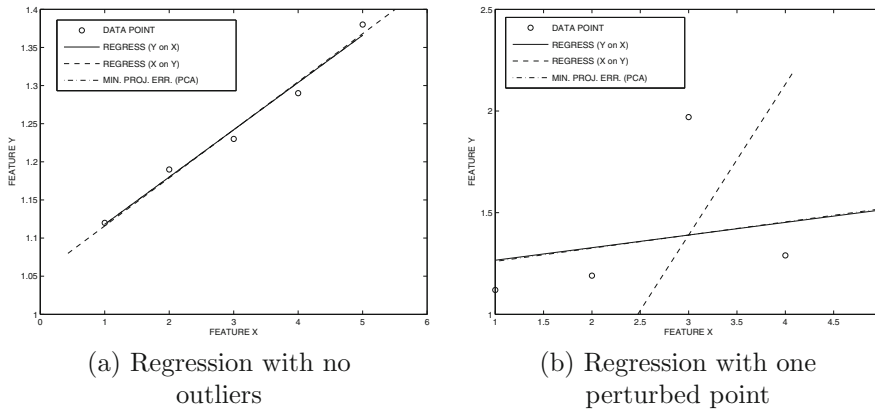


Figure 3.3: Drastic effects of outliers on quality of regression analysis

so as to globally fit the data well. The determination of such planes will be discussed in the next section.

The residual  $\epsilon_j$  provides useful information about the outlier score of the data point  $j$ . The mean of these residuals is expected to be 0, and the variance of these residuals can be estimated directly from the data. The Z-values of these residuals can also be used as outlier scores. The most common assumption in regression modeling is to assume that the error term  $\epsilon_i$  is a normal distribution, which is centered at zero. Then, the  $t$ -value test discussed in Chapter 2 can be used directly on the different residuals, and the outlying observations can be subsequently removed. The normal assumption on the residuals implies that the vector of coefficients is also normally distributed with mean and variances, as discussed earlier.

One problem with the approach is its instability to the presence of outliers. When the outliers have drastic effects on the regression, such as in the case of the  $X$ -on- $Y$  regression in Figure 3.3(b), the removal of outliers is likely to result in the removal of the wrong observations, since the regression parameters are drastically incorrect. Ironically, the presence of outliers prevents the proper modeling of outlier scores. One approach to address this problem is to use ensemble methods in which different subsets of points are sampled repeatedly to score the data points differently and average the scores. Note that all points can be scored with a given model because out-of-sample points can also be scored with the linear regression model. In cases where the number of training points is small, only out-of-sample points are scored for the averaging process. In other words, half the points are sampled to construct the model and the remaining half are scored. This process is repeated multiple times in order to score all the data points and provide the averaged predictions.

### 3.2.1.1 Applications of Dependent Variable Modeling

These scores can be used in a variety of applications associated with dependency-oriented data types. For example, in contextual data types, such as spatial and temporal data, these scores can be used to discover anomalous contextual outliers. These techniques will be discussed in detail in Chapters 9 and 11. These scores can also be used to remove those data points that are detrimental for learning and regression modeling applications. For example, in a regression modeling application, one can remove training data points with large training errors in order to improve the learning model.

It turns out that such models are useful even for *unsupervised* outlier detection of multi-

mensional data [429]. The basic idea is to repeatedly apply the regression modeling approach by fixing one of the attributes as the dependent variable and the remaining attributes as the independent variables. Thus, for a  $d$ -dimensional data set, a total of  $d$  regression models can be created. The squared errors of predicting each attribute are added in a weighted way in order to define the outlier score of a data point. This model is discussed in detail in section 7.7 of Chapter 7.

### 3.2.2 Linear Modeling with Mean-Squared Projection Error

The previous section discussed the case in which a particular variable (behavioral attribute) is considered special, and the optimal plane is determined in order to minimize the mean-squared error of the residuals for this variable. A more general form of regression modeling is one in which all variables are treated in a similar way, and the optimal regression plane is determined to minimize the *projection error* of the data to the plane. In the following discussion, it will be assumed that the data set containing  $N$  points is mean-centered, so that the mean of each dimension is 0.

The projection error of the data to the plane is the sum of the squares of the distances of the points to their projection into the plane. The projection of a point ‘A’ on the plane is the closest point on the plane to ‘A,’ and is computed using a line passing through ‘A’ in a normal direction to the plane. The point at which this line intersects the plane is the projection point. Thus, in this case, let us assume that we have a set of variables  $x_1 \dots x_d$ , and the corresponding regression plane is as follows:

$$w_1 \cdot x_1 + \dots + w_d \cdot x_d = 0 \quad (3.9)$$

It is assumed that the data is mean-centered, and therefore the constant term  $w_{d+1}$  is missing in this case. Each variable is associated with a coefficient, and the “special” (dependent) variable  $y$  is missing in this case. Consider a setting in which we have  $N$  instances corresponding to the  $d$ -dimensional data points, denoted by  $\overline{X}_1 \dots \overline{X}_N$ . If the number of points  $N$  is greater than the dimensionality  $d$ , the system of equations is over-determined and all the points in the data may not satisfy Equation 3.9, no matter how the vector of coefficients  $\overline{W} = (w_1 \dots w_d)$  is selected. Therefore, we need to allow for an error value  $\epsilon_j$ :

$$\overline{W} \cdot \overline{X}_j = \epsilon_j \quad \forall j \in \{1 \dots N\} \quad (3.10)$$

Therefore, the goal is to determine the row vector  $\overline{W} = (w_1 \dots w_d)$  of coefficients, so that the sum of the squared errors  $\sum_{i=1}^N \epsilon_j^2$  is minimized. In order to address issues of scaling and avoid the trivial solution  $\overline{W} = 0$ , a normalization constraint is assumed:

$$\|\overline{W}\|^2 = \sum_{i=1}^d w_i^2 = 1 \quad (3.11)$$

Note that this scaling constraint ensures that the absolute value of  $\overline{W} \cdot \overline{X}_j = \epsilon_j$  is exactly equal to the distance of data point  $\overline{X}_j$  from the hyperplane in Equation 3.9.

As before, let  $D$  be an  $N \times d$  matrix containing  $d$ -dimensional points in its rows, which are denoted by  $\overline{X}_1 \dots \overline{X}_N$ . One can succinctly write the  $N$ -dimensional column vector of distances of the different data points to this regression plane as  $\overline{\epsilon} = D\overline{W}^T$  according to Equation 3.10. This column vector contains the outlier scores. The  $L_2$ -norm  $\|D\overline{W}^T\|^2$  of the column vector of distances is the objective function, which needs to be minimized to

determine the optimal coefficients  $w_1 \dots w_d$ . This is a constrained optimization problem because of the normalization constraint in Equation 3.11. One can set the gradient of the Lagrangian relaxation  $\|D\overline{W}^T\|^2 - \lambda(\|\overline{W}\|^2 - 1)$  to 0 in order to derive a constraint on the optimal parameter vector  $\overline{W}$ . This constraint turns out to be in eigenvector form:

$$[D^T D]\overline{W}^T = \lambda\overline{W}^T \quad (3.12)$$

Which eigenvector of the positive semi-definite matrix  $D^T D$  provides the lowest objective function value? Substituting for  $[D^T D]\overline{W}^T$  from Equation 3.12 in the original objective function evaluates it to  $\lambda\|\overline{W}\|^2 = \lambda$ . *The aggregate squared error along a particular eigenvector is equal to the eigenvalue.* Therefore, the optimal vector  $\overline{W}$  is the smallest eigenvector of  $D^T D$ . Note that the matrix  $D^T D$  is a scaled version of the covariance matrix  $\Sigma$  of the mean-centered data matrix  $D$ :

$$\Sigma = \frac{D^T D}{N} \quad (3.13)$$

The scaling of a matrix does not affect its eigenvectors but it scales the eigenvalues to the *variances* along those directions instead of the *aggregate errors*. The data is therefore assumed to be compactly represented by the  $(d - 1)$ -dimensional hyperplane, which is perpendicular to the smallest eigenvector of the covariance matrix  $\Sigma$ . The outlier scores correspond to the distances of the data points from their nearest point on (i.e., perpendicular distance to) this hyperplane. However, a dimensionality of  $(d - 1)$  is often too large to discover sufficiently discriminative outlier scores. The aforementioned solution, nevertheless, provides a first step to an effective (and more general) solution to the problem, which is referred to as *principal component analysis (PCA)*. The PCA method generalizes the aforementioned solution to a  $k$ -dimensional hyperplane, in which the value of  $k$  can vary at any value between  $\{1 \dots d - 1\}$ . Because of its importance to outlier analysis, this method will be discussed in a dedicated section of its own along with corresponding applications.

### 3.3 Principal Component Analysis

---

The least-squares formulation of the previous section simply tries to find a  $(d - 1)$ -dimensional hyperplane which has an optimum fit to the data values and the score is computed along the remaining orthogonal direction. Principal component analysis can be used to solve a general version of this problem. Specifically, it can find optimal representation hyperplanes of *any* dimensionality. In other words, the PCA method can determine the  $k$ -dimensional hyperplane (for any value of  $k < d$ ) that minimizes the squared projection error over the remaining  $(d - k)$  dimensions. The optimization solution in the previous section is a special case of principal component analysis, which is obtained by setting  $k = d - 1$ .

In principal component analysis, the  $d \times d$  covariance matrix over  $d$ -dimensional data is computed, where the  $(i, j)$ th entry is equal to the covariance between the dimensions  $i$  and  $j$  for the set of  $N$  observations. As discussed in the previous section, consider a multidimensional data set  $D$  of dimensionality  $d$  and size  $N$ . The  $N$  different rows of  $D$  are denoted by  $\overline{X}_1 \dots \overline{X}_N$ . Each row is a  $d$ -dimensional instance in the data. The individual dimensions of the  $i$ th row are denoted by  $\overline{X}_i = [x_{i1} \dots x_{id}]$ , where  $x_{ij}$  is the  $j$ th dimension of the  $i$ th instance  $\overline{X}_i$ . Let us denote the  $d \times d$  covariance matrix of the data set by  $\Sigma$ , in which the  $(i, j)$ th entry is the covariance between the  $i$ th and  $j$ th dimensions. Since the data set  $D$  is mean-centered, the covariance matrix  $\Sigma$  may be expressed as follows:

$$\Sigma = \frac{D^T D}{N} \quad (3.14)$$

This matrix can be shown to be symmetric and positive semi-definite. It can therefore be diagonalized as follows:

$$\Sigma = P\Delta P^T$$

Here,  $\Delta$  is a diagonal matrix, and  $P$  is an a matrix, whose columns correspond to the (orthonormal) eigenvectors of  $\Sigma$ . The corresponding entries in the diagonal matrix  $\Delta$  provide the eigenvalues. In the aforementioned section, we stated that the *normal hyperplane* to the smallest eigenvector of  $\Sigma$  provides the  $(d-1)$ -dimensional hyperplane that approximates the data with the least-squared error. Another way of stating this is that the subspace *that is a linear combination of the  $(d-1)$  largest eigenvectors* provides an axis system in which the data can be approximately represented with very little loss. It is possible to generalize this argument with the  $k$  largest eigenvectors to define a corresponding  $k$ -dimensional subspace of (approximate) representation. Here,  $k$  can be any value in  $\{1 \dots d-1\}$  and it might not be set to only  $(d-1)$ . Outliers are data points for which the error of this approximation is high.

Therefore, in principal component analysis, the first step is to transform the data into a new axis system of representation. The orthonormal eigenvectors provides the axes directions along which the data should be projected. The key properties of principal component analysis, which are relevant to outlier analysis, are as follows:

**Property 3.3.1 (PCA Properties)** *Principal component analysis provides a set of eigenvectors satisfying the following properties:*

- *If the data is transformed to the axis-system corresponding to the orthogonal eigenvectors, the variance of the transformed data along each axis (eigenvector) is equal to the corresponding eigenvalue. The covariances of the transformed data in this new representation are 0.*
- *Since the variances of the transformed data along the eigenvectors with small eigenvalues are low, significant deviations of the transformed data from the mean values along these directions may represent outliers.*

More details and properties of PCA may be found in [33, 296]. PCA provides a more general solution than the 1-dimensional optimal solution of the previous section, because the PCA-solution provides a recursive solution of *any* dimensionality depending on the choice of parameter  $k$ . It is also noteworthy that the PCA approach potentially uses *all* the solutions of Equation 3.12 rather than using only the smallest eigenvector.

The data can be transformed to the new axis system of orthonormal eigenvectors, with transformed  $d$ -dimensional records denoted by  $\overline{X}'_1 \dots \overline{X}'_N$ . This can be achieved by using the product between the original row-vector representation  $\overline{X}_i$  and the matrix  $P$  containing the new axes (orthonormal eigenvectors) in its columns:

$$\overline{X}'_i = [x'_{i1} \dots x'_{id}] = \overline{X}_i P \quad (3.15)$$

Let  $D'$  be the transformed data matrix for which the  $i$ th row contains the transformed point  $\overline{X}'_i$ . The transformed data matrix  $D'$  in the de-correlated axis-system can be expressed in terms of the original data matrix  $D$  as follows:

$$D' = DP \quad (3.16)$$

In this new representation, the inter-attribute covariances in the new data matrix  $D'$  are zero, and the variances along the individual attributes correspond to the coordinates along

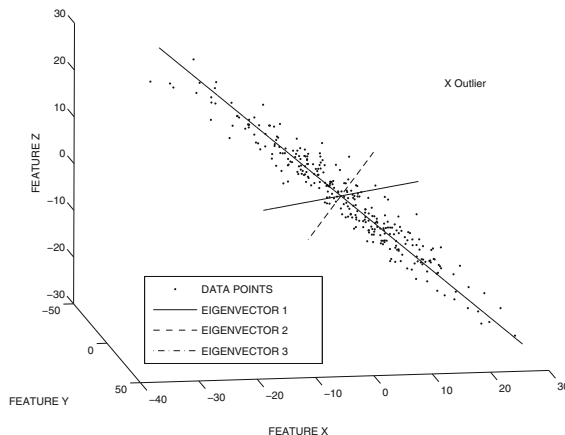


Figure 3.4: Eigenvectors correspond to directions of correlations in the data. A small number of eigenvectors can capture most of the variance in the data.

the eigenvectors eigenvalues. For example, if the  $j$ th eigenvalue is very small, then the value of  $x'_{ij}$  in this new transformed representation does not vary much over the different values of  $i$  and fixed  $j$ . For mean-centered data, these values of  $x'_{ij}$  are approximately equal to the mean value of 0, unless the data point is an outlier.

The beautiful part about PCA is that the largest eigenvectors provide the key directions of *global* correlation in a single shot. These directions are also referred to as the *principal components*, which are uncorrelated and retain most of the data variance. In real settings, it is common for a large fraction of the eigenvalues to be very small. This means that most of the data aligns along a *much lower-dimensional subspace*. This is very convenient from the perspective of outlier analysis, because the observations that do not respect this alignment can be assumed to be outliers. For example, for an eigenvector  $j$  that has a small eigenvalue, a large deviation of  $x'_{ij}$  for the  $i$ th record from other values of  $x'_{kj}$  is indicative of outlier behavior. This is because the values of  $x'_{kj}$  do not vary much, when  $j$  is fixed and  $k$  is varied. Therefore, the value  $x'_{ij}$  may be deemed to be unusual in these settings.

The effectiveness of principal component analysis in exposing outliers from the underlying data set can be illustrated with an example. Consider the scatterplot of the 3-dimensional data illustrated in Figure 3.4. In this case, the corresponding eigenvectors have been ordered by decreasing eigenvalues (variances), although this is not immediately obvious from the figure in this 2-dimensional perspective. In this case, the standard deviation along the first eigenvector is three times that along the second eigenvector and nine times that along the third eigenvector. Thus, most of the variance would be captured in the lower-dimensional subspace formed by the top two eigenvectors, although a significant amount of variance would also be captured by selecting only the first eigenvector. If the distances of the original data points to the 1-dimensional line corresponding to the first eigenvector (and passing through the mean of the data) are computed, the data point 'X' in the figure would be immediately exposed as an outlier. In the case of high-dimensional data, most of the variance of the data can be captured along a much lower  $k$ -dimensional subspace. The residuals for the data points can then be computed by examining the projection distances to this  $k$ -dimensional hyperplane passing through the mean of the data points. Data points that have very large distances from this hyperplane can be identified as outliers. Although it is

possible to use this distance as an outlier score, it is possible to sharpen the score further with normalization as follows. The squared Euclidean distance to this hyperplane can be decomposed into the sum of the squares of the  $(d - k)$  distances along the smallest eigenvectors. Each of these  $(d - k)$  squared distances should be divided by the corresponding eigenvalue (for variance standardization) and the scaled values should be added to provide the final result. The intuition behind this scaling is that the variance along an eigenvector is its eigenvalue, and therefore a large deviation along a smaller eigenvalue should be rewarded to a greater degree.

### 3.3.1 Connections with the Mahalanobis Method

In the aforementioned method, we remove the  $k$  largest eigenvectors in a *hard* way and compute a weighted sum of these squared distances along the remaining  $(d - k)$  distances as the outlier score. A simpler special case would be to use a *soft* approach for weighting the distances along *all* the different eigenvectors rather than *selecting* a particular set of eigenvectors in a *hard* way. This special case turns out to be the same as the Mahalanobis method in section 2.3.4 of Chapter 2.

This computation is achieved by evaluating the normalized distance of the data point to the centroid along the direction of *each principal component*. Let  $\bar{e}_j$  be the  $j$ th eigenvector with a variance (eigenvalue) of  $\lambda_j$  along that direction. The overall normalized outlier score of a data point  $\bar{X}$ , to the centroid<sup>1</sup>  $\bar{\mu}$  of the data is given by the sum of squares of these values:

$$\text{Score}(\bar{X}) = \sum_{j=1}^d \frac{|(\bar{X} - \bar{\mu}) \cdot \bar{e}_j|^2}{\lambda_j} \quad (3.17)$$

Note the presence of  $\lambda_j$  in the denominator, which provides a soft weighting. A simple algorithm to compute these scores for the rows of an  $n \times d$  data matrix  $D$  is as follows:

1. Compute the covariance matrix  $\Sigma$  of the original data matrix  $D$  and diagonalize it as  $\Sigma = P\Delta P^T$ .
2. Transform the data  $D$  to a new de-correlated axis-system as  $D' = DP$ .
3. Standardize each column of  $D'$  to unit variance by dividing it with its standard deviation.
4. For each row of  $D'$ , report its (squared) Euclidean distance from the centroid of  $D'$  as its outlier score.

It is important to note that most of the contribution to the outlier score in Equation 3.17 is provided by deviations along the principal components with small values of  $\lambda_j$ , when a data point deviates significantly along such directions. This is also captured by the important standardization step in the aforementioned description. This step recognizes that the principal components represent the independent concepts in the data and *implements a form of soft weighting of the transformed dimensions by standardization* rather than selecting a subset of transformed dimensions in a hard way. The sum of the squares of the Euclidean distances of the point from the centroid along the transformed dimensions is a  $\chi^2$ -distribution with  $d$  degrees of freedom. The value of the aggregate residual is compared

<sup>1</sup>Although the centroid is the origin for mean-centered data, it is also possible to transform non-centered data once the eigenvectors of the covariance matrix have been computed. The expression of Equation 3.17 continues to hold for settings in which  $\bar{\mu}$  is not necessarily 0.



to the cumulative distribution for the  $\chi^2$ -distribution in order to determine a probability value for the level of anomalousness. The aforementioned approach was first used in [493].

Although it might not be immediately apparent, the score computed above is the same as the Mahalanobis method discussed in section 2.3.4 of Chapter 2 (cf. Exercise 11). Specifically, the Mahalanobis distance value between  $\bar{X}$  and  $\bar{\mu}$  computed in Equation 2.21 of that section is *exactly the same* as the score in Equation 3.17 above, except that the eigenvector analysis above provides a better understanding of how this score is decomposed along the different directions of correlation. Another advantage of this interpretation of the method is that it can be naturally extended to settings in which the data is distributed on a *non-linear manifold* rather than the linear setting of Figure 3.4. This extension will be discussed in section 3.3.8.

### 3.3.2 Hard PCA versus Soft PCA

The Mahalanobis method can be viewed as a type of soft PCA in which the principal components are weighted rather than pre-selected. The PCA decomposition also allows the option of ignoring the large eigenvectors and using only the smallest  $\delta < d$  eigenvectors in order to compute the outlier scores. However, the benefits of doing so are unclear because the Mahalanobis approach already performs a kind of soft pruning with the inverse weighting of the contributions of each eigenvector by the eigenvalues. It is not uncommon for a rare value to align along a long eigenvector when all attributes are extreme values in a correlated data set. By explicitly pruning the long eigenvectors, such outliers can be missed. Therefore, hard PCA focuses only on finding dependency-oriented outliers, whereas the Mahalanobis method (soft PCA) can discover both dependency-oriented outliers and extreme values. In this sense, the Mahalanobis method can be viewed as a more elegant generalization of hard PCA. Hard PCA focuses on the *reconstruction* error of representing the data in a low-dimensional space, which introduces an additional parameter of selecting the dimensionality of the representation space. Introducing parameters always results in data-specific unpredictability of results in unsupervised settings, where there is no guided way of tuning the parameters.

### 3.3.3 Sensitivity to Noise

Principal component analysis is generally more stable to the presence of a few outliers than the dependent variable analysis methods. This is because principal component analysis computes the errors with respect to the *optimal hyperplane*, rather than a *particular variable*. When more outliers are added to the data, the optimal hyperplane usually does not change too drastically. Nevertheless, in some settings, the presence of outliers can cause challenges. In such cases, several techniques exist for performing robust PCA. For example, this approach can be used in order to determine the obvious outliers in the first phase. In the second phase, these outliers can be removed, and the covariance matrix can be constructed more robustly with the remaining data. The scores are then recomputed with the adjusted covariance matrix. This approach can also be applied iteratively. In each iteration, the obvious outliers are removed, and a more refined PCA model is constructed. The final outlier scores are the deviation levels in the last iteration.

### 3.3.4 Normalization Issues

PCA can sometimes yield poor results when the scales of the different dimensions are very different. For example, consider a demographic data set containing attributes such as *Age* and *Salary*. The *Salary* attribute may range in the tens of thousands, whereas the *Age* attribute is almost always less than a hundred. The use of PCA would result in the principal components being dominated by the high-variance attributes. For example, for a 2-dimensional data set containing only *Age* and *Salary*, the largest eigenvector will be almost parallel to the *Salary* axis, irrespective of very high correlations between the *Age* and *Salary* attributes. This can reduce the effectiveness of the outlier detection process. Therefore, a natural solution is to normalize the data, so that the variance along each dimension is one unit. This is achieved by dividing each dimension with its standard deviation. This implicitly results in the use of a *correlation matrix* rather than the *covariance matrix* during principal component analysis. Of course, this issue is not unique to linear modeling, and it is often advisable to use such pre-processing for most outlier detection algorithms.

### 3.3.5 Regularization Issues

When the number of data records  $N$  is small, the covariance matrix cannot be estimated very accurately, or it might be ill-conditioned. For example, a data set containing a small number of records might have zero variance along some of the dimensions, which might underestimate the true variability. In such cases, it is desirable to use regularization to avoid overfitting. This type of regularization intuitively similar to the notion of Laplacian smoothing discussed for the EM method in Chapter 2. The basic idea is to adjust the covariance matrix  $\Sigma$  by adding  $\alpha I$  to it, where  $I$  is a  $d \times d$  identity matrix and  $\alpha > 0$  is a small regularization parameter. The eigenvectors of  $(\Sigma + \alpha I)$  are then used for computing the scores. The basic effect of this modification is intuitively equivalent to adding a small amount of independent noise with variance  $\alpha$  to each dimension before computing the covariance matrix.

Alternatively, one can use cross-validation for scoring. The data is divided into  $m$  folds, and the PCA model is constructed only on  $(m - 1)$  folds. The remaining fold is scored. This process is repeated for each fold. An ensemble-centric variant is to sample a subset of the data to build the model, and score all the points. The process is repeated over multiple samples and the scores are averaged. The effectiveness of this approach is shown in [35].

### 3.3.6 Applications to Noise Correction

Most of this book is devoted to either removal of outliers as noise or identification of outliers as anomalies. However, in some applications, it may be useful to *correct* the errors in outliers so that they conform more closely to the broad patterns in the data. PCA is a natural approach for achieving this goal because the principal components represent the broad patterns. The basic idea is that the *projection of the data point on the  $k$ -dimensional hyperplane corresponding to the largest eigenvalues (and passing through the data mean) provides a corrected representation*. Obviously such an approach is likely to correct the outlier points significantly more than most of the other normal data points. Some theoretical and experimental results on why such an approach is likely to improve data quality is provided in [21].

A similar approach to PCA, referred to as *Latent Semantic Indexing (LSI)* has also been used for text data to improve retrieval quality [162, 425]. Text representations are inherently

noisy because the same word may mean multiple things (*synonymy*) or the same concept can be represented with multiple words (*polysemy*). This leads to numerous challenges in virtually all similarity-based applications. In particular, it has been observed in [425] that the use of such dimensionality reduction methods in text data significantly improves the effectiveness of similarity computations, because of the reduction in the noise effects of *synonymy* and *polysemy*. It has been observed [425] that such dimensionality reduction methods significantly improve similarity computations in text because of the reduction in the noise effects of *synonymy* and *polysemy*. The technique of LSI [162] is a variant of PCA, which was originally developed for efficient indexing and retrieval. However, it was eventually observed that the quality of similarity computations improves as well [425]. This observation was taken to its logical conclusion in [21], where significant noise reduction was shown both theoretically and experimentally with PCA-based techniques.

An even more effective approach to noise correction is to combine outlier removal and re-insertion with the correction process. The first step is to perform PCA, and remove the top outliers on the basis of a  $t$ -test with respect to the optimal plane of representation. Subsequently, PCA is performed again on this cleaner data set in order to generate the projection subspaces more accurately. The projections can then be performed on this corrected subspace. This process can actually be repeated iteratively, if desired in order to provide further refinement. A number of other approaches to perform regression analysis and outlier removal in a robust way are presented in [467].

### 3.3.7 How Many Eigenvectors?

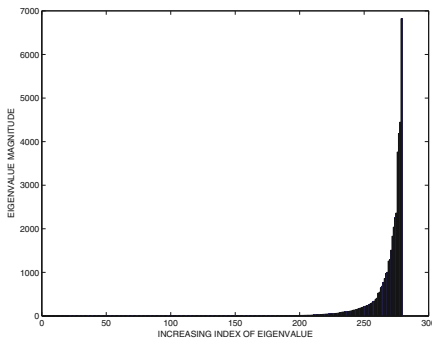
As discussed earlier, the eigenvectors with the largest variance provide the most informative subspaces for data representation, outlier analysis, and noise correction. In some cases, it is not necessary to select a subset of the dimensions in a hard way because a soft inverse weighting with eigenvalues is sufficiently effective. However, in many applications such as noise correction, the data needs to be projected into a subspace of lower dimensionality by selecting a specific number of eigenvectors. Therefore, a natural question arises, as to how the dimensionality  $k$  of the projection subspace should be determined.

One observation in most real data sets is that the vast number of eigenvalues are relatively small, and most of the variance is concentrated in a few eigenvectors. An example illustrated in Figure 3.5 shows the behavior of the 279 eigenvectors of the *Arrhythmia* data set of the *UCI Machine Learning Repository* [203]. Figure 3.5(a) shows the absolute magnitude of the eigenvalues in increasing order, whereas Figure 3.5(b) shows the total amount of variance retained in the top- $k$  eigenvectors. In essence, Figure 3.5(b) is derived by using the cumulative sum over the eigenvalues in Figure 3.5(a). While it was argued at the beginning of the chapter that the *Arrhythmia* data set is weakly correlated along many of the dimensions, on a pairwise basis, it is interesting to note that it is still possible<sup>2</sup> to find a small number of directions of global correlation along which most of the variance is retained. In fact, it can be shown that the smallest 215 eigenvalues (out of 279) *cumulatively* contain less than 1% of the variance in the data set.

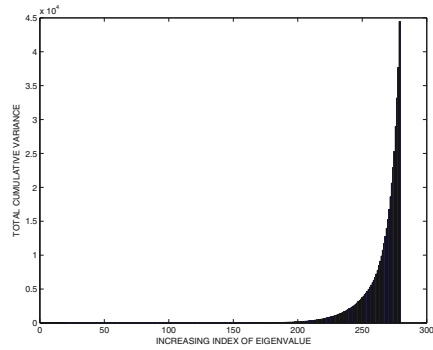
In other words, most eigenvalues are very small. Therefore, it pays to retain the eigenvectors corresponding to extremely large values, with respect to the average behavior of the

---

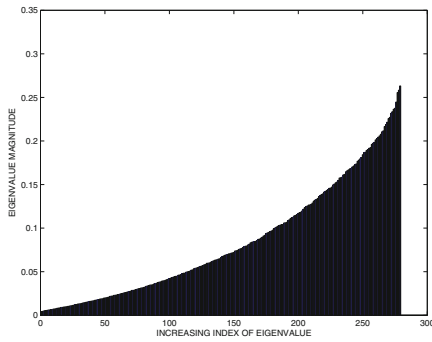
<sup>2</sup>This is partially because the data set is relatively small with only 452 records. For example, the results in Figures 3.5(c) and (d) show that even for a uniformly distributed data set of the same size, it is possible to find some skews in the eigenvalues because of (undesirable) overfitting. On the other hand, a strength of PCA is that the cumulative effects of even weak correlations become magnified with increasing dimensionality, and it becomes possible to find a much lower-dimensional subspace contain the informative projections.



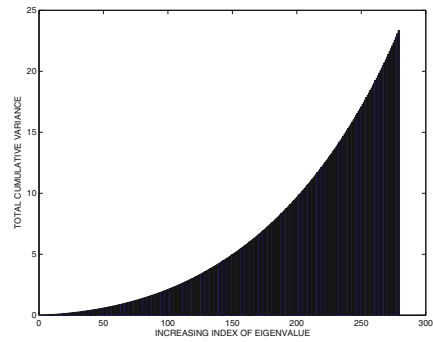
(a) Magnitude of eigenvalues  
(Increasing index): *Arrythmia*



(b) Variance in smallest  $k$   
eigenvalues: *Arrythmia*



(c) Magnitude of eigenvalues  
*Uniform: 452 records only*



(d) Variance in smallest  $k$  eigenvalues  
*Uniform: 452 records only*

Figure 3.5: A few eigenvalues contain most of the variance (*Arrythmia*)

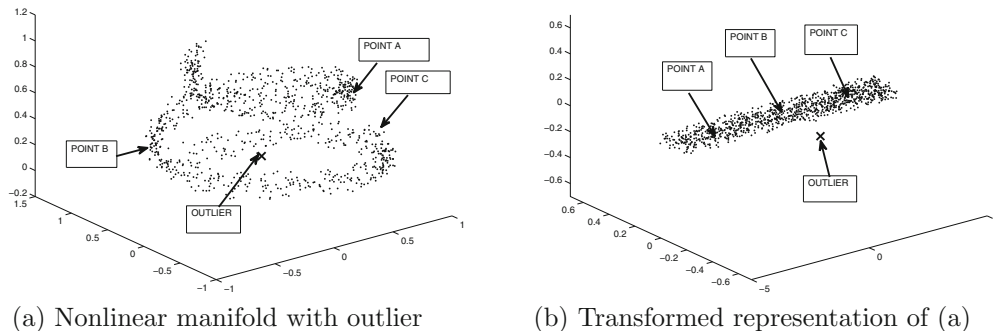


Figure 3.6: Nonlinear transformations can expose outliers that are hard to discover with linear models

eigenvalues. How to determine, what is “extremely large?” This is a classical case of extreme value analysis methods, which were introduced in Chapter 2. Therefore, each eigenvalue is treated as a data sample, and the statistical modeling is used to determine the large values with the use of hypothesis testing. A challenge in this case is that the sample sizes are often small unless the data set is very high dimensional. Even for relatively high-dimensional data sets (e.g., 50-dimensional data sets), the number of samples (50 different eigenvalues) available for hypothesis testing is relatively small. Therefore, this is a good candidate for the  $t$ -value test. The  $t$ -value test can be used in conjunction with a particular level of significance and appropriate degrees of freedom to determine the number of selected eigenvectors.

### 3.3.8 Extension to Nonlinear Data Distributions

Although the PCA methodology works well for linearly aligned data distributions such that those discussed in Figure 3.4, the vanilla version of PCA is not effective in cases where the data is aligned along nonlinear manifolds. An example of such a manifold is illustrated in Figure 3.6(a). In such cases, PCA can be extended to nonlinear settings with the use of a technique known as the *kernel trick*. The basic idea in nonlinear PCA is to use a data transformation so that the data aligns along a linear hyperplane in the transformed space. In other words, the mapping unfurls the non-linear manifold into a linear embedding. An illustrative example of such an embedding is illustrated in Figure 3.6(b). After such an embedding has been constructed, we can use the score computation method of Equation 3.17.

How can such embeddings be constructed? A naive approach would be to use *explicit* polynomial transformations of the data. For example, a second-order polynomial transformation would represent each point with  $d + d^2$  dimensions. In other words, the  $d$  original dimensions are included and  $d^2$  new dimensions are added corresponding to pairwise products of data values. Subsequently, *all nonzero* eigenvectors can be extracted from the covariance matrix of this new representation. Note that the number of nonzero eigenvectors can be larger than  $d$  in this setting because we do not know the dimensionality of the transformed space. However, the number of nonzero eigenvectors is always no larger than the number of data points. Equation 3.17 can be applied to this new representation in order to determine the scores. Unfortunately, this is not a practical approach because it expands the number of

dimensions to  $O(d^2)$ , and the size of the covariance matrix is  $O(d^4)$ . Such an approach would be impractical from a computational point of view. Fortunately, these transformations need not be done *explicitly*; rather, they can be done implicitly with the use of carefully designed  $N \times N$  similarity matrices between all pairs of data points. This technique is also referred to as the *kernel trick*, although a generalized point of view encompasses various methods such as *spectral embeddings* [297] and ISOMAP [542]. The approach described in this section is a direct nonlinear generalization of the Mahalanobis method with kernel functions, and is introduced in [35] as an ensemble-centric adaptation. Some of these ideas are also related to the concept of *kernel whitening* [541], which advocates scaling of all kernel features to unit variance before applying any learning method.

In order to simplify further exposition, we will first describe an alternative method for extracting (linear) PCA from a dot-product *similarity matrix* rather than a covariance matrix. This alternative description of linear PCA is easier to generalize to the nonlinear case. In our earlier description of PCA, we first construct the covariance matrix  $\Sigma = \frac{D^T D}{N}$  from the mean-centered matrix  $D$  and diagonalize it to extract its  $d \times d$  eigenvector matrix  $P$ , the columns of which are the new basis vectors. Subsequently, the transformed representation of data is given by projecting the data on these basis vectors as  $D' = DP$ . After the new representation has been determined, the outlier score is simply equal to the distance from the centroid, after standardizing the transformed data to unit variance along each (transformed) dimension. Therefore, the key is to extract the new *embedding*  $D'$  of the data representation in order to extract the outlier scores. The basis representation in the columns of  $P$  only provides an intermediate step for computing the embedding  $D'$ , and is itself not essential for computing the outlier scores. The similarity-matrix approach to PCA extracts this embedding  $D'$  directly without computing these basis vectors.

In the similarity-matrix approach to PCA, the  $N \times N$  dot product matrix  $S = DD^T$  is constructed instead of the  $d \times d$  covariance matrix  $D^T D$ . Note that  $S$  is a dot-product similarity matrix in which the  $(i, j)$ th entry is equal to the dot-product between the  $i$ th and  $j$ th data points. Subsequently, the positive-semidefinite similarity matrix  $S$  is diagonalized as follows:

$$S = DD^T = Q\Lambda^2Q^T \quad (3.18)$$

Here,  $Q$  is an  $N \times N$  matrix whose columns contain orthonormal eigenvectors, although only the top- $d$  eigenvectors have non-zero eigenvalues because the matrix  $DD^T$  has rank at most  $d$ .  $\Lambda^2$  is a diagonal matrix containing the non-negative eigenvalues of the positive semi-definite matrix. Then, in the alternative similarity-matrix approach to PCA, it can be shown that the top- $d$  scaled eigenvectors  $Q\Lambda$  (i.e., first  $d$  columns of  $Q\Lambda$ ) yield the new  $N \times d$  data representation  $D'$  [515]. Therefore, this alternative approach<sup>3</sup> to PCA *directly* extracts the embedding from the eigenvectors and eigenvalues of the  $N \times N$  dot-product similarity matrix  $S = DD^T$  without computing a basis matrix  $P$ . This is particularly useful for nonlinear dimensionality reduction because it is often hard to interpret the transformed basis in terms of the original features.

The key idea in nonlinear PCA is that by replacing the dot-product similarity matrix  $S = DD^T$  with a carefully chosen *kernel similarity matrix* in the aforementioned approach, nonlinear manifolds are mapped on linear hyperplanes such as the mapping from Figure 3.6(a) (data matrix  $D$ ) to Figure 3.6(b) (transformed embedding  $Q\Lambda$ ). Such similarity matrices will be discussed in section 3.3.8.1. This approach is the essence of a well-known technique known as the *kernel trick* and the corresponding approach is referred to as *kernel*

<sup>3</sup>This equivalence can be shown using the SVD decomposition of matrix  $D$ . In fact, the nonzero eigenvalues of  $DD^T$  and  $D^T D$  are the same.

PCA. The key is to assume that the  $(i, j)$ th value of  $S$  is equal to  $\Phi(\overline{X}_i) \cdot \Phi(\overline{X}_j)$  in some transformed space according to some unknown transformation  $\Phi(\overline{X}_i)$  of higher dimensionality. Therefore, the similarity matrix  $S$  can be used to compute the Mahalanobis scores as follows:

1. Construct the positive semi-definite  $N \times N$  kernel similarity function  $S$  using an off-the-shelf kernel function or other similarity matrix computation (e.g., sparsified and normalized similarity matrix used in spectral methods [378]).
2. Diagonalize the matrix  $S = Q\Lambda^2Q^T$  and set the new  $N \times k$  embedding  $D'$  to the first  $k$  columns of  $Q\Lambda$  corresponding to largest eigenvalues. The default assumption is to set  $k$  so that all nonzero eigenvectors of  $S$  are included. Note that  $k$  can be larger than  $d$  in the nonlinear setting.
3. Standardize each column of  $D'$  to unit variance by dividing it with its standard deviation.
4. For each row of  $D'$ , report its (squared) Euclidean distance from the centroid of  $D'$  as its outlier score.

Why did we use all nonzero eigenvectors? In problems like outlier detection, aggregate trends are not sufficient because one is looking for exceptions. In fact, most outliers are emphasized along the *smaller* eigenvectors. Even a single data point deviating significantly along a small eigenvector is important for outlier detection. It is important to select all nonzero eigenvectors, so that important outliers are not missed. Assume that the columns of  $Q$  and  $\Lambda$  are sorted in order of decreasing eigenvalue. One should select the (unique) value of  $k$  such that  $\Lambda_{kk} > 0$ , and  $\Lambda_{k+1,k+1} = 0$ . In practice, the value of the latter will not be exactly zero because of numerical errors during eigenvalue computation. Furthermore, such computational errors magnify the inaccuracy in scoring along small eigenvectors. Therefore, some conservative threshold  $\epsilon$  on the eigenvalue (e.g.,  $10^{-8}$ ) can be used to select the cut-off point. Another caveat is that the approach can give poor results because of overfitting when all  $N$  eigenvectors are nonzero; in such cases, either dropping the *largest* eigenvectors or using an out-of-sample implementation (see section 3.3.8.2) is helpful.

### 3.3.8.1 Choice of Similarity Matrix

How does one choose the similarity matrix  $S$ ? The  $(i, j)$ th entry of  $S$  is defined by the kernel similarity  $K(\overline{X}_i, \overline{X}_j)$  between the data points  $\overline{X}_i$  and  $\overline{X}_j$ . There are a number of off-the-shelf kernel functions that are commonly used in such settings, although the general consensus is in favor of the Gaussian kernel [270, 541]. Some common choices of the kernel function  $S = [K(\overline{X}_i, \overline{X}_j)]$  are as follows:

Function	Form
Linear Kernel	$K(\overline{X}_i, \overline{X}_j) = \overline{X}_i \cdot \overline{X}_j$ (Defaults to PCA)
Gaussian Radial Basis Kernel	$K(\overline{X}_i, \overline{X}_j) = e^{-\ \overline{X}_i - \overline{X}_j\ ^2 / \sigma^2}$
Polynomial Kernel	$K(\overline{X}_i, \overline{X}_j) = (\overline{X}_i \cdot \overline{X}_j + c)^h$
Sigmoid Kernel	$K(\overline{X}_i, \overline{X}_j) = \tanh(\kappa \overline{X}_i \cdot \overline{X}_j - \delta)$

Note that the special case of linear PCA without any transformation (section 3.3) corresponds to  $K(\bar{X}_i, \bar{X}_j) = \bar{X}_i \cdot \bar{X}_j$ , and the corresponding kernel is also referred to as the linear kernel. Since PCA is technically defined for mean-centered kernel matrices, it is possible to mean-center<sup>4</sup> the kernel matrix. However, it is not essential in either the linear or nonlinear setting, and can sometimes cause unexpected problems in the nonlinear setting when all eigenvectors are nonzero.

Many of these kernel functions have parameters associated with them. The shape of the embedding will depend on the choice of the kernel function and its parameters. Note that the use of an arbitrary kernel function might result in an embedding somewhat different from Figure 3.6(b), which is only illustrative in nature. Furthermore, different choices of kernel functions or parameters might have different levels of effectiveness in exposing the outliers. Unfortunately, in unsupervised methods there is no meaningful way of choosing such kernel functions or parameters. The use of the Gaussian<sup>5</sup> kernel is particularly common because of its stability across different data sets. In that case, the parameter  $\sigma$  should be of the same order of magnitude as the pairwise distances between points. Smaller values of  $\sigma$  can model complex distributions but also cause overfitting. Large values of  $\sigma$  yield results similar to the linear Mahalanobis method. For small data sets with less than 1000 points, values of  $\sigma$  between two and three times the median pairwise distance between points might be desirable [35].

Kernel functions provide only one of many ways of computing the similarity matrix  $S$ . Some other ways of computing the similarity matrix are as follows:

1. One can sparsify the similarity matrix by keeping only the mutual  $k$ -nearest neighbors of each data point. All other entries in the similarity matrix are set to 0. This type of similarity matrix is used in spectral methods [378] and it tends to favor local embeddings. Furthermore, the  $(i, j)$ th entry can be normalized by the geometric mean of the sum of row  $i$  and row  $j$  in order to provide local normalization [297]. Such local normalization is used in many settings such as the LOF method discussed in section 4.2.1 of Chapter 4.
2. One can use the pairwise distance-computation between points by using the geodesic methodology of ISOMAP. Subsequently, the pairwise distances are transformed to pairwise similarities with the use of the cosine law. This approach is discussed in [33]. Unlike spectral methods, this type of approach will tend to favor global embeddings.

By selecting a particular type of similarity matrix one can effectively control the types of outliers that one will discover in various applications.

### 3.3.8.2 Practical Issues

It is noteworthy that the kernel embedding can be  $N$ -dimensional for a data set containing  $N$  points. In the case of the Gaussian kernel, the embedding dimensionality is often more than the dimensionality of the input data set. Each eigenvector defines a dimension of the embedding. Although large eigenvectors are more important for applications like clustering, the variations along the *small* eigenvectors are more important for outlier detection.

<sup>4</sup>A kernel matrix can be mean-centered by using the update  $S \leftarrow (I - U/N)S(I - U/N)$ , where  $I$  is an identity matrix and  $U$  is an  $N \times N$  matrix containing only 1s.

<sup>5</sup>It is noteworthy that a factor of 2 is omitted from the denominator in the exponent of the Gaussian kernel function for simplicity. All discussions on choice of  $\sigma$  are consistent with this convention.



The standardization of the dimensions of the transformed representation to unit variance emphasizes the importance of these eigenvectors, as in the case of the linear Mahalanobis method. The conservative approach is to use all the nonzero eigenvectors (while excluding those nonzero eigenvectors caused by numerical errors). However, when nearly all  $N$  eigenvalues are nonzero, it is sometimes the result of overfitting, and the results can be poor. A second issue is that the size of the kernel matrix  $S$  is  $O(N^2)$  which can be very large. For example, for a data set containing a hundred thousand points, this type of approach is not practical.

One way of resolving both problems is to use an ensemble-centric adaptation [35] based on variable subsampling [32]. One can repeatedly draw a random sample from the data of size  $s$  between 50 and 1000 points. Each such sample is used to (efficiently) construct an embedding of each of the  $N$  points in a space whose dimensionality is at most  $s$ .

The first step is to construct the  $s \times s$  similarity matrix  $S$  using the sample. Subsequently, we extract the  $s$  eigenvectors and eigenvalues of  $S$  as  $S = Q\Lambda^2Q^T$ , where  $Q$  and  $\Lambda$  are both  $s \times s$  matrices. However, we drop the zero eigenvectors of  $Q$  and therefore retain only the  $k < s$  nonzero<sup>6</sup> eigenvectors of  $Q$ . The corresponding  $s \times k$  matrices are  $Q_k$  and  $\Lambda_k$ , respectively. Therefore,  $Q_k\Lambda_k$  provides the embedding of the  $s$  in-sample points.

For the remaining  $(N - s)$  out-of-sample points, an  $(N - s) \times s$  kernel similarity  $S_o$  is constructed<sup>7</sup> between the out-of-sample points and in-sample points. In other words, the  $(i, j)$ th entry in this matrix is the kernel similarity between the  $i$ th out-of-sample point and the  $j$ th in-sample point, which can also be expressed as the dot product between the points in transformed space. We will show how to use this matrix in order to derive the  $k$ -dimensional embedding of each out-of-sample data point that is consistent with the aforementioned embedding of the in-sample points.

Let the unknown  $(N - s) \times k$  matrix containing the  $k$ -dimensional embedding of the out-of-sample points (in its rows) be denoted by  $V_k$ . Since the matrix  $V_k$  is unknown, the goal is to use  $S_o$  and the in-sample embedding  $Q_k\Lambda_k$  in order to estimate it. Note that each entry in the similarity matrix  $S_o$  can be approximately expressed as the pairwise dot product of the out-of-sample points (rows of  $V_k$ ) and in-sample points (rows of  $Q_k\Lambda_k$ ) in transformed space. One can express this relationship using matrix products as  $S_o = V_k(Q_k\Lambda_k)^T$ . Then, by post-multiplying this relationship with  $Q_k\Lambda_k^{-1}$ , it is evident that the  $(N - s) \times k$  embedding matrix of these out-of-sample points is given [481] by  $V_k = S_oQ_k\Lambda_k^{-1}$ . The  $s \times k$  matrix  $Q_k\Lambda_k$  (which is the embedding of in-sample points) is stacked with the  $(N - s) \times k$  matrix  $V_k = S_oQ_k\Lambda_k^{-1}$  to create a single  $N \times k$  embedding matrix  $E$  of all  $N$  data points:

$$E = \begin{pmatrix} Q_k\Lambda_k \\ S_oQ_k\Lambda_k^{-1} \end{pmatrix} \quad (3.19)$$

Each column of  $E$  is standardized to zero mean and unit variance. This embedding is used to score all data points in one shot. Note that the mean row-vector of the standardized matrix  $E$  is a vector of zeros because of the standardization operation. The squared distance of each row of  $E$  to this row-wise mean of  $E$  (which is the origin) is reported as the outlier score of that row. As a result, each point will receive one outlier score. The score is divided by the dimensionality of  $E$  to ensure that the average score over all points in each ensemble

<sup>6</sup>Some eigenvectors might be nonzero or negative due to numerical errors. In such cases, it is advisable to use a very conservative threshold like  $10^{-8}$  to distinguish truly nonzero values from those caused by numerical errors. This is also important because numerical errors hurt the computations along small eigenvalues more severely due to re-scaling of  $D'$ .

<sup>7</sup>For some data-dependent kernels like spectral methods and ISOMAP, it is not possible to construct out-of-sample similarity matrices like  $S_o$  exactly, although approximate methods exist [79].

component is 1 (because of standardization). The entire process is repeated  $m$  times with different sample sets, so that each point receives  $m$  different outlier scores. The average score of each point is reported as the final result.

### 3.3.8.3 Application to Arbitrary Data Types

One of the beautiful aspects of this approach is that *as long as a positive semidefinite similarity matrix exists between a set of  $N$  objects*, one can always use this approach by extracting a multidimensional embedding from the similarity matrix. The main technical obstacle is that any matrix  $S$ , which can be expressed as  $DD^T$  to obtain the embedding  $D$  in transformed space, can be shown to be positive semi-definite. For example, if we computed the  $N \times N$  similarity matrix between a set of  $N$  time-series using some domain-specific function and did not find the similarity matrix  $S$  to be positive semi-definite, it means that *no embedding exists* whose dot products will give you the required similarities. There are two ways in which we can avoid this problem:

- Kernel similarity functions have been designed for a variety of data types such as time-series, strings, and graphs. These similarity functions are special because they are guaranteed to be positive semi-definite. A discussion of these different types of kernel functions may be found in [33]. Note that it is not always possible to use kernels because the similarity matrix might be defined in a domain-dependent way.
- For an arbitrary similarity matrix  $S$  with negative eigenvalues, we can drop the eigenvectors with negative eigenvalues, if their absolute magnitude is relatively small. One can view this approach as an indirect way of slightly adjusting the similarity matrix to force it to satisfy the requirements of a multidimensional embedding. The level of approximation of the similarity matrix can be shown to be dependent on the *absolute* magnitude of the dropped eigenvalues. Therefore, it is desirable for the absolute magnitudes of the negative eigenvalues to be small. An alternative approach is to regularize  $S$  to the positive semidefinite matrix  $S + \alpha I$ , where  $\alpha > 0$  is the magnitude of the most negative eigenvalue. This approach restricts itself to increasing only the self-similarity (norms) of data points without changing inter-point similarity.

The second approach should be used in cases in which one cannot use an off-the-shelf kernel, and the similarity function is domain-dependent. Nevertheless, the dropping of eigenvectors does have the drawback that some outliers might be missed.

## 3.4 One-Class Support Vector Machines

---

One-class support vector machines (SVMs) can be viewed as variants of linear and logistic regression models in which the notion of *margin* is used to avoid overfitting, just as regularization is used in regression models. These error penalties are computed with the notion of *slack variables*. Although it is possible to use squared loss (as in regression), it is more common to use other forms of the loss function (such as *hinge-loss*) in support vector machines. This section assumes a familiarity with support-vector machines for classification. The uninitiated reader is referred to [33] for an introduction to the basics of the support-vector machines.

The main problem in using support-vector machines for outlier detection is that the model is primarily designed for *two* classes which need to be separated with the use of a decision boundary (and their associated margin hyperplanes). However, in outlier detection,

the data are not labeled, and therefore (a possibly noisy) model is constructed assuming that all the provided examples belong to the normal class. In order to address this issue, it is assumed that *the origin of a kernel-based transformed representation belongs to the outlier class*. Note that the quality of the separation between the normal class and the outliers in a particular data domain and corresponding transformation will depend significantly on the validity of this assumption.

For now, we will assume that the data point  $\bar{X}$  is transformed to  $\Phi(\bar{X})$  using the unknown function  $\Phi(\cdot)$ . This transformation need not be performed explicitly, as it is performed implicitly within the optimization problem with the use of kernel similarity matrices (like the nonlinear PCA method of section 3.3.8). Note that  $\Phi(\bar{X})$  is itself a high-dimensional vector in some transformed space, and the corresponding coefficients of  $\Phi(\bar{X})$  is the vector  $\bar{W}$ , which has the same dimensionality as the transformed space. The corresponding decision boundary that separates the normal class from the outlier class is given by the following:

$$\bar{W} \cdot \Phi(\bar{X}) - b = 0 \quad (3.20)$$

Here,  $b$  is a variable that controls the bias. We should formulate the optimization problem so that the value of  $\bar{W} \cdot \Phi(\bar{X}) - b$  is positive for as many of the  $N$  training examples as possible, because all training examples are assumed to belong to the normal (positive) class. Therefore, to account for any training example in which  $\bar{W} \cdot \Phi(\bar{X}) - b$  is negative, we impose the *slack penalty*  $\max\{b - \bar{W} \cdot \Phi(\bar{X}), 0\}$ . On the other hand, the origin is rewarded for lying on the opposite side of this separator and therefore a negative value of  $\bar{W} \cdot \Phi(\bar{X}) - b$  is desirable in the case when  $\Phi(\bar{X}) = 0$ . This is possible only when  $b$  is positive. This situation is illustrated in Figure 3.7(a). Therefore, in order to reward the origin for being as far away from the separator as possible on the opposite side of the normal points, we subtract  $b$  from the objective function formulation. This has the effect of pushing the separator as far away from the origin as possible towards the normal points. Furthermore, we add the margin regularizer term, which is  $\frac{1}{2}\|\bar{W}\|^2$ . Therefore, the overall objective function is as follows:

$$\text{Minimize } J = \underbrace{\frac{1}{2}\|\bar{W}\|^2}_{\text{Regularizer}} + \underbrace{\frac{C}{N} \sum_{i=1}^N \max\{b - \bar{W} \cdot \Phi(\bar{X}_i), 0\}}_{\text{Training Data Penalty}} - \underbrace{b}_{\text{Origin Reward}} \quad (3.21)$$

The constant  $C > 1$  can be viewed as the differential weight of the normal points as compared to the outlier points. Specifically, one can view  $\nu = 1/C$  as a prior probability that a data point in the training set is an outlier. In other words, the value of  $C$  regulates the trade-off between false positives and false negatives in this model. It is noteworthy that  $\bar{W} = 0$  and  $b = 0$  is a degenerate solution to this optimization problem with zero objective function value. Therefore, it is never possible for  $b$  to be strictly negative at optimality because negative values of  $b$  lead to a strictly positive value of the objective function  $J$ . The situation with a negative value of  $b$  is shown in Figure 3.7(b), which is always suboptimal with respect to the degenerate solution. It is noteworthy that if the wrong<sup>8</sup> type of kernel transformation is used, in which the origin is not linearly separable from the data points, and the value of  $C$  is very large, it can lead to the unfortunate situation<sup>9</sup> of the degenerate solution  $\bar{W} = 0$  and  $b = 0$ .

<sup>8</sup>Mean-centering the kernel matrix could result in a degenerate solution. On the other hand, any nonnegative kernel matrix like an uncentered Gaussian kernel can always avoid the degenerate solution because all pairwise angles between points are less than  $90^\circ$  in transformed space; therefore, the origin is always linearly

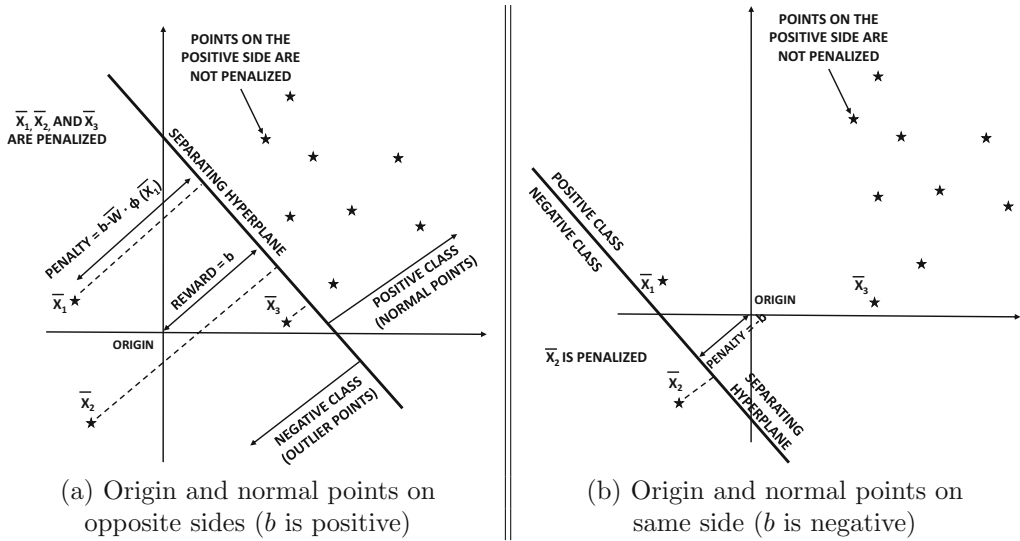


Figure 3.7: The useful solutions correspond to the case shown in (a). The case shown in (b) is always suboptimal with respect to the degenerate solution  $\bar{W} = 0$  and  $b = 0$ . Therefore,  $b$  can never be negative at optimality. Nonnegative kernels transform all points to a single orthant with a linearly separable origin and so a nontrivial solution always exists.

The positive side of the linear separator in the kernel representation corresponds to a “small” region of arbitrary shape in the *input* representation containing most of the points. Penalties are caused by points lying outside this region. An illustrative example of how the linear separator of Figure 3.7(a) might create an arbitrarily shaped region in the original input space is shown in Figure 3.8. Note that the three outliers  $\bar{X}_1$ ,  $\bar{X}_2$  and  $\bar{X}_3$  lie outside this region because they violate the linear decision boundary in transformed space. Increasing the value of  $C$  increases the volume of the enclosed region containing inliers and also changes its shape.

Note that this optimization formulation is defined in an unknown transformed space  $\Phi(\cdot)$ . Furthermore, the transformation  $\Phi(\cdot)$  is often defined *indirectly* in terms of pairwise (kernel) similarities among points. Rather than solving directly for  $\bar{W}$ , a more common approach is to predict the (optimized) value of  $\bar{W} \cdot \Phi(\bar{X}) + b$  for test point  $\bar{X}$  by using the *kernel trick* within a dual formulation. This is achieved by explicitly materializing the (non-negative) *slack variables*  $\xi_1 \dots \xi_N$  for the  $N$  training points:

$$\xi_i \geq b - \bar{W} \cdot \Phi(\bar{X}_i) \quad (3.22)$$

One can then incorporate this constraint into the optimization formulation (together with non-negativity constraints on slack variables), and substitute  $\max\{b - \bar{W} \cdot \Phi(\bar{X}_i), 0\}$  with  $\xi_i$  in the objective function  $J$ . This constrained formulation allows for an approach based on Lagrangian relaxation for which the dual formulation can be constructed. The dual

---

separable from the orthant in which all the data lies, and one rarely notices this problem in software executions. Nevertheless, the quality of the results from one-class SVMs tend to be highly unpredictable [184, 384].

<sup>9</sup>The original paper [479] incorrectly states that large training data penalties lead to negative values of  $b$ . The sensitivity of this approach to the kernel representation is, in fact, its major weakness. One possible solution to the degeneracy problem is to impose the constraint  $\|W\| = 1$  and get rid of the regularizer.

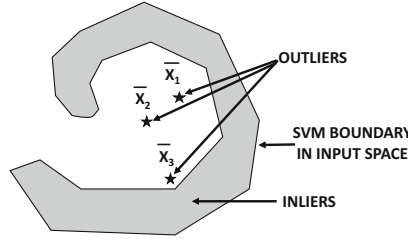


Figure 3.8: A hypothetical illustration of the linear SVM boundary of Figure 3.7(a) in the original input space. This is only an illustrative example and does not reflect an actual computation. The shape of the enclosed region is sensitive to the kernel and the parameter  $C$ .

formulation has  $N$  variables  $\bar{\alpha} = [\alpha_1 \dots \alpha_N]^T$ , each of which is a Lagrangian parameter corresponding to one of the constraints in Equation 3.22 for a training point. Interestingly, the dual formulation can be expressed in terms of the  $N \times N$  kernel similarity matrix  $S = [K(\bar{X}_i, \bar{X}_j)] = [\Phi(\bar{X}_i) \cdot \Phi(\bar{X}_j)]$  (rather than explicitly transformed points), which is the essence of the kernel trick. This similarity matrix is the same as the one that was used for nonlinear PCA in section 3.3.8. The dual formulation, described in [480], is as follows:

$$\begin{aligned} &\text{Minimize } \frac{1}{2} \bar{\alpha} S \bar{\alpha}^T \\ &\text{subject to:} \\ &0 \leq \alpha_i \leq \frac{C}{N} \quad \forall i \in \{1 \dots N\} \\ &\sum_{i=1}^N \alpha_i = 1 \end{aligned}$$

Although the coefficient vector  $\bar{W}$  is not explicitly computed by the dual approach, it can still be determined which side of the decision boundary a data point  $\bar{Y}$  lies on by using the following equivalence, which is shown in [128, 479]:

$$\bar{W} \cdot \Phi(\bar{Y}) - b = \sum_{i=1}^N \alpha_i \cdot K(\bar{Y}, \bar{X}_i) - b \quad (3.23)$$

The right-hand side can be computed as long as  $\alpha_1 \dots \alpha_N$  and  $b$  are known. The values of  $\alpha_1 \dots \alpha_N$  can be computed by solving the aforementioned dual optimization problem with gradient descent (cf. section 3.4.1). The value of  $b$  can be learned by computing  $b = \bar{W} \cdot \Phi(\bar{Y}) = \sum_{i=1}^N \alpha_i \cdot K(\bar{Y}, \bar{X}_i)$  for any training data point  $\bar{Y}$  which lies on the separating hyperplane (i.e., the data point is a *free support vector*). It can be shown that any data point  $\bar{X}_j$  for which  $0 < \alpha_j < C/N$  is a free support vector. We can average the computed value of  $b$  over all such data points.

Once the model has been trained, any data point  $\bar{X}$  (including out-of-sample points not included in the training data) can be scored using the following expression derived from the decision boundary of Equation 3.23:

$$\text{Score}(\bar{X}) = \sum_{i=1}^N \alpha_i \cdot K(\bar{X}, \bar{X}_i) - b \quad (3.24)$$

From the perspective of the modeling assumptions of the support-vector machine, a negative value of the score indicates that the data point is an outlier, whereas a positive value indicates that the data point is a non-outlier. However, one can also treat the computed value as a score, although it is sensitive to the value of  $C$ , which regulates the trade-off between outliers and inliers.

### 3.4.1 Solving the Dual Optimization Problem

The dual optimization problem can be solved using gradient-descent. However, during the descent, it is important to ensure that the constraints are not violated. The gradient of the dual objective function, which is  $\frac{1}{2}\bar{\alpha} S \bar{\alpha}^T$ , can be shown to be the  $N$ -dimensional vector  $S\bar{\alpha}$ . Therefore, the gradient-descent approach iteratively repeats the following steps after initializing each  $\alpha_i$  to  $1/N$ :

1.  $\bar{\alpha} \leftarrow \bar{\alpha} - \eta \cdot S\bar{\alpha}$
2. Set any negative values in  $\bar{\alpha}$  to 0 and any value of  $\alpha_i$  larger than  $C/N$  to  $C/N$ .
3. Scale the vector  $\bar{\alpha}$  so that  $\sum_{i=1}^N \alpha_i = 1$ .

Here,  $\eta > 0$  is the learning rate. These steps are repeated to convergence. The last two steps are executed to force the current solution to (roughly) satisfy the optimization constraints although the constraints may not be fully satisfied in the early iterations because executing step 3 might again result in  $\alpha_i$  exceeding  $C/N$ . At convergence, all the constraints will typically be satisfied. More effective steps for SVM training may be found in [128].

A number of off-the-shelf libraries such as *LIBSVM* [128] and *scikit-learn* [629] provide off-the-shelf implementations of one-class support vector machines. The *scikit-learn* code, which provides a different interface, calls the *LIBSVM* solver under the covers. It is noteworthy that *LIBSVM* uses a more sophisticated coordinate descent method rather than the simplified gradient-descent method described above. Like kernel PCA methods, one class support-vector machines have the desirable property that they can be used for arbitrary data types provided that a kernel similarity function can be defined between data objects.

### 3.4.2 Practical Issues

An insightful evaluation of some of these models for outlier detection in document data is provided in [384]. In particular, the main challenge associated with support-vector machines is that these methods can be sensitive to the choice of the kernels and the many hidden parameters associated with the method such as the value of  $C$  and the parameters associated with the kernel. The evaluation in [384] makes the following observations about the one-class support-vector machine:

“However, it turns out to be surprisingly sensitive to specific choices of representation and kernel in ways which are not very transparent. For example, the method works best with binary representation as opposed to tf-idf or Hadamard representations which are known to be superior in other methods. In addition, the proper choice of a kernel is dependent on the number of features in the binary vector. Since the difference in performance is very dramatic based on these choices, this means that the method is not robust without a deeper understanding of these representation issues.”

In another recent experimental evaluation of several detectors [184], the one-class SVM was the most poorly performing detector and frequently provided worse-than-random performance.

The one-class SVM assumes that the origin is a prior for the outlier class, which is not optimal even in kernel feature space. For example, a simple operation such as mean-centering the kernel matrix can have unfortunate effects. As a general rule, it is harder to use kernels in unsupervised problem settings (as compared to supervised settings) because of the natural difficulties in model selection and parameter tuning. For example, for the Gaussian kernel, the median of the pairwise distance between points provides a rough approximation to the bandwidth  $\sigma$ , although the value of  $\sigma$  is also sensitive to the data distribution and size. A second issue is that the size of the kernel matrix  $S$ , which also defines the number of terms in the dual objective function, is  $O(N^2)$ . For example, for a data set containing a hundred thousand points, this type of approach is not practical.

Both these problems can be partially solved using variable subsampling [32], which works particularly well in the context of unpredictable data size-sensitive parameters. The basic idea is to repeatedly sample a variable number of training points from the data, which vary between  $n_{min} = 50$  and  $n_{max} = 1000$ . Therefore, in each training model the size of the kernel matrix is at most  $1000 \times 1000$ . Subsequently, all  $N$  points are scored with respect to this training model, and the scores are normalized to Z-values. This is possible in kernel support-vector machines, because out-of-sample points can be scored with the learned model. In fact, the out-of-sample scores can often be more robust because of less overfitting. In each training model constructed from the sample, a different kernel function and choice of parameters (within a reasonable range) may be used. This process can be repeated as many times as needed. The final outlier score can be reported as the average of the score across various detectors. This approach is an ensemble technique, which is discussed in detail in Chapter 6.

### 3.4.3 Connections to Support Vector Data Description and Other Kernel Models

The one-class SVM is intimately connected to many other kernel models. The support-vector data description (SVDD) [539] is a different kernel SVM method in which data is enclosed in a hypersphere of radius  $R$  in the transformed feature space (rather than a linear separator from the origin). The squared radius is minimized together with penalties for margin violation. The SVDD approach is closely related both to the one-class SVM of section 3.4 as well as to the kernel Mahalanobis method discussed in section 3.3.8.

The SVDD and linear SVM models are roughly<sup>10</sup> equivalent if the embedded points have an origin-centered spherical geometry [539]. The most common example is that of a Gaussian kernel, which embeds all points on the unit sphere. *The solutions are otherwise quite different.* For example, the degeneracy problem of one-class SVMs, which is discussed in section 3.4, is not encountered in SVDD even with mean-centered kernels. In fact, the SVDD predictions do not change by mean-centering the kernel matrix, because the origin is not assumed as a prior. On the other hand, the SVDD approach performs poorly with polynomial kernels because the data tends to be elongated in particular directions in the

<sup>10</sup>We use the term “roughly” because the SVDD optimization formulation (with the Gaussian kernel) can be transformed into a formulation like one-class SVM in which a normalization constraint  $\|\bar{W}\| = 1$  is imposed instead of including the regularizer  $\|\bar{W}\|^2/2$  in the objective function [539]. The experimental results in [184] suggest that SVDD provides slightly better solutions than one-class SVMs with the Gaussian kernel.

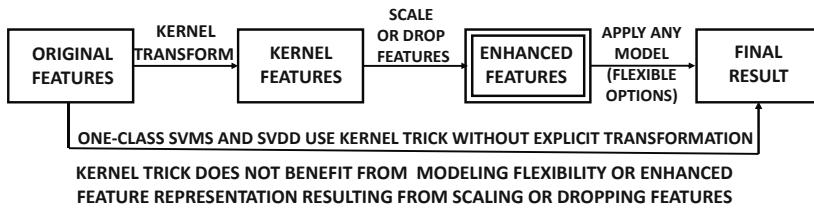


Figure 3.9: Relationship between explicit and implicit kernel transformations

transformed feature space, and a sphere fits poorly around it.

The kernel Mahalanobis method is closely related to SVDD. The former can be viewed as a way of creating a new representation of the data in which the distance to the center of the transformed data directly provides the outlier score, and there is no need to find a hard boundary of specific radius  $R$  with an optimization model. A key difference between the kernel Mahalanobis method and other one-class methods is the use of feature-space scaling of all transformed dimensions to unit variance in the former. This type of normalization is especially helpful because it prevents high-variance directions from masking the outliers. The normalization step can provide additional benefits that are not available in the original one-class SVM or SVDD solution based on the kernel trick within the dual. For example, one could very easily transform the data to unit-variance directions in feature space using the approach in section 3.3.8 and then apply a *linear* one-class SVM on the embedded representation to earn these additional benefits. In fact, the use of a circular separator (like SVDD) on the *normalized* kernel representation works reasonably well with a polynomial kernel, which is generally known to work poorly with SVDD. It is helpful to think of the nonlinear transformation in section 3.3.8 as a (normalized) *data-specific Mercer kernel map*, on top of which many trivially simple anomaly detection models (e.g., distance from centroid) become extremely effective. Furthermore, one gains the flexibility of using any other outlier detection method on the extracted representation. For example, one might use mixture modeling, distance-based methods [475], or clustering (cf. section 4.2.1 of Chapter 4) on the kernel representation. Explicit feature mapping with kernels is highly under-appreciated in spite of its obvious flexibility in enabling the use of other models, or in enhancing the underlying representation by feature scaling/selection. For example, dropping small eigenvectors leads to better accuracy in clustering and two-class SVMs [33, 481] because of de-noising effects. Similarly, dropping large eigenvectors sometimes leads to better accuracy of one-class SVMs because of outlier enhancement effects; this is simply a hard version of the softer feature scaling. Furthermore, the computational advantages of working with the dual are not guaranteed with respect to out-of-sample implementations of explicit feature mapping (see Equation 3.19); the only “disadvantage” of explicit transformation is that it renders the use of the endearing kernel trick as redundant within the SVM formulation once it has been used for creating the embedding. The relationship between explicit and implicit kernel transformation methods is illustrated in Figure 3.9.

The one-class support-vector machine may also be viewed as an approach that (roughly) finds a maximum margin separation of the data matrix  $D$  with its negative set  $-D$  (instead of separating  $D$  from the origin) in kernel feature space. This intuitive view provides a clearer understanding of the relationship to two-class SVMs, and it also facilitates the adaptation of other two-class optimization models such as the Fisher’s linear discriminant to the one-class setting. This view has been used to adapt the kernel Fisher discriminant method to



anomaly detection [466]. The main difference between the kernel Fisher approach [466] and kernel SVMs is that the former identifies a direction that maximizes the ratio of inter-class separation to intra-class separation (between  $D$  and  $-D$ ), whereas the latter finds the maximum margin separator between  $D$  and  $-D$ . However, both methods implicitly transform these data to the same kernel feature representation before optimization.

Among all these models, the kernel Mahalanobis method has the advantage of requiring the least number of free parameters (which only correspond to the kernel setting). The other methods attempt to find a *hard* boundary (cf. Figure 3.8) between outliers and inliers and therefore must use parameters like  $C$  to regulate the trade-off between outliers and inliers *early on* in the modeling. The Mahalanobis method is a soft approach that focuses on finding scores and (appropriately) leaves the hard labeling to the end, when more insight is available on the score distribution. Minimizing the number of user-driven parameters is always desirable in unsupervised problems like outlier detection especially because the underlying kernel feature representations are semantically opaque. An evaluation of the kernel Mahalanobis method in an ensemble-centric setting may be found in [35].

### 3.5 A Matrix Factorization View of Linear Models

---

PCA can be viewed as a type of matrix factorization. In order to understand this point, consider the rank- $k$  representation of PCA from a mean-centered data matrix  $D$ .

$$D' = DP_k \tag{3.25}$$

Here,  $D'$  is the  $N \times k$  reduced representation of matrix  $D$ , and  $P_k$  is a  $d \times k$  matrix containing the largest  $k$  (orthonormal) eigenvectors of the covariance matrix  $\Sigma = \frac{D^T D}{n}$  in its columns. As discussed earlier,  $D'$  also corresponds to  $Q_k \Lambda_k$ , where the rank- $k$  diagonalization of the dot-product (i.e., similarity) matrix  $DD^T$  is given by  $Q_k \Lambda_k^2 Q_k^T$ . Here,  $Q_k$  is an  $N \times k$  matrix containing the orthonormal eigenvectors of  $DD^T$ . Therefore, we have:

$$D' = DP_k \approx Q_k \Lambda_k \tag{3.26}$$

Interestingly, it can be shown that the matrices  $Q_k$ ,  $\Lambda_k$  and  $P_k$  can be used to create a rank- $k$  factorization of the original matrix  $D$ . By post-multiplying each expression in Equation 3.26 with the matrix  $P_k^T$  and setting  $P_k P_k^T = I$ , we obtain:

$$D \approx Q_k \Lambda_k P_k^T \tag{3.27}$$

This particular relationship is also referred to as the rank- $k$  Singular Value Decomposition (SVD) of the data matrix  $D$ . By absorbing the diagonal matrix in one of the factors, one can express  $D$  as a factorization into two matrices, each of which have orthogonal columns. In particular, we define the  $N \times k$  matrix  $U$  as  $Q_k \Lambda_k$  and the  $d \times k$  matrix  $V$  as  $P_k$ . Therefore, SVD can be expressed as a factorization into two low-rank matrices as follows:

$$D \approx UV^T \tag{3.28}$$

Interestingly, the matrices  $U$  and  $V$ , can be learned by solving the following optimization problem:

$$\begin{aligned} &\text{Minimize } \|D - UV^T\|^2 \\ &\text{subject to:} \\ &\quad \text{Columns of } U \text{ are mutually orthogonal} \\ &\quad \text{Columns of } V \text{ are mutually orthonormal} \end{aligned}$$

Here,  $\|\cdot\|^2$  represents the Frobenius norm of the error matrix  $(D - UV^T)$ . The Frobenius norm is defined as the sum of the squares of the entries. Note that this interpretation is consistent with mean-squared error optimization of PCA. It is also noteworthy that the absolute values of the entries of  $D - UV^T$  provide *entry-wise* outlier scores, which are more detailed than the *row-wise* outlier scores discussed in earlier sections. These values represent the inability of the compressed representation  $UV^T$  to fully explain the values in the original matrix because of their deviations from the underlying correlation structure.

This view of SVD is particularly useful because it paves the way for using other types of matrix factorization methods. For example, the objective function can be changed in a variety of ways, regularization can be added, and the constraints can be modified to incorporate specific (desired) properties into the factor matrices  $U$  and  $V$ . The simplest generalization is one in which the orthogonality constraints on  $U$  and  $V$  are removed; this leads to *unconstrained* matrix factorization. Alternatively, for a non-negative data matrix  $D$ , we can impose non-negativity constraints on the factors to create non-negative representations, which are more interpretable. This type of factorization is very useful in text data and network data. In Chapter 12, we will provide a specific example of such a factorization. Finally, the most useful aspect of this type of matrix factorization is that it can be used for *anomaly detection in incomplete data sets*, and even provide insights about the specific *entries* of the matrix that lead to anomalies. This is not possible with the straightforward version of PCA, which is undefined for incomplete data matrices. In the next section, we will provide an example of unconstrained matrix factorization of incomplete data.

### 3.5.1 Outlier Detection in Incomplete Data

In this section, we will show how unconstrained matrix factorization can be used to discover anomalous rows or even anomalies *entries* of an incomplete data matrix. The latter can be useful in an application such as recommender systems. In a recommender system, we might have an  $N \times d$  ratings matrix  $D$  with  $N$  users and  $d$  items. It may be desirable to discover anomalous ratings, such as the fake ratings created by “shills” in the recommender system. In such applications, each column (item) might typically contain less than 100 ratings over a population of millions of users. In such sparse settings, even the covariance matrix cannot be estimated accurately, as needed in PCA. In other applications, many data values might be missing because of weaknesses in data collection mechanism. For example, in a user survey, users might choose to leave many fields blank. In such settings, it might be desirable to discover anomalous *rows* of the data matrix. All these complex settings can be addressed with a straightforward generalization of the aforementioned PCA model.

For, the purpose of the following discussion, we will assume that the  $(i, j)$ th entry of  $D$  (when not missing) is denoted by  $x_{ij}$ . Consider a setting in which the set of entries in the matrix that are specified (i.e., not missing) is denoted by  $H$ . In other words, we have the following:

$$H = \{(i, j) : x_{ij} \text{ is observed (not missing)}\} \quad (3.29)$$

Then, we would like to factorize the data matrix  $D$  into the representation  $UV^T$ , so that  $U = [u_{is}]$  is an  $N \times k$  matrix,  $V = [v_{js}]$  is a  $d \times k$  matrix, and  $k$  is the rank of the factorization. The predicted value of an entry  $(i, j)$  is  $\sum_{s=1}^k u_{is}v_{js}$ , and we wish to minimize the aggregate error with respect to observed values. However, since we know only the subset of entries  $H$  in the data matrix, we must formulate the optimization problem *only over the specified entries*. Furthermore, it is important to use regularization to avoid overfitting because the number of specified entries might be small. This optimization problem can be written (together with

a regularization term) as follows:

$$\text{Minimize } J = \underbrace{\frac{1}{2} \sum_{(i,j) \in H} (x_{ij} - \sum_{s=1}^k u_{is}v_{js})^2}_{\text{Error on observed entries}} + \underbrace{\frac{\alpha}{2} (\|U\|^2 + \|V\|^2)}_{\text{Regularizer}}$$

subject to:

No constraints on  $U$  and  $V$

Here,  $\alpha > 0$  is the regularization parameter and the squared Frobenius norm on the factor matrices is included in the objective function. We have dropped the constraints on  $U$  and  $V$  to simplify the optimization process. For observed entries in  $D$ , the error  $e_{ij}$  of the  $(i, j)$ th entry is the difference between the observed value  $x_{ij}$  and the predicted values  $\sum_{s=1}^k u_{is}v_{js}$ :

$$e_{ij} = x_{ij} - \sum_{s=1}^k u_{is}v_{js} \quad (3.30)$$

Note that the error term is defined only for the observed entries in  $H$ . We can, therefore, rewrite the objective function  $J$  as follows:

$$J = \frac{1}{2} \sum_{(i,j) \in H} e_{ij}^2 + \frac{\alpha}{2} (\|U\|^2 + \|V\|^2) \quad (3.31)$$

This error term is key; solving the optimization problem yields the (squared) error terms as the outlier scores of the individual *entries* in the data matrix.

The optimization problem can be solved using gradient descent. Therefore, we can compute the partial derivatives of the objective function with respect to the parameters  $u_{is}$  and  $v_{js}$ :

$$\begin{aligned} \frac{\partial J}{\partial u_{is}} &= \sum_{j:(i,j) \in H} e_{ij}(-v_{js}) + \alpha \cdot u_{is} \\ \frac{\partial J}{\partial v_{js}} &= \sum_{i:(i,j) \in H} e_{ij}(-u_{is}) + \alpha \cdot v_{js} \end{aligned}$$

In gradient-descent, we create an  $[(N + d) \cdot k]$ -dimensional vector  $\overline{W}$  of parameters corresponding to the entries in  $U$  and  $V$  and make the updates  $\overline{W} \leftarrow \overline{W} - \eta \nabla J$ . Note that  $\nabla J$  is defined by the entire vector of  $(N + d) \cdot k$ -dimensional vector of partial derivatives computed above. One can equivalently perform these updates with sparse matrix multiplication. Let  $E$  be an  $N \times d$  sparse matrix in which only specified entries (i.e., entries in  $H$ ) take on the value of  $e_{ij}$ . The missing entries take on zero values. Note that it makes sense to compute only the observed entries in  $E$  and use a sparse data structure to represent  $E$ . The aforementioned gradient-descent steps can be shown to be equivalent to the following matrix-wise updates:

$$\begin{aligned} U &\leftarrow U(1 - \alpha \cdot \eta) + \eta EV \\ V &\leftarrow V(1 - \alpha \cdot \eta) + \eta E^T U \end{aligned}$$

These iterations can be executed to convergence. The value of  $\eta > 0$  corresponds to the learning rate. Selecting the learning rate to be too large might result in overflows. On the other hand, a very small learning rate will lead to convergence which is too slow. A faster variant of this approach is stochastic gradient descent in which we cycle through the observed entries  $x_{ij}$  in  $H$  in random order and make the following updates:

$$\begin{aligned} u_{is} &\leftarrow u_{is}(1 - \alpha \cdot \eta) + \eta e_{ij} v_{js} & \forall s \in \{1 \dots k\} \\ v_{js} &\leftarrow v_{js}(1 - \alpha \cdot \eta) + \eta e_{ij} u_{is} & \forall s \in \{1 \dots k\} \end{aligned}$$

The resulting iterations are also executed to convergence. More details may be found in [34].

### 3.5.1.1 Computing the Outlier Scores

It remains to show how one can compute the entry-wise or row-wise outlier scores from the factors. Note that factorization approximated reconstructs the original data matrix (as in SVD). Deviations from the reconstructed values are presumed to be outliers. The error  $e_{ij}$  for the observed entry  $x_{ij}$  is defined according to Equation 3.30:

$$e_{ij} = x_{ij} - \sum_{s=1}^k u_{is} v_{js} \quad (3.32)$$

These errors are also referred to as *residuals*. Entries with large positive or negative residuals tend to be outliers because they do not conform to the normal model of low-rank factorization. Therefore, we use the squares of these entries as the outlier scores.

We can define the row-wise outlier score in a similar way. For any particular row of the data matrix, its outlier score is defined as the mean of the squared-residuals in its observed entries. Therefore, for the  $i$ th row  $\overline{X}_i$  of  $D$ , we define its outlier score as follows:

$$\text{Score}(\overline{X}_i) = \frac{\sum_{j:(i,j) \in H} e_{ij}^2}{n_i} \quad (3.33)$$

Here,  $n_i$  denotes the number of observed entries in  $\overline{X}_i$ . Interestingly, it is possible to define column-wise outlier scores in an exactly analogous way. This can be useful in some applications. For example, in a collaborative filtering application, the merchant might be interested in items (columns) with unusual ratings patterns. Matrix factorization is one of the most general forms of linear modeling and it finds applications in network data as well. Some of these applications will be discussed in Chapter 12.

## 3.6 Neural Networks: From Linear Models to Deep Learning

---

Neural networks are computational learning models that simulate the human nervous system. In humans, learning is performed by changing the strength of synaptic connections between cells, which are referred to as *neurons*. The strengths of synaptic connections are changed in response to artificial stimuli. In the case of *artificial* neural networks, the individual nodes are referred to as *neurons*. The neurons receive inputs from other neurons using weighted connections (or from external training data), and might in turn transmit their outputs to other neurons after performing a computation on their inputs. Just as synaptic strengths are changed in biological neural networks for learning, artificial neural

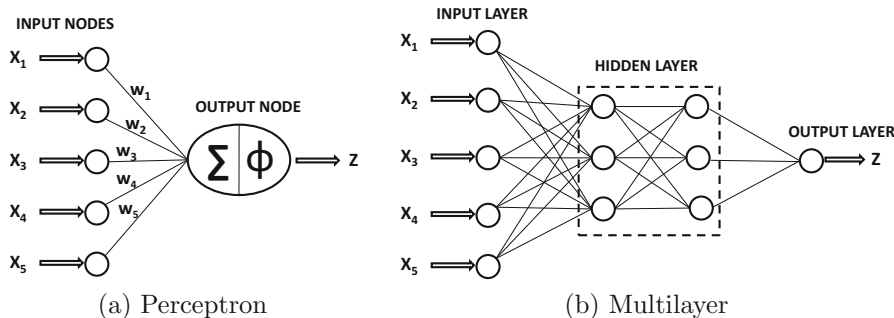


Figure 3.10: Single- and multilayer neural networks

networks change their weights in response to their inputs. The most basic form of input to a neuron is a feature of the training data, which can be considered the analog of the external stimulus used in biological neural networks.

The simplest form of a neural network is the *perceptron*. In its most basic form, the perceptron is virtually identical to a simple linear regression model or PCA/matrix factorization model. However, its neat conceptualization as a unit of computation allows us to put multiple perceptrons together in a multilayer network. This type of multilayer network allows the computation of any nonlinear function. For this reason, neural networks are referred to as *universal function approximators*. The perceptron contains two layers of nodes, which correspond to the input nodes and a single output node. The number of input nodes is exactly equal to the dimensionality  $d$  of the data. Let  $\bar{X} = (x_1 \dots x_d)$  be the  $d$  inputs, which correspond to the  $d$  feature values of the data records. The output of a perceptron is computed as a function of the underlying inputs with the associated weight vector  $\bar{W} = (w_1 \dots w_d)$ :

$$z = \bar{W} \cdot \bar{X} = \sum_{i=1}^d w_i x_i \tag{3.34}$$

This function is also referred to as the *linear activation function*. It is noteworthy that the weights in the activation function are exactly analogous to those used in section 3.2 for linear regression and rank-1 PCA. In general settings, however, we can use an arbitrary activation function of this linear model and also incorporate a bias term:

$$z = \Phi(\bar{W} \cdot \bar{X} + b) \tag{3.35}$$

As we will discuss later, the activation function  $\Phi$  is often a nonlinear function like the sigmoid or tanh functions. However, for now, we will work with the simpler form discussed in section 3.34 because of its similarity to the other models discussed in this chapter. A pictorial representation of the perceptron architecture is illustrated in Figure 3.10(a).

Neural networks can be used for either of the two settings discussed in section 3.2; recall that in one setting (cf. section 3.2.1) the dependent variable is treated as special, whereas in the other setting (cf. section 3.2.2) all attributes are treated in a homogeneous way and the mean-squared projection error on the hyperplane is minimized. The first type of setting is used to create *replicator neural networks* [160, 250] in which each attribute of the data is predicted using other attributes, and the errors of the prediction are used to quantify outlier scores. Such types of neural networks are also referred to as *autoencoders* and are discussed in section 3.6.2. A generalization of this approach with the use of *any* regression model (and not just neural networks) is discussed in section 7.7 of Chapter 7.

In this section, we will focus on two different types of neural networks. First, we will focus on a setting in which we are trying to create *one-class* neural networks in which the output of the network is always zero in spite of the fact that the weights are nonzero. This setting is somewhat less common and used rarely. However, we include it because it is directly related to the optimization approach discussed in section 3.2.2. In section 3.6.2, we will discuss the use of replicator neural networks and autoencoders for outlier detection. Both types of methods can be understood within the framework of dimensionality reduction methods, although the autoencoder framework is more general and potentially more powerful. This approach is used more commonly in outlier detection.

Since all training points are assumed to be normal points in the one-class setting, the prediction  $z$  of Equation 3.34 is expected to be 0. Note that this is *exactly identical* to the assumption in Equation 3.9 for linear modeling which is replicated here:

$$\sum_{i=1}^d w_i \cdot x_i \approx 0 \quad (3.36)$$

Therefore, any non-zero value of  $z$  predicted by the one-class neural network is assumed to be a result of outliers that do not conform to the model of normal data. Therefore, for a single instance  $\overline{X}_i$ , in which the neural network predicts  $z_i$ , the squared error for the  $i$ th point from our one-class assumption is as follows:

$$J_i = z_i^2 = (\overline{W} \cdot \overline{X}_i)^2 \quad (3.37)$$

Therefore, one must update the weights of the neural network to account for this error. This is achieved with a gradient-descent update. This update may be written as follows:

$$\begin{aligned} \overline{W} &\leftarrow \overline{W} - \eta \nabla J_i \\ &= \overline{W} - \eta z_i \overline{X}_i \end{aligned}$$

Here,  $\eta > 0$  is the learning rate. Also, in order to avoid the trivial solution  $\overline{W} = 0$ , the updated vector  $\overline{W}$  is scaled to unit norm. The training phase of the neural network feeds the  $d$ -dimensional records  $\overline{X}_1 \dots \overline{X}_N$  to the perceptron one by one, and performs the aforementioned updates to the vector  $\overline{W}$  until convergence is reached. This entire process is a version of stochastic-gradient descent, and the result would be almost identical to the solution we would obtain with PCA (cf. section 3.2.2) or with matrix factorization (cf. section 3.5) with the rank set to  $(d - 1)$ .

In order to score a given data point  $\overline{X}_i$ , we use the learned model to compute its outlier score as follows:

$$\text{Score}(\overline{X}_i) = (\overline{W} \cdot \overline{X}_i)^2 \quad (3.38)$$

Outliers will have larger scores. It is possible to score out-of-sample points using this model. In fact, as discussed later in section 3.6.3, out-of-sample points will have more robust outlier scores because they do not overfit the training data. The perceptron is equivalent to the use of matrix factorization or PCA with rank  $(d - 1)$ , which is equivalent to scoring the data only along the smallest eigenvector after projecting it into the space of top- $(d - 1)$  principal components. With this type of conservative approach to scoring, it is possible for all points to have outlier scores of 0 if the data set has rank strictly less than  $d$ . As a result, many true outliers will be missed. This is a classical manifestation of overfitting. Therefore, a more reasonable approach is to train the neural network using only half the points, and then score the remaining half as *out-of-sample* test points. The scores of the test points are

standardized to zero mean and unit variance. The process is repeated multiple times over multiple random samples, and the scores of points are averaged. An alternative approach with the use of multiple output nodes will be discussed later.

### 3.6.1 Generalization to Nonlinear Models

So far, it does not seem as if one-class neural networks have achieved anything different from what we would be able to accomplish with a special case of matrix factorization or PCA. So, what is the point of this entire exercise? The major point is that the conceptualization of such models as neural network units helps us put them together in a *multilayer neural-network architecture*, which can model arbitrarily complex patterns in the underlying data. In other words, the conceptualization of a perceptron provides a black-box framework for “putting together” these simpler models into a more complex model. In fact, the generality of neural networks is even greater than the nonlinear PCA approach discussed in section 3.3.8 in terms of the types of decision boundaries that such a technique can model. Technically, given sufficient data, a multilayer neural network with a modest number of units can model virtually any one-class data distribution without making any assumptions about the shape of this distribution. As a result, neural networks are also sometimes referred to as “universal function approximators.”

Multi-layer neural networks have an additional *hidden layer*, in addition to the input and output layers. The hidden layer might itself be connected in different types of topologies. A common type of topology is one in which the hidden layer has multiple layers, and the nodes in one layer feed forward into the nodes of the next layer. This is referred to as a *feed-forward network*. An example of a feed-forward network with two hidden layers is illustrated in Figure 3.10(b). Furthermore, one need not use linear functions in any of the layers. Various activation functions  $\Phi(\cdot)$  (based on Equation 3.35) such as the *tanh* and *sigmoid* functions are used.

$$\Phi(z) = \frac{e^{2z} - 1}{e^{2z} + 1} \quad (\text{tanh function})$$

$$\Phi(z) = \frac{1}{1 + e^{-z}} \quad (\text{sigmoid function})$$

$$\Phi(z) = \exp\left(-\frac{(z - \mu)^2}{2\sigma^2}\right) \quad (\text{Gaussian Radial Basis Function})$$

In the single-layer neural network, the training process is simple because the expected output of the neuron is known to be 0. The problem in the case of the hidden layer is that we do not know what the outputs of neurons in these units should be. We only know that the final output at the output layer should be 0. In other words, some type of feedback is required from later layers to earlier layers about the expected output. This is achieved with the use of the *backpropagation* algorithm. The backpropagation algorithm has a forward phase and a backward phase, which is applied during the training of each instance. In the forward phase, the activation function is applied first in the input layer, then in the hidden layer, until the output propagates to the output layer. Then the error is computed, and the error estimates of various neurons also propagate backwards. These are then used to update the weights of various nodes. As in the case of the perceptron, it is important to scale the weights of the output node to unit norm throughout the update process. This is done in order to avoid the trivial solution where all weights in the output node are 0s. Furthermore, if  $W$  is the  $h_1 \times h_2$  matrix of weights between any particular pair of hidden

layers with  $h_1$  and  $h_2$  nodes (where  $h_2 \leq h_1$ ), we need to impose the constraint  $W^T W = I$  to avoid trivial transformations in which each node in a hidden layer corresponds to the same transformation or a zero output. These additional requirements necessitate the use of constrained gradient-descent methods, which are generally much harder.

As in the case of the perceptron, this process is applied for each training data point; furthermore, we cycle through the various training data points until convergence is reached. A detailed discussion of the backpropagation algorithm is provided in [84]. By increasing the number of layers in the network and using different types of activation functions, we can model arbitrarily complex non-linear patterns. The process of learning the parameters of a neural network with a large number of layers requires a number of specialized tricks; this class of methods is referred to as *deep learning* [223].

**Reducing Representation Rank with Multiple Outputs:** The aforementioned approach uses only one output node. In the case of the perceptron, this corresponds to a linear reduction of rank  $(d - 1)$ . This is equivalent to using only the scores along the smallest eigenvector in PCA. In practice, this is too large a representation rank to obtain a meaningful reduction. One way of reducing the representation rank is to use multiple (say  $r$ ) output nodes so that the output of each node is expected to be zero. This case is analogous to scoring along the  $r$  smallest eigenvectors in PCA (i.e., using a representation rank of  $(d - r)$ ). Therefore, the error is equal to the sum of the squares of the outputs of the various nodes. In addition, we need to incorporate some constraints on the weights in the output layer. Let  $W$  be the  $h \times r$  matrix of weights in the output layer, where  $h \geq r$  is the number of nodes in the last hidden layer. In order to ensure mutual orthogonality and normalization of the weight vector, we need to impose the additional constraint  $W^T W = I$ . Such an approach will require constrained gradient descent for the weights in the output layer. This makes the optimization problem more challenging. A more satisfactory and interpretable way of achieving these goals is the use of replicator neural networks, which will be discussed in the next section.

**Outlier Scoring:** For any given point  $\overline{X}_i$ , if the multilayer network with a single output node outputs  $z_i$ , the outlier score is  $z_i^2$ . If the neural network has  $r$  outputs  $z_i(1) \dots z_i(r)$ , the outlier score is  $\sum_{j=1}^r z_i(j)^2$ . Note that these scores are constructed in a very similar way to the approach used in matrix factorization and PCA.

### 3.6.2 Replicator Neural Networks and Deep Autoencoders

Although the approach in the previous section is intuitively appealing because of its natural relationship to supervised learning in traditional neural networks, it is rarely used. This is because of the constraints in the underlying optimization problem and the fact that it cannot be used to derive a compressed representation of the data. A more common methodology is to use *autoencoders*. An example of an autoencoder with three hidden layers is shown in Figure 3.11. Note that the number of outputs is the same as the number of inputs, and each input  $x_i$  is reconstructed to  $x'_i$  for the  $i$ th dimension. The aggregate error of reconstruction  $\sum_{i=1}^d (x_i - x'_i)^2$  over all  $d$  dimensions is summed up over all data points, and is minimized by neural network training. The point-specific error of reconstruction, which is  $\sum_{i=1}^d (x_i - x'_i)^2$ , provides the outlier score of that point. The use of three hidden layers in replicator neural networks is common, and was also used in the approach discussed in [250].

Autoencoders are a natural choice for outlier detection because they are commonly used for dimensionality reduction of multidimensional data sets as an alternative to PCA or matrix factorization. Note that the number of nodes in the middle hidden layer of Figure 3.11 is much less than the number of nodes in the input layer (which is very typical), and the



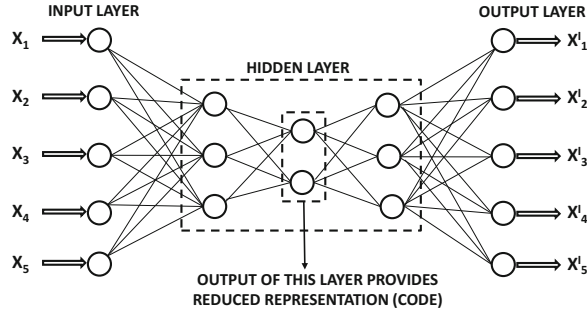


Figure 3.11: The autoencoder architecture: It is instructive to compare the autoencoder architecture with that in Figure 3.10(b).

output of the nodes in the middle hidden layer can be viewed as a reduced representation of the data. The use of neural networks for dimensionality reduction is discussed in [259, 264]. In this context, it has been shown [68] that certain simplified architectures of autoencoders yield dimensionality reductions that are closely related to those obtained using principal component analysis. Note that the architecture of the neural network on both sides of the middle hidden layer is often (but not necessarily) symmetric. In fact, one can divide the autoencoder on two sides of the middle hidden layer into two portions corresponding to an encoder and a decoder, which provides a point of view very similar to that in matrix factorization and PCA.

In order to understand why autoencoders are effective for outlier detection, we will first understand traditional matrix factorization as a method for encoding and decoding data. The factorization  $D \approx UV^T$  can be viewed as a kind of encoder that compresses the data matrix  $D$  into low-rank factors  $U$  and  $V$ . Furthermore, the product  $UV^T$  provides a reconstructed matrix  $D' = UV^T$ . Therefore, the multiplication of  $U$  and  $V^T$  can be viewed as a type of decoder. Note that  $D'$  is not exactly the same as  $D$  and the absolute values of the entries in  $(D - D')$  provide the entry-wise outlier scores. The entire process of matrix factorization (in the context of an encoder-decoder architecture) is illustrated in Figure 3.12(a).

When using PCA or matrix factorization, one makes the implicit assumption of linear compression. However, the multilayer neural network architecture provides a more general type of dimensionality reduction in which any type of nonlinear reduction is possible with the neural network model. In fact, by splitting up the replicator neural network of Figure 3.11 on two sides of the middle hidden layer, we obtain a multilayer neural network on each side of the middle hidden layer corresponding to the encoder and the decoder portions. This situation is illustrated in Figure 3.12(b). The first part of the network learns the encoding function  $\phi$  and the second part of the neural network learns the decoding function  $\psi$ . Therefore,  $\phi(D)$  represents a compressed representation (i.e, dimensionality reduction) of the data set  $D$ , and applying the decoder function  $\psi$  to  $\phi(D)$  yields the reconstructed data  $D' = (\psi \circ \phi)(D)$ , which may not be exactly the same as  $D$ . Outlier entries are resistant to compression and will show the greatest variation between  $D$  and  $D'$ . The absolute values of the entries in the residual matrix  $(D - D')$  provide the entry-wise outlier scores. As in the case of matrix factorization, one can convert these entry-wise scores to either row-wise scores or column-wise scores. The main difference from matrix factorization is the greater power of autoencoders in representing arbitrary data distributions. With a larger number

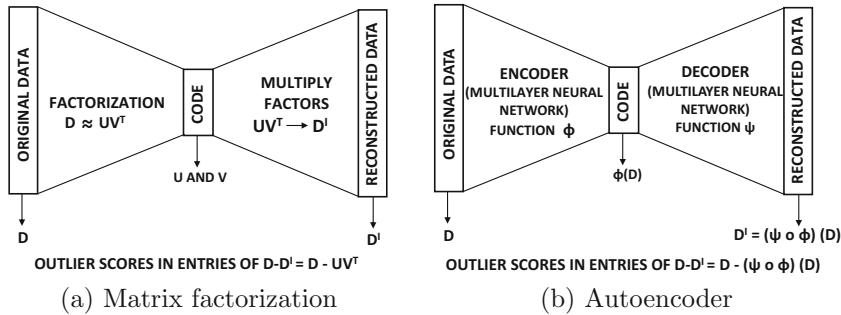


Figure 3.12: Similarities between autoencoders and dimensionality reduction/matrix factorization methods

of hidden units, one can obtain a low-dimensional representation of any data distribution whether it is linear or not. In fact, one can often model more complex distributions with autoencoders than with kernel PCA.

Although the splitting of the replicator neural network into an encoder and a decoder is useful in understanding the relationship to dimensionality reduction, it is not necessary in practice. The splitting of a replicator neural network into an encoder-decoder pair is a particular type of architecture (referred to as n/p/n-architecture) that allows the creation of a compressed representation. One can work with even more general architectures if one does not explicitly need the reduced representation of the data. In the original work on replicator neural networks [250], three hidden layers were used. The tanh activation function was used in the hidden layers, and the linear or sigmoid function is used in the output layer. The middle hidden layer used a stepwise variation on the tanh function. The error in the output layer is given by the reconstruction error and backpropagation of these errors is used to train the neural network. The trade-off between the generality of the approach and the tendency to overfit is regulated by the number of hidden layers, and the number of units in each hidden layer. The number of hidden layers and the number of hidden nodes in each layer can be determined empirically with the use of a validation set [160]. The validation set is also useful for determining the termination criterion for the training process. The training phase is terminated when the error on the validation set begins to rise.

The success of the approach follows from its ability to model complex nonlinear distributions, although one must always guard against excessively increasing the number of hidden layers and causing overfitting. As shown in [264], careful design choices can provide better reductions than PCA with neural networks. In particular, when working with very deep networks, an unsupervised method, referred to as *pretraining* [80, 459] is essential in achieving a meaningful dimensionality reduction. In pretraining, a greedy approach is used to train the network one layer at a time by learning the weights of the outer hidden layers first and then learning the weights of the inner hidden layers. The resulting weights are used as starting points for a final phase of traditional neural network backpropagation in order to fine tune them.

An example of pretraining for the network of Figure 3.11 is shown in Figure 3.13. The basic idea is to assume that the two (symmetric) outer hidden layers contain a first-level reduced representation of larger dimensionality, and the inner hidden layer contains a second level reduced representation of smaller dimensionality. Therefore, the first step is to learn the first-level reduced representation and the corresponding weights associated with the outer hidden layers using the simplified network of Figure 3.13(a). In this network, the

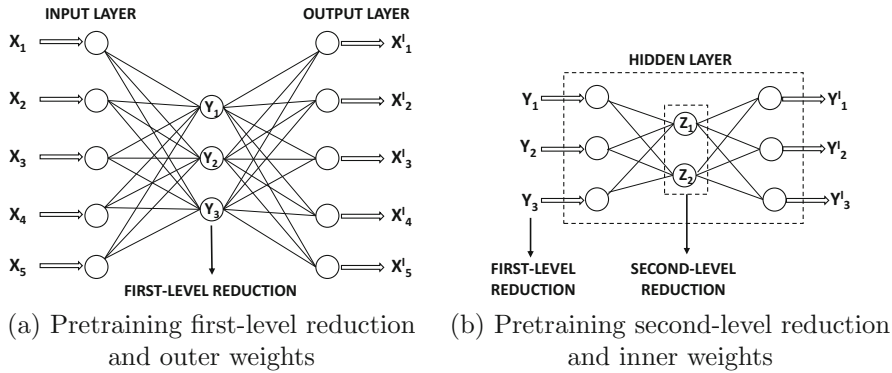


Figure 3.13: Pretraining the neural network of Figure 3.11.

middle hidden layer is missing and the two symmetric outer hidden layers are collapsed into a single hidden layer. The assumption is that the two outer hidden layers are related to one another in a symmetric way like a smaller replicator neural network. In the second step, the reduced representation in the first step is used to learn the second-level reduced representation (and weights) of the inner hidden layers. Therefore, the inner portion of the neural network is treated as a smaller replicator neural network in its own right. Since each of these pretrained subnetworks is much smaller, the weights can be learned without much overfitting. This initial set of weights is then used to train the entire neural network of Figure 3.11 with backpropagation. Note that this process can be performed in layerwise fashion for a deep neural network containing any number of hidden layers. It is noteworthy that the symmetric autoencoder naturally constructs hierarchically related dimensionality reductions of different levels of compression.

Without pretraining, it is possible for the approach to obtain a trivial reduction, so that the reconstruction always returns the average of the training data [264]. This occurs because of overfitting in the large parameter space of a deep network. From the point of view of optimization of parameters, one can view overfitting as a process of getting stuck in local minima. Pretraining helps in conditioning the parameters towards a basin of more attractive local minima by selecting a good initialization point. These techniques are recent results in deep learning that can only be described as breakthrough advancements [186]. For example, it has been shown in [264] that one can convert a 784 pixel image into just 6 real numbers with deep autoencoders. This is not possible with PCA. Just as deep autoencoders provide better reconstruction than matrix-factorization and PCA [264], such methods are also likely to be more accurate for anomaly detection. A number of different implementations of neural networks in different domains are provided in [147, 160, 249, 552].

### 3.6.3 Practical Issues

There are two problematic issues with the use of neural networks. The first issue is that neural networks are slow to train. There is little that one can do to address this issue because computational complexity is an inherent problem with neural networks. Nevertheless, recent hardware and algorithmic advancements have made neural networks (in general) and deep learning (in particular) more feasible.

The second problem with neural networks is that they are sensitive to noise. In fact, since outliers are treated as normal points during the training phase, there will inevitably

be errors in the model. The problem of treating outliers as normal points will be manifested as overfitting. This problem is particularly significant in the case of multilayer networks. For a sufficiently complex multilayer network, every point in the training data will be able to obtain an outlier score of 0 even when pretraining is used to avoid overfitting. However, out-of-sample points would continue to have more realistic scores because they were not included in the training process. Therefore, it is advisable to use only a random subset of the data to train the model and score the remaining points with the resulting model. The size of this random subset depends on the complexity of the multilayer network used. For more complex networks, larger training data sets are required. Therefore, one should select the complexity of the neural network depending on the amount of available training data. A natural approach is to repeatedly sample points from the training data, create the model, and score the remaining points with the resulting model. The scores of each point over various random samples (in which they are scored) are then averaged in order to provide the final result. One challenge with the use of this approach is that neural networks are slow to train and therefore it is sometimes computationally infeasible to train multiple models. Nevertheless, some recent methods like *dropout training* [510] can be leveraged to simulate ensemble performance in an efficient way without explicitly creating multiple network models.

### 3.6.4 The Broad Potential of Neural Networks

The aforementioned sections showed two different ways of modeling outliers in the neural network setting by using outputs with different interpretations. In practice, there are almost an unlimited number of ways in which one could use neural networks to model outliers. For example, neural networks can be used to capture various unsupervised models such as self-organizing maps [593], mixture modeling and clustering methods by appropriately defining the output nodes and corresponding optimization problem. Since methods like clustering and mixture modeling can be used for outlier detection (cf. Chapters 2 and 4), one can also adapt these neural network methods for scoring data points as outliers. Given the recent advances in deep learning, these directions represent an untapped potential of neural networks. However, one always needs to be careful to design the model in such a way so as to avoid the pervasive problem of overfitting with neural networks.

## 3.7 Limitations of Linear Modeling

---

Regression analysis has a few limitations as a tool for outlier detection. The most significant of these shortcomings was discussed at the very beginning of this chapter, in which the data-specific nature of regression analysis was explored. In particular, the data needs to be highly correlated, and aligned along lower-dimensional subspaces for regression analysis techniques to be effective. When the data is uncorrelated, but highly clustered in certain regions, such methods may not work effectively. In such cases, nonlinear models and kernel methods provide a useful choice. However, such methods are computationally intensive and may often result in overfitting.

Another related issue is that the correlations in the data may not be global in nature. A number of recent analytical observations [7] have suggested that the subspace correlations are specific to particular localities of the data. In such cases, the global subspaces found by PCA are suboptimal for outlier analysis. Therefore, it can sometimes be useful to combine linear models with proximity-models (discussed in the next chapter), in order to create

more general local subspace models. This will be the topic of high-dimensional and subspace outlier detection, which is discussed in detail in Chapter 5.

As with any model-based approach, overfitting continues to be an issue, when used with a small set of data records. In this context, the relationship of the number of records to the data dimensionality is important. For example, if the number of data points are less than the dimensionality, it is possible to find one or more directions along which the variance is zero. Even for cases, where the data size is of greater (but similar) magnitude as the data dimensionality, considerable skew in the variances may be observed. This is evident from the results of Figure 3.5(c) and (d), where there is considerable skew in the eigenvalues for a small set of uniformly distributed data. This skew reduces as the data size is increased. This is a classic case of overfitting, and it is important to interpret the results carefully, when the data set sizes are small.

The interpretability of regression-based methods is rather low. These methods project the data into much lower-dimensional subspaces, which are expressed as a linear (positive or negative) combination of the original feature space. This cannot be easily interpreted in terms of physical significance in many real application. This also has the detrimental effect of reducing the intensional knowledge of the user for a particular application. This is undesirable, because it is usually interesting to be able to explain *why* a data point is an outlier in terms of the features of the original data space.

Finally, the computational complexity of the approach may be an issue when the dimensionality of the data is large. When the data has dimensionality of  $d$ , this results in an  $d \times d$  covariance matrix, which may be rather large. Furthermore, the diagonalization of this matrix will slow down at least quadratically with increasing dimensionality. A number of techniques have recently been proposed, which can perform PCA in faster time than quadratic dimensionality [230]. The computational issue is particularly challenging in the case of kernel generalizations of linear methods. These issues can also be ameliorated with ensemble methods [32]. With advances in methods for matrix computation and the increasing power of computer hardware, this issue has ceased to be as much of a problem in recent years. Such dimensionality reduction techniques are now easily applied to large text collections with a dimensionality of several hundreds of thousands of words. In fact, methods like neural networks and deep learning have become increasingly feasible in recent years. If one can overcome the computational challenges associated with these methods, they can often be used to provide robust results.

## 3.8 Conclusions and Summary

---

This chapter presents linear models for outlier detection. Many data sets show significant correlations among the different attributes. In such cases, linear modeling may provide an effective tool for removing the outliers from the underlying data. In most cases, principal component analysis provides the most effective methods for outlier removal, because it is more robust to the presence of a few outliers in the data. Such methods can also be extended to nonlinear models, although the approach is computationally complex and can sometimes overfit the data. Many other mathematical models such as SVMs, matrix factorization and neural networks uses different variations of these concepts. Multilayer neural networks can model complex and nonlinear patterns, especially with the use of deep-learning methods. Other than neural networks, most of these models are global models, which do not recognize the varying subspace and correlation patterns in different data localities. However, it provides a general framework, which can be used for generalized local linear models, which

are discussed in Chapters 4 and 5.

### 3.9 Bibliographic Survey

---

The relationships between the problems of regression and outlier detection has been explored extensively in the literature [467]. Outlier analysis is generally seen as an enormous challenge to robust regression in terms of the *noise* effects, and this has motivated an entire book on the subject. In many cases, the presence of outliers may lead to unstable behavior of regression analysis methods. An example of this was illustrated in Figure 3.3(b) of this chapter, where a single outlier completely changed the regression slope to one that does not reflect the true behavior of the data. It can be shown that under specific conditions, outliers can have an arbitrarily large effect on the estimation of the regression coefficients. This is also referred to as the *breakdown point* [245, 269] of regression analysis. Such circumstances are very undesirable in outlier analysis, because of the likelihood of very misleading results. Subsequently, numerous estimators have been proposed with higher breakdown points [467]. In such cases, a higher level of contamination would need to be present in the data for breakdown to occur.

The method of *principal component analysis* is also used frequently in the classical literature [296] for regression analysis and dimensionality reduction. Its application for noise correction in the text domain was first observed in [425], and then modeled theoretically in [21]. It was shown that the projection of the data points onto the hyperplanes with the greatest variance provides a data representation, with higher quality of similarity computations because of the removal of noise from the data. Latent Semantic Indexing [162, 425], a variant of PCA, was initially proposed in the context of text data for the purpose of reducing dimensionality for retrieval, rather than for noise reduction. However, many years of experience with LSI have revealed that the quality of retrieval actually improved, as was explicitly noted in [425]. Later, this was theoretically modeled for relational data [21]. PCA and LSI are dimensionality reduction techniques that summarize the data by finding linear correlations among the dimensions.

PCA-based techniques have been used in order to detect outliers in a wide variety of domains such as statistics [116], astronomy [177], ecological data [282], network intrusion detection [334, 493, 544], and many kinds of time-series data. Some of the aforementioned applications are temporal, whereas others are not. Because of the relationship between PCA and time series correlation analysis, much of the application of such regression methods has been to the temporal domain. Regression-based methods will be re-visited in Chapter 9, where a number of methods for temporal outlier analysis will be discussed. In the context of temporal data, the outlier analysis problem is closely related to the problem of *time series forecasting*, where deviations from forecasted values in a time series are flagged as outliers. A variety of regression-based methods for noise reduction and anomaly detection in time-series sensor data streams are also discussed in [22]. In addition, a number of methods that resemble structural and temporal versions of PCA have been used for anomaly detection in graphs [280, 519]. In such methods, an augmented form of the adjacency matrix, or the similarity matrix, may be used for eigenvector analysis. Such methods are commonly referred to as *spectral methods*, and are discussed in Chapter 12. Nonlinear dimensionality reduction methods are discussed in [481], and applications to novelty detection is discussed in [270, 541]. The work in [270], however, uses reconstruction error (hard kernel PCA) as the anomaly score, rather than the soft approach [35] discussed in this book. The kernel Mahalanobis method is formally proposed and tested as a distinct method in [35]. The ap-

plication of such nonlinear dimensionality reductions for outlier detection is also discussed in [475]. However, the approach in [475] uses spectral methods instead of global dimensionality reduction to enhance the local nature of the reduction process. This type of approach will be more effective if different portions of the data show different manifold structures.

Another general model beyond global PCA is one in which the data is modeled as a probabilistic mixture of PCAs [549]. This is referred to as *Probabilistic PCA (PPCA)*. Such methods are quite prone to noise in the underlying data during the process of mixture modeling. A method proposed in [161] increases the robustness of PCA by modeling the underlying noise in the form of a Student's  $t$ -distribution. The effects of outliers on PCA-based clustering algorithms are significant. The work in [7] provides a methods for providing the outliers as a side product of the output of the clustering algorithm. Furthermore, methods for using local PCA in outlier analysis will be discussed in detail in Chapter 5 on outlier analysis in high-dimensional data. A recent technique for using dimensionality reduction methods in high-dimensional data is provided in [591]. A complementary approach to dimensionality reduction is the RODS framework in which *sparse coding* is used [178]. Sparse coding methods transform the data to a high-dimensional and sparse representation. Outliers are defined as data points containing dictionary atoms that do not normally occur in other data points are therefore specific to the anomalies. A framework that discovers sparse codings and outliers jointly is discussed in [3].

Kernel support vector machines have been used frequently for novelty detection with the use of a one-class version of the model [480, 384]. A general discussion of support vector machines may be found in [33]. The work in [384] is particularly noteworthy because it provides an interesting evaluation of such models with respect to their performance in different data sets. One-class support-vector machines can be sensitive to the data domain, feature representations, and the choice of the kernel function. The variations and sensitivity in the performance of support-vector machines are discussed in [384, 460]. Other one-class methods for support-vector machine classification are discussed in [52, 113, 303, 384, 460, 538, 539].

In principle, any matrix factorization technique can be used for outlier analysis. An example of an outlier analysis method that uses matrix-factorization is discussed in [576]. The core principle is that dimensionality reduction methods provide an approximate representation of the data along with a corresponding set of residuals. These residuals can be used as the outlier scores. The matrix factorization methodology discussed in this chapter is commonly used in recommender systems [34].

Neural networks are discussed in detail in [84]; the problem of deep learning is discussed in [223]. The application of one-class neural networks to outlier detection is discussed in [268, 388, 389, 529]. A specific application of one-class neural networks to document classification is provided in [385]. Another class of neural networks that is commonly used is the *replicator neural network* [53, 147, 250, 567, 552], which is used to score data points as outliers. A recent implementation of replicator neural networks may be found in [160]. The use of deep-learning autoencoders for anomaly detection is explored in [53].

## 3.10 Exercises

---

1. Consider the data set of the following observations:  $\{(1, 1), (2, 0.99), (3, 2), (4, 0.98), (5, 0.97)\}$ . Perform a regression with  $Y$  as the dependent variable. Then perform a regression with  $X$  as the dependent variable. Why are the regression lines so different? Which point should be removed to make the regression lines more similar?

2. Perform Principal Component Analysis on the data set of Exercise 1. Determine the optimal 1-dimensional hyperplane to represent the data. Which data point is furthest from this 1-dimensional plane?
3. Remove the outlier point found in Exercise 2, and perform PCA on the remaining four points. Now project the outlier point onto the optimal regression plane. What is the value of the corrected point?
4. Suppose that you have survey containing numerical data. You know that participants have occasionally made mistakes in exactly one of the fields, because it is so difficult to fill correctly. Discuss how you would detect such outliers.
5. How would your answer to the previous question change if the participants could have made mistakes in any of the fields and not just a specific field.
6. Download the *KDD CUP 1999 data set* from the UCI Machine Learning Repository [203], and perform PCA on the quantitative attributes. What is the dimensionality of the subspace required to represent (i) 80% of the variance, (ii) 95% of the variance, and (iii) 99% of the variance.
7. Repeat Exercise 6 with the use of the *Arrhythmia* data set from the *UCI Machine Learning Repository* [203].
8. Generate 1000 data points randomly in 100-dimensional space, where each dimension is generated from the uniform distribution in  $(0, 1)$ . Repeat Exercise 6 with this data set. What happens, when you use 1,000,000 data points instead of 1000?
9. Consider a 2-dimensional data set with variables  $X$  and  $Y$ . Suppose that  $Var(X) \ll Var(Y)$ . How does this impact the slope of the  $X$ -on- $Y$  regression line, as compared to the slope of the  $Y$ -on- $X$  regression lines. Does this provide you with any insights about why one of the regression lines in Figure 3.3(b) shifts significantly compared to that in Figure 3.3(a), because of the addition of an outlier?
10. Scale each dimension of the *Arrhythmia* data set, such that the variance of each dimension is 1. Repeat Exercise 7 with the scaled data set. Does the scaling process increase the number of required dimensions, or reduce them? Why? Is there any general inference that you can make about an arbitrary data set from this?
11. Let  $\Sigma$  be the covariance matrix of a data set. Let the  $\Sigma$  be diagonalized as follows:

$$\Sigma = PDP^T$$

Here,  $D$  is a diagonal matrix containing the eigenvalues  $\lambda_i$ , and  $D^{-1}$  is also a diagonal matrix containing the inverse of the eigenvalues (i.e.  $1/\lambda_i$ )

- Show that  $\Sigma^{-1} = PD^{-1}P^T$
- For a given data point  $\bar{X}$  from a data set with mean  $\bar{\mu}$ , show that the value of the Mahalanobis distance  $(\bar{X} - \bar{\mu})\Sigma^{-1}(\bar{X} - \bar{\mu})^T$  between  $\bar{X}$  and the mean  $\bar{\mu}$  reduces to the same expression as the score in Equation 3.17.



---

---

## Chapter 4

# Proximity-Based Outlier Detection

---

---

“To lead the orchestra, you have to turn your back to the crowd.” – Max Lucado

### 4.1 Introduction

---

Proximity-based techniques define a data point as an outlier when its locality (or *proximity*) is sparsely populated. The proximity of a data point may be defined in a variety of ways, which are subtly different from one another but are similar enough to merit unified treatment within a single chapter. The most common ways of defining proximity for outlier analysis are as follows:

- **Cluster-based:** The non-membership of a data point in any of the clusters, its distance from other clusters, the size of the closest cluster, or a combination of these factors are used to quantify the outlier score. The clustering problem has a complementary relationship to the outlier detection problem in which points either belong to clusters or they should be considered outliers.
- **Distance-based:** The distance of a data point to its  $k$ -nearest neighbor (or other variant) is used in order to define proximity. Data points with large  $k$ -nearest neighbor distances are defined as outliers. Distance-based algorithms typically perform the analysis at a much more detailed granularity than the other two methods. On the other hand, this greater granularity often comes at a significant computational cost.
- **Density-based:** The *number* of other points within a specified local region (grid region or distance-based region) of a data point, is used in order to define local density. These local density values may be converted into outlier scores. Other kernel-based methods or statistical methods for density estimation may also be used. The major difference between clustering and density-based methods is that clustering methods partition the data *points*, whereas density-based methods partition the data *space*.

Clearly, all these techniques are closely related because they are based on some notion of *proximity* (or similarity). The major difference is at the detailed level of *how* this proximity

is defined. These different ways of defining outliers may have different advantages and disadvantages. In many cases, the distinctions between these different classes of methods become blurred when the outlier scores are defined using<sup>1</sup> more than one of these concepts. This chapter addresses these issues in a unified way.

One major difference between distance-based and the other two classes of methods lies in the level of *granularity* at which the analysis is performed. In both clustering- and density-based methods, the data is pre-aggregated before outlier analysis by either partitioning the points or the space. The data points are compared to the distributions in this pre-aggregated data for analysis. On the other hand, in distance-based methods, the  $k$ -nearest neighbor distance to the *original data points* (or a similar variant) is computed as the outlier score. Thus, the analysis in nearest-neighbor methods is performed at a more detailed level of granularity than clustering methods. Correspondingly, these methods provide different trade-offs between effectiveness and efficiency for data sets of different sizes. Nearest-neighbor methods may require  $O(N^2)$  time to compute all  $k$ -nearest neighbor distances for a data set with  $N$  records, unless indexing or pruning techniques are used to speed up the computations. However, indexing and pruning techniques generally work well only in some restricted settings such as lower-dimensional data sets. Furthermore, pruning is not designed for outlier score computation, and it can only be used in settings in which the binary labels (indicating whether points are outliers) need to be reported. In spite of these disadvantages, nearest-neighbor methods remain exceedingly popular. This is because such methods can often provide more detailed and accurate analysis, especially for smaller data sets in which robust clustering or density analysis is not possible. Thus, the particular choice of the model depends on the nature of the data and its size.

Proximity-based methods are naturally designed to detect both noise and anomalies, although different methods are suited to these different kinds of outliers. For example, weak definitions of proximal sparsity, such as the non-membership of data points in clusters are naturally designed to detect weak outliers (or noise), whereas large levels of deviation or sparsity in terms of density- or distance-based definitions can also detect strong outliers (or anomalies). These methods are extremely popular because of their intuitive simplicity and interpretability. In fact, a number of methods for intuitive exploration and explanation of outliers [318] are based on proximity-centered definitions. Because of the simplicity of the underlying methods, they can be easily generalized to almost all types of data such as time-series data, sequence data, or graph data.

This chapter is organized as follows. Section 4.2 discusses methods for using clusters in outlier analysis. Section 4.3 discusses distance-based methods for outlier detection. Density-based methods are discussed in section 4.4. The limitations of proximity-based outlier detection are discussed in section 4.5. Section 4.6 presents the conclusions and summary.

## 4.2 Clusters and Outliers: The Complementary Relationship

---

A well-known complementary relationship exists between clustering and outlier detection. A simplistic view would be that every data point is either a member of a cluster or an outlier. In clustering, the goal is to partition the points into dense subsets, whereas in outlier detection, the goal is to identify points that do not seem to fit naturally in these

---

<sup>1</sup>It will be discussed later in this chapter, that the well-known LOF method [96] can be interpreted either as a distance-based or density-based method, depending on how it is presented.

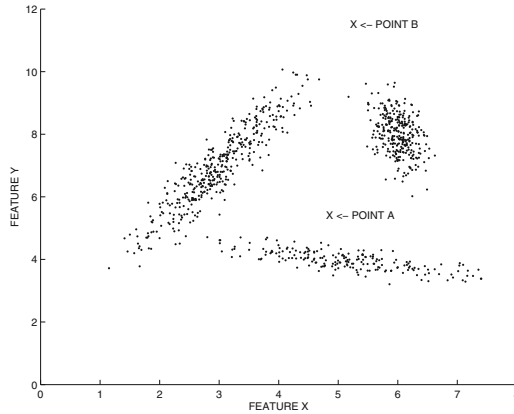


Figure 4.1: The example of Figure 2.9 revisited: Proper distance computations can detect better outliers

dense subsets. In fact, most clustering algorithms report outliers as a side-product of their analysis.

However, it is important to understand that using only the complementary relationship of points to clusters in order to define outliers results in the discovery of *weak* outliers or noise. This is because non-membership of data points in clusters is a rather blunt hammer to measure the *level of* deviation of a data point from the normal patterns. For example, a data point that is located at the fringes of a large cluster is very different from one that is completely isolated from all the other clusters. Furthermore, *all* data points in very small clusters may sometimes also be considered outliers. Therefore, when using clustering for outlier detection, a more nuanced approach (than that of cluster non-membership) is used for computing the outlier scores.

A simple definition of the outlier score may be constructed by using the distances of data points to cluster centroids. Specifically, the distance of a data point to its closest cluster centroid may be used as a proxy for the outlier score of a data point. Since clusters may be of different shapes and orientations, an excellent distance measure to use is the *Mahalanobis distance*, which scales the distance values by local cluster variances along the directions of correlation. Consider a data set containing  $k$  clusters. Assume that the  $r$ th cluster in  $d$ -dimensional space has a corresponding  $d$ -dimensional row vector  $\overline{\mu}_r$  of attribute-wise means, and a  $d \times d$  co-variance matrix  $\Sigma_r$ . The  $(i, j)$ th entry of this matrix is the local covariance between the dimensions  $i$  and  $j$  in that cluster. Then, the squared Mahalanobis distance  $\mathcal{MB}(\overline{X}, \overline{\mu}_r, \Sigma_r)^2$  between a data point  $\overline{X}$  (expressed as row vector) and the cluster distribution with centroid  $\overline{\mu}_r$  and covariance matrix  $\Sigma_r$  is defined as follows:

$$\mathcal{MB}(\overline{X}, \overline{\mu}_r, \Sigma_r)^2 = (\overline{X} - \overline{\mu}_r)\Sigma_r^{-1}(\overline{X} - \overline{\mu}_r)^T \tag{4.1}$$

After the data points have been scored with the local Mahalanobis distance, any form of extreme-value analysis can be applied to these scores to convert them to binary labels.

One can also view the Mahalanobis distance as an adjusted Euclidean distance between a point and the cluster centroid after some transformation and scaling. Specifically, the point and the centroid are transformed into an axis system defined by the principal component directions (of the cluster points). Subsequently, the squared distance between the candidate outlier point and cluster centroid is computed along each of the new axes defined by these

principal components, and then divided by the variance of the cluster points along that component. The sum of these scaled values over all the components provides the squared Mahalanobis distance. The effect of the Mahalanobis distance is to provide statistical normalization based on the characteristics of a particular data locality. Even small distances along directions in which cluster variances are small may be *statistically* significant within that data locality. Similarly, large distances along directions in which cluster variances are large may not be statistically significant within that data locality. Such an approach will yield more refined results than a global use of the Euclidean distance because it is better tailored to the data locality at hand. This is evident from the example illustrated in Figure 4.1, in which the data point ‘A’ is more obviously an outlier than data point ‘B’ because the latter could be (weakly) related to one of the elongated clusters. However, this subtle distinction cannot be detected with the use of the Euclidean distance, according to which the data point ‘A’ is closest to the nearest cluster centroid. It is noteworthy that the use of the Mahalanobis distance achieves similar goals of local normalization as achieved by some other local density-based methods discussed later in this chapter (like LOF and LOCI).

The outlier scoring criterion should always be tied closely to the objective function that is optimized in the clustering algorithm. When the Mahalanobis distance is used for scoring, it should also be used within the clustering process for distance computations. For example, the Mahalanobis  $k$ -means algorithm [33] can be used in the clustering phase. Therefore, in each assignment iteration, data points are assigned to clusters based on their Mahalanobis distance to the various cluster centroids. As a result, the clustering process will be sensitive to the different shapes and orientations of the underlying clusters (as in Figure 4.1). In fact, the EM algorithm discussed in Chapter 2 can be considered a soft version of a Mahalanobis  $k$ -means algorithm [23]. Note that the term in the exponent of the Gaussian distribution for each mixture component of the probabilistic model in Chapter 2 is the (squared) Mahalanobis distance. Furthermore, the fit value computed by the EM algorithm is generally dominated by the exponentiated Mahalanobis distance to the nearest cluster centroid. The Mahalanobis  $k$ -means algorithm converts the soft probabilities into hard assignments. Thus, cluster-based outlier analysis methods are hard avatars of the (soft) probabilistic mixture models introduced in Chapter 2.

In addition to distance-based criteria, it is common to use cluster cardinality as a component of the outlier score. For example, the negative logarithm of the fraction of points in the nearest cluster can be used as a component of the outlier score. One can create two separate  $N$ -dimensional vectors of scores based on the distance and cardinality criteria, standardize each of the vectors to unit variance, and then add them. The cardinality criterion is especially popular in histogram-based methods in which the space is partitioned into regions of roughly equal size. It is noteworthy that histogram-based methods are variants of clustering methods. Incorporation of cluster cardinality into the scores is helpful in distinguishing small groups of clustered outliers from normal points occurring in larger clusters. The identification of clustered anomalies can be achieved even more effectively by using a minimum threshold on the number of data points in each cluster. An example is illustrated in Figure 4.2, where the use of a threshold of 4 is sufficient to identify the three isolated data points. Such outliers are common in real applications, because the same (rare) process might generate these outliers multiple times, albeit a small number of times. In general, clustering methods are much better than histogram-based methods in handling clustered anomalies because they partition the data *points* rather than the data *space* in a more flexible way; they can therefore detect and adjust for these types of small clusters more easily during the partitioning process.

Clustering-based methods naturally have a high variability of prediction depending on

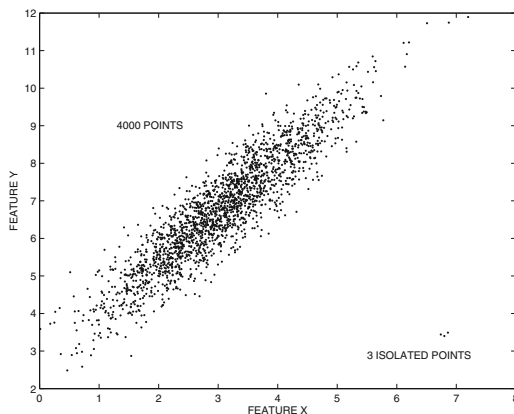


Figure 4.2: The example of Figure 1.5 revisited: Proper combination of global and local analysis in proximity-based methods can identify such outliers

the specific choice of model, randomized initialization, or parameter setting. As discussed in Chapter 6, this type of variability is a theoretical indication of a suboptimal detector *in expectation*. To improve the performance, it is often advisable to use an average of the outlier scores from multiple clusterings (with different parameter settings) to obtain better results. Even in cases where the optimal parameter setting is known, it is helpful to average the (normalized) outlier scores over different runs of a randomized clustering algorithm to obtain good results. A deterministic clustering algorithm can be suitably randomized by running it on samples of the data, using different initialization points, or explicitly randomizing specific steps of the algorithm. In general, sufficient randomization is often more important than the quality of the base clustering method. A particularly useful method in this context is the use of *extremely-randomized clustering forests* [401]. Even though the scores from a single application of clustering are often suboptimal, the use of this type of ensemble approach provides surprisingly good results. More details on such methods are provided in Chapter 6.

One interesting property of clustering ensembles is that the type of outlier discovered will be sensitive to the type of clustering that is used. For example, a subspace clustering will yield a subspace outlier (cf. section 5.2.4 of Chapter 5); a correlation-sensitive clustering will yield correlation sensitive outliers, and a locally-sensitive clustering will yield locally-sensitive outliers. By combining different base clustering methods, diverse types of outliers may be discovered. One can even extend this broad approach to other data types. Clustering ensembles have been used for discovering edge outliers [17] in graph data (see section 12.3.2.3 of Chapter 12).

### 4.2.1 Extensions to Arbitrarily Shaped Clusters

The discussion in the previous section on the use of the (local) Mahalanobis distance shows that the computation of distances to the nearest cluster centroid should be sensitive to the shape of the corresponding cluster. Although the Mahalanobis computation is effective for the case of elliptically shaped (Gaussian) clusters, it is not quite as effective for the case of clusters that are of arbitrary and non-convex shape. Examples of such clusters are illustrated in Figure 4.3. In the case of Figure 4.3(a) (cf. Figure 3.6(a) of Chapter 3), the entire data is arranged into a single, spiral-like, global *manifold*. The case of Figure 4.3(b)

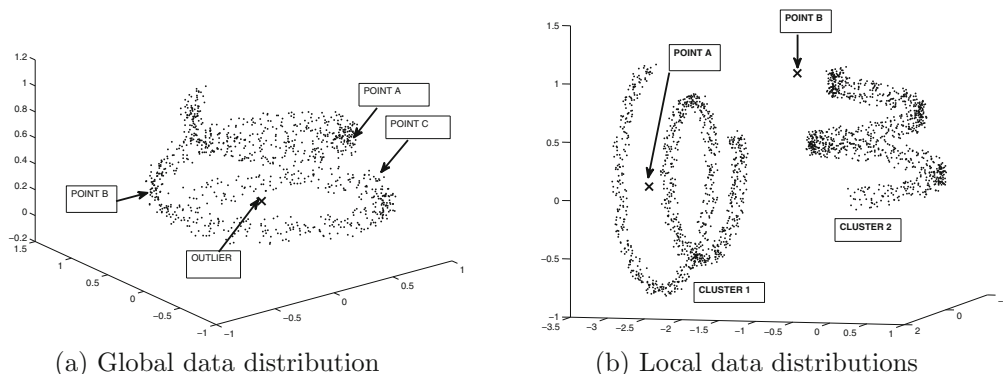


Figure 4.3: Different clusters might have different shapes. The sparsification of similarity matrices is crucial for creating embeddings that respect the varying cluster shapes in different localities.

is even more challenging because different localities of the data contain clusters of different shapes. It is also noteworthy that some of the outliers in Figures 4.3(a) and (b) are actually placed *inside* sparse regions of the non-convex clusters. An example of such an outlier is point ‘A’ of cluster 1 in Figure 4.3(b). The point ‘A’ is much closer to the centroid of cluster 1 than many of the other points belonging to the cluster. Clearly, we need some type of data transformation that can map the points to a new space in which such outliers are exposed.

The case of Figure 4.3(a) is somewhat simpler than 4.3(b), because the entire data set is arranged in a single global distribution in the former but not in the latter. For the case of Figure 4.3(a) it suffices to use methods like nonlinear principal component (or kernel principal component) analysis to map the points to a new space in which Euclidean distances can be used effectively. Such an approach is discussed in section 3.3.8 of Chapter 3. However, such methods require some modifications to adjust for the effect of varying data locality in Figure 4.3(b). Before reading further, the reader is advised to revisit the nonlinear PCA approach discussed in section 3.3.8 of Chapter 3.

As in the case of nonlinear PCA, we can use the largest eigenvectors of an  $N \times N$  kernel similarity matrix  $S = [s_{ij}]$  to embed the data into a multidimensional space in which Euclidean distance functions can be used effectively for clustering. The  $(i, j)$ th entry of  $S$  is equal to the kernel similarity between the  $i$ th and  $j$ th data points. In this sense, the kernel similarity functions described in section 3.3.8.1 of Chapter 3 provide a good starting point. The Gaussian kernel is typically preferred and the bandwidth is set to the median distance between sampled pairs of data points. However, these methods are best suited to discovering *global* embeddings of the data for cases like Figure 4.3(a) in which the entire data set is arranged in a single distribution. There is, therefore, a crucial modification in the construction of these similarity matrices in order to handle the varying distributions of the data in different localities (as in Figure 4.3(b)). These modifications, which are derived from the notion of *spectral clustering* [378], are those of similarity matrix *sparsification* and *local normalization* [297, 378]:

- **Sparsification:** In this case, we retain the computed similarities in  $S$  for the entry  $(i, j)$ , if  $i$  is among the  $k$ -nearest neighbors of  $j$ , or if  $j$  is among the  $k$ -nearest neighbors of  $i$ . Otherwise such entries of  $S$  are set to 0. This step helps in reducing the similarity

between points from different clusters, albeit in a noisy way. Therefore, it helps in creating embeddings in which different clusters dominate different dimensions of the embedding. After performing this step, the matrix  $S$  becomes sparse, because most values of  $s_{ij}$  are 0s.

- **Local normalization:** This step is useful for performing a type of local normalization that helps<sup>2</sup> in adjusting to the varying density of different local regions. This step is, however, optional. Let  $\rho_i$  be the sum of the similarities in the  $i$ th row of  $S$ . Note that  $\rho_i$  represents a kind of local density near the  $i$ th data point because it is defined by the sum of its similarities to its neighbors. Each similarity value  $s_{ij}$  is then divided by the geometric mean of  $\rho_i$  and  $\rho_j$  [297]. In other words, we set  $s_{ij} \leftarrow s_{ij} / \sqrt{\rho_i \cdot \rho_j}$ .

Subsequently, the top- $m$  eigenvectors of the matrix  $S$  are extracted and are stacked in the columns of an  $N \times m$  matrix  $D'$ . Typically, the value of  $m$  is quite small such as 10 to 15, although it might be data set dependent. Each column of the matrix  $D'$  is scaled to unit norm. This matrix provides the  $m$ -dimensional representation of the  $N$  data points. The value of  $m$  should be roughly set to the number of clusters that are extracted. Subsequently, the data in  $D'$  is clustered into  $m$  different clusters using the  $k$ -means algorithm on this new representation. All points are assigned to their closest cluster even if they seem like outliers. By clustering the data in the transformed representation, clusters of arbitrary shape can be discovered.

The distance of each point to its closest centroid is reported as the outlier score. However, while computing the outlier scores, it is important to use a larger number of eigenvectors than  $m$  because outliers are often emphasized along the small eigenvectors. Clustering can be performed using only a small number of eigenvectors but anomalies are often (but not always) hidden along the smaller eigenvectors. Therefore, all non-zero eigenvectors of  $S$  are extracted (while throwing away the nonzero eigenvalues caused by obvious numerical errors). This results in an  $N \times n$  representation  $D_n$ , where  $n \gg m$ . The rows of matrix  $D_n$  are partitioned into the points belonging to the various clusters, which were discovered using the  $m$ -dimensional representation. Let the data matrices containing the  $n$ -dimensional representations of these clusters be denoted by  $D_n^{(1)} \dots D_n^{(c)}$ . The columns of each  $D_n^{(j)}$  are scaled<sup>3</sup> to unit norm in order to compute the local Mahalanobis distance in the *embedded* space. The  $n$ -dimensional representations of the aforementioned cluster centroids are constructed by computing the means of the  $n$ -dimensional points in each cluster  $D_n^{(j)}$  (even though the clustering itself was performed in  $m$ -dimensional space). The squared Euclidean distance of each point to its cluster centroid in this  $n$ -dimensional space is reported as the outlier score.

One problem in the use of such methods is that some of the entries of the similarity matrix  $S$  are noisy. For example, an entry with high similarity between points from different clusters is noisy. This is possible in the original representation because it is hard to compute similarities accurately with kernel similarity functions that are dependent on the Euclidean distances in the original space (like the Gaussian kernel). Even a small number of such noisy entries can seriously hurt the spectral embedding. An approach is discussed in [475] for iteratively clustering the points and using the clusters to correct the noisy entries in  $S$ . This process is repeated to convergence. However, the algorithm in [475] uses a  $k$ -nearest neighbor method for scoring the points rather than the cluster-centroid distance.

<sup>2</sup>Although we do not describe the rationale for this local normalization in detail, it is similar to that described in section 4.4 for the Local Outlier Factor (LOF) algorithm.

<sup>3</sup>Some of the columns may again need to be removed, as they might have zero (local) variance. Therefore, some clusters might have less than  $n$  dimensions.

Robustness can be obtained by using cluster ensemble methods in which the number of clusters, number of eigenvectors, sparsity level and kernel bandwidth are varied within a modest range over different executions of the basic method. The final outlier score of a point is obtained by using the average score over different ensemble components. One can also construct the clusters over samples of data points to improve diversity and the efficiency of the clustering process.

#### 4.2.1.1 Application to Arbitrary Data Types

An important advantage of such spectral methods is that they can be applied to arbitrary data types as long as a similarity function can be defined between the objects. One can use any of the available kernel similarity methods in the literature that have been defined for different data types such as time series, strings, and graphs [33]. Once the similarity function has been defined, one can create a *sparsified* similarity matrix by removing the edges with low weight. The eigenvectors of this matrix are used to create an embedding and perform the clustering. For each data point, its nearest distance to the closest cluster centroid is used to define its outlier score.

#### 4.2.2 Advantages and Disadvantages of Clustering Methods

An important advantage of clustering methods is that they are relatively fast compared to the (more popular) distance-based methods. Distance-based methods have a running time that is quadratic in data dimensionality. On the other hand, many fast clustering algorithms exist in various data domains. One can also use the spectral methods discussed earlier in this section to discover outliers embedded near arbitrarily shaped clusters, or for arbitrary data types by defining appropriate similarity functions.

The main disadvantage of clustering methods is that they might not always provide insights at the required level of detail in smaller data sets. The granularity of outlier analysis methods is generally better when using distance computations directly with respect to the original data points, rather than with respect to aggregated representatives such as cluster centroids. Therefore, clustering methods are most effective when the number of available data points is sufficiently large; in such cases, these methods also have efficiency advantages. Another issue with clustering methods is that the scores have a high level of variability between different randomized executions or parameter choices. Therefore, averaging the (standardized) vector of outlier scores from different executions is essential to obtain robust results.

### 4.3 Distance-Based Outlier Analysis

---

Distance-based methods are a popular class of outlier-detection algorithms across a wide variety of data domains, and define outlier scores on the basis of *nearest neighbor distances*. The simplest example is the case in which the  $k$ -nearest neighbor distance of a point is reported as its outlier score. Because of the simplicity of this definition, it is often easy to generalize this technique to other data types. While this chapter focuses on multidimensional numerical data, such methods have been generalized to almost all other domains such as categorical data, text data, time-series data, and sequence data. The later chapters of this book will present distance-based methods for those cases.

Distance-based methods work with the natural assumption that the  $k$ -nearest neighbor distances of outlier data points are much larger than those of normal data points. A major



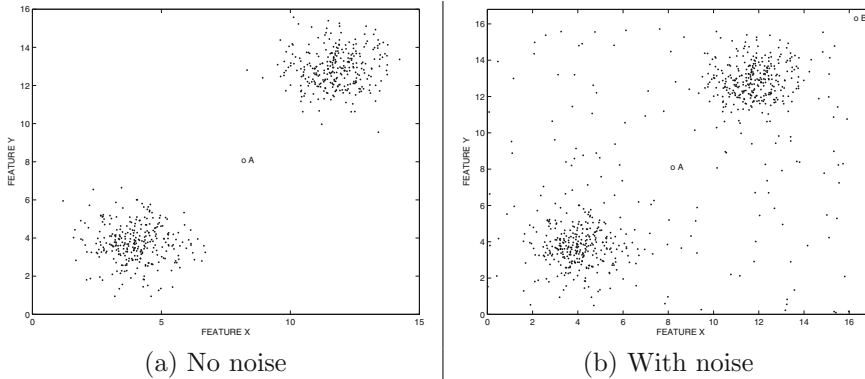


Figure 4.4: The example of Figure 1.1 re-visited: Nearest neighbor algorithms may be more effective than clustering-based algorithms in noisy scenarios because of better granularity of analysis

difference between clustering and distance-based methods is in the *granularity* of the analytical process. Distance-based methods generally have a higher granularity of analysis as compared to clustering-based methods. This property of distance-based methods can enable a more refined ability to distinguish between weak and strong outliers in noisy data sets. For example, in the case of Figure 4.4, a clustering-based algorithm will not be able to distinguish between noise and anomalies easily. This is because the distance to the nearest cluster centroid for the data point ‘A’ will remain the same in Figures 4.4(a) and (b). On the other hand, a  $k$ -nearest neighbor algorithm will distinguish between these situations because the noisy data points will be included among the distance evaluations. On the other hand, the clustering approach will not be able to distinguish between these situations quite as well because the cluster centroids are relatively insensitive to the noise in the underlying data. Of course, it is also possible to modify cluster-based methods to include the effects of noise. In those cases, the two approaches converge to very similar schemes. This is because the two types of methods are closely related.

Distance-based methods are also able to identify isolated clusters of closely related outliers. For example, in order to identify a small (anomalous) cluster containing  $k_0$  data points, one needs to use a value of  $k \geq k_0$  in the  $k$ -nearest neighbor algorithm. Although such anomalies can also be identified by clustering methods by setting a threshold on the number of points in each cluster, such points may sometimes enter clusters and bias the corresponding cluster centroids. This can affect the outlier scoring process in a detrimental way.

The most general output of distance-based methods is in the form of scores. However, if the outlier score of each data point is required, then the (vanilla version of the) algorithm requires operations *exactly* proportional to  $N^2$ . In the binary decision version of identifying *whether* a data point is an outlier, it is possible to use various types of pruning and indexing structures to substantially speed up the approach. In the following, we will discuss algorithms for both types of outputs.

### 4.3.1 Scoring Outputs for Distance-Based Methods

The distance-based outlier score of a point is based on its  $k$ th nearest-neighbor distance to the remaining data set. There are two simple variations of this scoring mechanism cor-

responding to the exact  $k$ -nearest neighbor and the average  $k$ -nearest neighbor detectors. Most of the earliest methods for outlier detection focused on the use of the exact  $k$ -nearest neighbor detector.

**Definition 4.3.1 (Exact  $k$ -Nearest Neighbor Score)** *In a database  $\mathcal{D} = \overline{X_1} \dots \overline{X_N}$ , the outlier score of any data point  $\overline{X_i}$  is equal to its  $k$ th nearest neighbor distance to points in  $\mathcal{D} - \{\overline{X_i}\}$ .*

Note that the scored point  $\overline{X_i}$  is itself not included among the  $k$ -nearest neighbors in order to avoid overfitting. For example, if we used  $k = 1$  and allowed the inclusion of the candidate point among the 1-nearest neighbors, every point would be its own nearest neighbor and the outlier scores of all points would be 0. The exclusion of the candidate point from among the neighbors avoids this situation.

The main problem with this definition is that it is difficult to know the “correct” value of  $k$  for any particular data point. In unsupervised problems like outlier detection, there is often no way of parameter tuning with methods like cross-validation, because such methods require knowledge of the ground-truth. An alternative [58] that is more robust to varying choices of  $k$  is the *average  $k$ -nearest neighbor detector*, which is also referred to as the *weighted  $k$ -nearest neighbor detector*.

**Definition 4.3.2 (Average  $k$ -Nearest Neighbor Score)** *Consider a database  $\mathcal{D} = \overline{X_1} \dots \overline{X_N}$ . The outlier score of any data point  $\overline{X_i}$  is equal to its average distance to its  $k$  nearest neighbors in  $\mathcal{D} - \{\overline{X_i}\}$ .*

In general, if we know the “correct” value of  $k$ , the exact  $k$ -nearest neighbor tends to give better results than that given by the best value of  $k$  for the average  $k$ -nearest neighbor detector. However, in unsupervised problems like outlier detection, it is impossible to know the correct value of  $k$  for any particular algorithm, and an analyst might use a range of values of  $k$ . For example an analyst might test the algorithm with equally spaced values of  $k$  in  $[1, N/10]$ . In such cases, the average  $k$ -nearest neighbor method is less sensitive to different choices of  $k$ , because it effectively averages the exact  $k$ -nearest neighbor scores over a range of different values of  $k$ . A related approach, which is rarely employed, is the use of the *harmonic average* instead of the arithmetic average.

**Definition 4.3.3 (Harmonic  $k$ -Nearest Neighbor Score)** *Consider a database  $\mathcal{D} = \overline{X_1} \dots \overline{X_N}$ . The outlier score of any data point  $\overline{X_i}$  is equal to the harmonic mean of its distances to its  $k$  nearest neighbors in  $\mathcal{D} - \{\overline{X_i}\}$ .*

Care must be taken to remove repeated points from the data set in order to use this approach robustly, because the harmonic mean of any set of numbers containing 0 is always 0. Harmonic averages are always dominated by smaller distances (as compared to arithmetic averages), and therefore using a large value of the parameter  $k$  is advisable for greater robustness. In fact, one can set  $k = N$  in the case of harmonic averaging and still obtain high-quality scores. For example, if we set  $k = N$  in an (arithmetically) averaged  $k$ -nearest neighbor detector, then only multivariate extreme values will be discovered, and isolated central points might be ignored. On the other hand, the harmonically averaged scores at  $k = N$  will also be able to discover isolated central points in the data, especially for large data sets. The reason for this behavior of the harmonic average is rooted in its connection to density-based methods (cf. section 4.4.4.1). The main advantage with the harmonic average is that we now have a *parameter-free* detector by setting  $k = N$ , and the results are still of high quality. Although harmonic nearest neighbor detectors remain unexplored in the literature, their potential is significant.

Computing the scores of all the data points is generally computationally intensive. All pairs of distances between data points need to be computed. Therefore, exactly  $O(N^2)$  operations are required. This can be very expensive when the number of data points is large. For example, even for data sets containing a few hundred thousand points, the approach may not return results in a reasonable amount of time. Although it is often claimed that one can use indexing structures for efficiently finding  $k$ -nearest neighbors in  $O(N \cdot \log(N))$  time, the approach is useful for only low-dimensional data sets (i.e., dimensionality less than 10). This is because index structures are not very effective at pruning in the high-dimensional case. Furthermore, such index structures have large constant overheads, which further hurts performance.

A more effective approach is to pre-select a sample of the data points. All  $N$  data points are scored with respect to this sample after excluding the candidate point from the sample (if needed). The results can be averaged over various samples in order to improve the quality of the results with the ensemble [35]. This is especially the case for the less stable variants such as harmonic averaging.

### 4.3.2 Binary Outputs for Distance-Based Methods

Although score-based outputs are more general than binary outputs, they have limited practical application beyond the binary problem of discovering which points are outliers. The advantage of using binary outputs is that it is possible to prune many of the  $O(N^2)$  computations. Therefore, only the top-scored points are reported as outliers, and we do not care about the scores of non-outlier points. This can be achieved either by specifying a minimum threshold on the nearest-neighbor distance [317] (score) or by using a maximum threshold on the *rank* of the  $k$ -nearest neighbor distance [456]. The former parametrization presents a challenge to the analyst in selecting<sup>4</sup> a (possibly nonintuitive) value of an absolute distance threshold up front. The original threshold-based definition of distance-based outliers [317] was based on parameterizing it with fraction  $f$  and distance-threshold  $\beta$ :

**Definition 4.3.4 (Score Threshold-Based Distance Outliers)** *An object  $O$  in a data set  $\mathcal{D}$  is a  $DB(f, \beta)$  outlier, if at least fraction  $f$  of the objects in  $\mathcal{D}$  lies greater than distance  $\beta$  from  $O$ .*

Note that score-based algorithms have a single parameter  $k$  corresponding to the  $k$ th nearest neighbor, whereas binary-thresholding algorithms have two parameters  $f$  and  $\beta$ . The parameter  $f$  is virtually equivalent to using a parameter like  $k$  in the original definition. Instead of using a fraction  $f$ , we can use the exact  $k$ th nearest neighbor distance by setting  $k = \lceil N(1 - f) \rceil$ . For uniformity in discussion throughout this chapter, we restate this definition in terms of the  $k$ th nearest-neighbor distance:

**Definition 4.3.5 (Score Threshold-Based Distance Outliers)** *An object in a data set  $\mathcal{D}$  is an outlier, if its exact  $k$ th-nearest neighbor distance is at least  $\beta$ .*

A second definition [456] is based on top- $r$  thresholding rather than the thresholding of the absolute values of the scores. Therefore, the points are *ranked* in decreasing order of the  $k$ -nearest neighbor distance. The top- $r$  such data points are reported as outliers. Therefore, the threshold is on the distance *rank* rather than the distance *value*.

---

<sup>4</sup>It is possible to compute the outlier scores of a sample of the data points and set the estimate based on the mean and standard deviation of these scores.

**Definition 4.3.6 (Rank Threshold-Based Distance Outliers)** *An object in a data set  $\mathcal{D}$  is an outlier, if its exact  $k$ th-nearest neighbor distance is among the top- $r$  such values in the data set.*

The two definitions are virtually identical, except for the parameter choice presented to the user. In fact, for every choice of distance threshold  $\beta$ , an appropriate value of  $r$  can be selected in order to yield the same result in the two cases.

Note that the most straightforward approach for solving this problem is to first compute all pairwise  $k$ -nearest neighbor distances with a nested loop approach. Subsequently, the appropriate thresholding criterion can be applied to report the relevant points as outliers. However, this naive approach is computationally inefficient. After all, the main advantage of computing binary outputs over scoring outputs is that we can couple the outlier detection process with a pruning methodology to make the approach more efficient.

In all the aforementioned definitions, we have used the exact  $k$ -nearest neighbor distance because of the preponderance of this definition in the literature. However, all the aforementioned definitions and some of the associated pruning methods can be generalized to the average  $k$ -nearest neighbor distance. In the following, we will discuss various pruning methods for the exact  $k$ -nearest neighbor distance, and also investigate their generalizations to the average  $k$ -nearest neighbor distance.

### 4.3.2.1 Cell-Based Pruning

The *cell-based* technique [317] is based on the score-wise thresholding of Definition 4.3.5. The method is designed for the exact  $k$ -nearest neighbor distance, and it cannot be easily generalized to the average  $k$ -nearest neighbor distance. In the cell-based technique, the data space is divided into cells, the width of which is a function of the distance threshold  $\beta$  and the data dimensionality  $d$ . Specifically, each dimension is divided into cells of width at most  $\frac{\beta}{(2 \cdot \sqrt{d})}$ . This odd value of the width is chosen to force certain distance-centric properties of the points in the cells, which are exploited for pruning and efficient processing. The approach is best explained in the 2-dimensional case. Consider the 2-dimensional case, in which successive grid-points are at a distance of at most  $\beta/(2 \cdot \sqrt{2})$ . An important point to be kept in mind is that the number of grid-cells is based on a partitioning of the data *space*, and is independent of the number of data points. This is an important factor in the efficiency of the approach for low dimensional data, in which the number of grid-cells is likely to be modest. On the other hand, this approach is not suited to data of higher dimensionality.

For a given cell, its  $L_1$  neighbors are the cells reached by crossing a single cell-to-cell boundary. Note that two cells touching at a corner are also  $L_1$  neighbors. The  $L_2$  neighbors are the cells obtained by crossing either 2 or 3 boundaries. A particular cell marked  $X$ , along with its set of  $L_1$  and  $L_2$  neighbors, is illustrated in Figure 4.5. It is evident that an interior cell has 8  $L_1$  neighbors and 40  $L_2$ -neighbors. Then, the following properties can be immediately observed.

1. The distance between a pair of points in a cell is at most  $\beta/2$ .
2. The distance between a point and a point in its  $L_1$  neighbor is at most  $\beta$ .
3. The distance between a point and a point in its  $L_r$  neighbor (where  $r > 2$ ) is at least  $\beta$ .

The only cells for which immediate conclusions cannot be drawn, are those in  $L_2$ . This represents the region of uncertainty for the data points in a particular cell. All the distance computations in the approach are performed between pairs of points in this region

	L2	L2	L2	L2	L2	L2	L2
	L2	L2	L2	L2	L2	L2	L2
	L2	L2	L1	L1	L1	L2	L2
	L2	L2	L1	★X	L1	L2	L2
	L2	L2	L1	L1	L1	L2	L2
	L2	L2	L2	L2	L2	L2	L2
	L2	L2	L2	L2	L2	L2	L2

Figure 4.5: Cell-based partitions of data space

of uncertainty. However, further pruning gains are possible with a number of rules that are able to efficiently identify some of the points as outliers or non-outliers without having to materialize all these distance computations. These are as follows:

1. If more than  $k$  data points are contained in a cell *together with its  $L_1$  neighbors*, then *none* of these data points are outliers.
2. If no more than  $k$  data points are contained in a cell ‘A’ and its  $L_1$  and  $L_2$  neighbors, then *all* points in cell ‘A’ are outliers.

The approach uses these various properties and rules to label points as outliers or non-outliers in an efficient way. The first step is to directly label all points in cells containing more than  $k$  points as non-outliers because of the first rule. Furthermore, all neighbor cells of such cells exclusively contain non-outliers. In order to obtain the full pruning power of the first rule, the sum of the points in each cell and its  $L_1$  neighbors is computed. If the total number is greater than  $k$ , then all these points are labeled as non-outliers as well.

Next, the pruning power of the second rule is leveraged. For each cell ‘A’ containing at least one data point, the sum of the number of points in it, and the numbers in its  $L_1$  and  $L_2$  neighbors is computed. If this number is no more than  $k$ , then all points in cell ‘A’ are labeled as outliers. At this point, many cells may have been labeled as outliers or non-outliers. This provides major pruning gains.

The data points in cells that have not been labeled as either outlier or non-outlier need to have their  $k$ -nearest neighbor distance computed explicitly. Even for such data points, the computation of the  $k$ -nearest neighbor distances can be made faster with the use of the cell structure. Consider a cell ‘A’ which has not been labeled as a pure outlier or pure non-outlier cell so far. Such cells may possibly contain a mixture of outliers and non-outliers. The main region of uncertainty for the data points in cell ‘A’ are the set of points in the  $L_2$ -neighbors of this cell ‘A’. It cannot be known whether the points in the  $L_2$  neighbors of ‘A’ are within the threshold distance of  $\beta$  for the points in cell ‘A’. Explicit distance computations are required in order to determine the number of points within the threshold  $\beta$  for the data points in cell ‘A’. Those data points for which no more than  $k$  points in  $L_1$  and  $L_2$  have distance less than  $\beta$  are declared outliers. Note that distance computations need to be explicitly performed only from points in cell ‘A’ to the points in the  $L_2$  neighbors

of cell ‘A.’ This is because all points in  $L_1$  neighbors are already known to be at a distance less than  $\beta$  from any point in ‘A’, and all points in  $L_r$  for  $r > 2$  are already known to be at least a distance of  $\beta$  from any point in ‘A’. Therefore, an additional level of savings is achieved in the distance computations.

The aforementioned description is for the 2-dimensional case. The approach can also be extended to higher dimensions. The main difference for the  $d$ -dimensional case is in terms of the width of a cell (which is now  $\beta/(2 \cdot \sqrt{d})$ ) and the definition of  $L_2$ -neighbors. In the case of 2-dimensional data, the  $L_2$ -neighbors of a cell are defined as is non-neighbor cells that are at most three cells away. In the general case of higher dimensions,  $L_2$  is defined as the set of cells that are at most  $\lceil 2 \cdot \sqrt{d} \rceil$  cells away but not immediate neighbors. All other steps of the algorithm remain identical. However, for the high-dimensional case, this approach becomes increasingly expensive because the number of cells increases exponentially with data dimensionality. Thus, this approach is generally suited to low-dimensional data.

In many cases, the data sets may not be available in main memory, but may be stored on disk. The data-access efficiency therefore becomes a concern. It has been shown in [317] how this approach can be applied to disk-resident data sets with the use of clustered page reads. This algorithm requires at most three passes over the data. More details are available in [317].

#### 4.3.2.2 Sampling-Based Pruning

Sampling methods are extremely flexible, in that they can handle the score-based thresholding (Definition 4.3.5) or the rank-based thresholding (Definition 4.3.6). Furthermore, they can be used with either the exact  $k$ -nearest neighbor detector or the average  $k$ -nearest neighbor detector. They are extremely efficient at pruning, and can also be used as robust ensemble methods [32] (cf. Chapter 6). For these reasons, sampling methods should be considered the first line of attack for efficient distance-based outlier detection. In the following discussion, we will use the rank-based thresholding definition together with an exact  $k$ -nearest neighbor detector. However, the generalization of the methodology to any other combination of thresholding and detection is straightforward and therefore omitted.

The first step is to select a sample  $\mathcal{S}$  of size  $s \ll N$  from the data  $\mathcal{D}$ , and compute all pairwise distances between the data points in sample  $\mathcal{S}$  and those in database  $\mathcal{D}$ . There are a total of  $N \cdot s$  such pairs. This process requires  $O(N \cdot s) \ll O(N^2)$  distance computations. Thus, for each of the sampled points in  $\mathcal{S}$ , the  $k$ -nearest neighbor distance is already known exactly. The top  $r$ th ranked outlier in sample  $\mathcal{S}$  is determined, where  $r$  is the number of outliers to be returned. The score of the  $r$ th rank outlier provides a *lower bound*<sup>5</sup>  $L$  on the  $r$ th ranked outlier score over the entire data set  $\mathcal{D}$ . For the data points in  $\mathcal{D} - \mathcal{S}$ , only an *upper bound*  $V^k(\bar{X})$  on the  $k$ -nearest neighbor distance is known. This upper bound is equal to the  $k$ -nearest neighbor distance of each point in  $\mathcal{D} - \mathcal{S}$  to the sample  $\mathcal{S} \subset \mathcal{D}$ . However, if this upper bound  $V^k(\bar{X})$  is no larger than the lower bound  $L$  already determined, then such a data point  $\bar{X} \in \mathcal{D} - \mathcal{S}$  can be excluded from further consideration as a top- $r$  outlier. Typically, this will result in the removal of a large number of outlier candidates from  $\mathcal{D} - \mathcal{S}$  immediately, as long as the underlying data set is clustered well. This is because most of the data points in clusters will be removed, as long as at least one point from each cluster is included in the sample  $\mathcal{S}$ , and at least  $r$  points in  $\mathcal{S}$  are located in somewhat sparse regions. This can often be achieved with modest values of the sample size  $s$  in real-world data sets. After removing these data points from  $\mathcal{D} - \mathcal{S}$ , the remaining set of points is  $\mathcal{R} \subseteq \mathcal{D} - \mathcal{S}$ . The  $k$ -nearest neighbor approach can be applied to a much smaller set of candidates  $\mathcal{R}$ .

<sup>5</sup>Note that higher  $k$ -nearest neighbor distances indicate greater outlieriness.

The top- $r$  ranked outliers in  $\mathcal{R} \cup \mathcal{S}$  are returned as the final output. Depending on the level of pruning already achieved, this can result in a very significant reduction in computational time, especially when  $|\mathcal{R} \cup \mathcal{S}| \ll |\mathcal{D}|$ .

### Early Termination Trick with Nested Loops

The approach discussed in the previous section can be improved even further by speeding up the second phase of computing the  $k$ -nearest neighbor distances of each data point in  $\mathcal{R}$ . The idea is that the computation of the  $k$ -nearest neighbor distance of any data point  $\bar{X} \in \mathcal{R}$  need not be followed through to termination once it has been determined that  $\bar{X}$  cannot possibly be among the top- $r$  outliers. In such cases, the scan of the database  $\mathcal{D}$  for computation of the  $k$ -nearest neighbor of  $\bar{X}$  can be terminated early.

Note that one already has an estimate (upper bound)  $V^k(\bar{X})$  of the  $k$ -nearest neighbor distance of every  $\bar{X} \in \mathcal{R}$ , based on distances to sample  $\mathcal{S}$ . Furthermore, the  $k$ -nearest neighbor distance of the  $r$ th best outlier in  $\mathcal{S}$  provides a lower bound on the “cut-off” required to make it to the top- $r$  outliers. This lower-bound is denoted by  $L$ . This estimate  $V^k(\bar{X})$  of the  $k$ -nearest neighbor distance of  $\bar{X}$  is further tightened (reduced) as the database  $\mathcal{D} - \mathcal{S}$  is scanned, and the distance of  $\bar{X}$  is computed to each point in  $\mathcal{D} - \mathcal{S}$ . Because this running estimate  $V^k(\bar{X})$  is always an upper bound on the true  $k$ -nearest neighbor distance of  $\bar{X}$ , the process of determining the  $k$ -nearest neighbor of  $\bar{X}$  can be terminated as soon as  $V^k(\bar{X})$  falls below the known lower bound  $L$  on the top- $r$  outlier distance. This is referred to as *early termination* and provides significant computational savings. Then, the next data point in  $\mathcal{R}$  can be processed. In cases where early termination is not achieved, the data point  $\bar{X}$  will almost<sup>6</sup> always be among the top- $r$  (current) outliers. Therefore, in this case, the lower bound  $L$  can be tightened (increased) as well, to the new  $r$ th best outlier score. This will result in even better pruning when the next data point from  $\mathcal{R}$  is processed to determine its  $k$ -nearest neighbor distance value. To maximize the benefits of pruning, the data points in  $\mathcal{R}$  should not be processed in arbitrary order. Rather, they should be processed in decreasing order of the initially sampled estimate  $V^k(\cdot)$  of the  $k$ -nearest neighbor distances (based on  $\mathcal{S}$ ). This ensures that the outliers in  $\mathcal{R}$  are found early on, and the bound  $L$  is tightened as fast as possible for even better pruning. Furthermore, in the inner loop, the data points  $\bar{Y}$  in  $\mathcal{D} - \mathcal{S}$  can be ordered in the opposite direction, based on *increasing* value of  $V^k(\bar{Y})$ . Doing so ensures that the  $k$ -nearest neighbor distances are updated as fast as possible, and the advantage of early termination is maximized. The nested loop approach can also be implemented without the first phase<sup>7</sup> of sampling, but such an approach will not have the advantage of proper ordering of the data points processed. Starting with an initial lower bound  $L$  on the  $r$ th best outlier score obtained from the sampling phase, the nested loop is executed as follows:

---

<sup>6</sup>We say “almost,” because the very last distance computation for  $\bar{X}$  may bring  $V(\bar{X})$  below  $L$ . This scenario is unusual, but might occasionally occur.

<sup>7</sup>Most descriptions in the literature omit the first phase of sampling, which is very important for efficiency maximization. A number of implementations in time-series analysis [311] do order the data points more carefully but not with sampling.

```

for each  $\bar{X} \in \mathcal{R}$  do begin
  for each  $\bar{Y} \in \mathcal{D} - \mathcal{S}$  do begin
    Update current  $k$ -nearest neighbor distance estimate  $V^k(\bar{X})$  by
      computing distance of  $\bar{Y}$  to  $\bar{X}$ ;
    if  $V^k(\bar{X}) \leq L$  then terminate inner loop;
  endfor
  if  $V^k(\bar{X}) > L$  then
    include  $\bar{X}$  in current  $r$  best outliers and update  $L$  to
      the new  $r$ th best outlier score;
endfor

```

Note that the  $k$ -nearest neighbors of a data point  $\bar{X}$  do not include the data point itself. Therefore, care must be taken in the nested loop structure to ignore the trivial cases where  $\bar{X} = \bar{Y}$  while updating  $k$ -nearest neighbor distances.

### 4.3.2.3 Index-Based Pruning

Indexing and clustering are two other common forms of data localization and access. Therefore, it is natural to explore, whether some of the traditional clustering methods or index structures can be used in order to improve the complexity of distance-based computations. The original approach [456] uses the ranking-based thresholding (Definition 4.3.6) with an exact  $k$ -nearest neighbor detector.

As in the case of sampling-based pruning, an upper bound  $V^k(\bar{X})$  of the  $k$ -nearest neighbor distance of candidate data point  $\bar{X} = (x_1 \dots x_d)$  is progressively tightened in conjunction with pruning. The approach uses minimum bounding rectangles of sets of points to estimate distance bounds of the candidate  $\bar{X}$  to any point in the set. These bounds can be used to prune these sets without explicit distance computation in a manner that is discussed later. Let  $R$  be a minimum bounding rectangle, where the lower and upper bounds along the  $i$ th dimension are denoted by  $[r_i, r'_i]$ . Then, the minimum distance  $min_i$  of  $x_i$  along the  $i$ th dimension to any point in the minimum bounding rectangle  $R$  is potentially 0, if  $x_i \in [r_i, r'_i]$ . Otherwise, the minimum distance is  $\min\{|x_i - r_i|, |x_i - r'_i|\}$ . Therefore, by computing this minimum value along each dimension, it is possible to estimate the total minimum bound to the entire rectangle  $R$  by  $\sum_{i=1}^d min_i^2$ . Similarly, the maximum distance  $max_i$  of  $\bar{X}$  along the  $i$ th dimension to the bounding rectangle  $R$  is given by  $\max\{|x_i - r_i|, |x_i - r'_i|\}$ . The corresponding total maximum value can be estimated as  $\sum_{i=1}^d max_i^2$ . The aforementioned bounds can be used in conjunction with index structures such as the  $R^*$ -Tree [78] for estimating the  $k$ -nearest neighbor distance of data points. This is because such index structures use minimum bounding rectangles in order to represent the data at the nodes. In order to determine the outliers in the data set, the points are processed one by one in order to determine their  $k$ -nearest neighbor distances. The highest  $r$  such distances are maintained dynamically over the course of the algorithm. A branch-and-bound pruning technique is used on the index structure in order to determine the value of  $V^k(\bar{X})$  efficiently. When the minimum distance estimate to a bounding rectangle is larger than the value of  $V^k(\bar{X})$ , then the bounding rectangle obviously does not contain any points which would be useful for updating the value of  $V^k(\bar{X})$ . Such subtrees of the  $R^*$ -Tree can be completely pruned from consideration.

Aside from the index-based pruning, individual data points can also be discarded from consideration early. The score (i.e.,  $k$ -nearest neighbor distance) of the  $r$ th best outlier found so far is denoted by  $Dmin$ . Note that  $Dmin$  is exactly the same as the bound  $L$  used in the



previous section on sampling. The estimate of  $V^k(\bar{X})$  for a data point  $\bar{X}$  is monotonically decreasing with algorithm progression, as better nearest neighbors are found. When this estimate falls below  $Dmin$ , the point  $\bar{X}$  can be discarded from consideration.

Many variations of this basic pruning technique have been proposed for different data domains [311]. Typically, such algorithms work with a nested loop structure, in which the outlier scores of candidate data points are computed one by one *in a heuristically ordered outer loop* which approximates a decreasing level of outlier score. For each point, the nearest neighbors are computed in the inner loop in a *heuristic ordering* that approximates increasing distance to the candidate point. The inner loop can be abandoned, when its currently approximated nearest neighbor distance is less than the  $r$ th best outlier found so far ( $V^k(\bar{X}) < Dmin$ ).

A good heuristic ordering in the outer and inner loops can ensure that the data point can be discarded from consideration early. Different tricks can be used to determine this heuristic ordering in various application settings. In many cases, the method for finding the heuristic ordering in the outer loop uses the complementarity of the clustering and outlier detection problem, and orders the data points on the basis of the cardinality of the clusters they are contained in. Data points in clusters containing very few (or one) point(s) are examined first. Typically, a very simple and efficient clustering process is used to create the outer-loop ordering. The method for finding the heuristic ordering in the inner loop typically requires a fast approximation of the  $k$ -nearest neighbor ordering, and is dependent upon the specific data domain or application. An adaptation of this approach for time-series data [311] is discussed in Chapter 9.

### Partition-based Speedup

The approach discussed above may require the reasonably accurate computation of  $V^k(\bar{X})$  for a large number of points, if the bound estimation process discussed above is not sufficiently robust. This can still be expensive in spite of pruning. In practice, the value of  $r$  is quite small, and many data points  $\bar{X}$  can be excluded from consideration without estimating  $V^k(\bar{X})$  explicitly. This is achieved by using clustering [456] in order to perform partitioning of the data space, and then analyzing the data at this level of granularity. A partition-based approach is used to prune away those data points which could not possibly be outliers in a computationally efficient way. This is because the partitioning represents a less granular representation of the data, which can be processed at lower computational costs. For each partition, a lower bound and an upper bound on the  $k$ -nearest neighbor distances of *all* included data points is computed. If the aforementioned upper bound on the  $k$ -nearest neighbor distance estimate of any point in the partition is less than a current value of  $Dmin$ , then the *entire partition* can be pruned from consideration of containing *any* outlier points. The partition-based method also provides a more efficient way for approximating  $Dmin$ . First, the partitions are sorted by decreasing lower bound. The first  $l$  partitions containing at least  $r$  points are determined. The lower bound on the  $l$ -th partition provides an approximation for  $Dmin$ . The upper and lower bound for each partition is computed using the minimum bounding rectangle of the node in the index structure containing the points. More savings may be obtained by using the fact that the distances from each (unpruned) candidate data point  $\bar{X}$  do not need to be computed to data points in partitions that are guaranteed to be further away than the current upper bound on the  $k$ -nearest neighbor distance of the point  $\bar{X}$  (or its containing partition).

Thus, this analysis is performed at a less detailed level of granularity. This makes its efficiency closer to that of clustering-based methods. In fact, the partitions are themselves

generated with the use of a clustering algorithm such as BIRCH [611]. Thus, this approach prunes many data points, and then works with a much smaller set of candidate partitions on which the analysis is performed. This greatly improves the efficiency of the approach. The exact details of computing the bounds on the partitions use the aforementioned estimations on the minimum and maximum distances to the bounding rectangles of different partitions and are discussed in detail in [456]. Because of the close relationship between distance-based and clustering methods, it is natural to use clustering methods to improve the approximations on the  $k$ -nearest neighbor distance. A number of other techniques in the literature use clustering in order to achieve better pruning and speedups in distance-based algorithms [58, 219, 533].

### 4.3.3 Data-Dependent Similarity Measures

The  $k$ -nearest neighbor method can be paired with other types of similarity and distance functions. Unlike the Euclidean distance, data-dependent similarity measures depend not just on the pair of points at hand but also on the statistical distribution of the other points [33]. This can help in incorporating various data-dependent characteristics into the similarity function such as *locality-sensitivity*, *correlation-sensitivity*, or both. An intuitive example of locality-sensitivity is that the same pair of Caucasians would be considered more similar in Asia than in Europe [548].

A well-known locality-sensitive similarity measure is the *shared nearest neighbor similarity measure* [287], in which the intersection cardinality of the  $k$ -nearest neighbor sets of two points is used as the similarity value. The basic idea here is that in a dense region, two points have to be very close<sup>8</sup> in order to have a large number of common nearest neighbors, whereas in a sparse region it is possible for reasonably distant points to have a larger number of common neighbors. The main drawback of the shared nearest-neighbor measure is that we now have two parameters for the number of nearest neighbors, which can potentially be different; one parameter is for the similarity computation and the other is for the distance-based outlier detection algorithm. A larger number of parameters creates uncertainty in unsupervised problems because of difficulty in setting the parameters optimally. The computation of the shared nearest-neighbor measure is also expensive. It is possible to compute this similarity in a robust and efficient way by repeatedly sampling the data, computing the measure, and averaging the measure over different samples. When computing the similarity with sampled data, the shared nearest neighbor distances are computed between all pairs of points (whether they are sampled or not), but only the number of shared neighbors within the sample are counted.

The pairwise Mahalanobis distance, which adapts the Mahalanobis method to compute distances between pairs of points, is a correlation-sensitive distance measure. This measure is equivalent to computing the Euclidean distance between pairs of points after transforming the data to uncorrelated directions with PCA and normalizing each dimension in the transformed data to unit variance. One advantage of this measure is that it uses the correlation structure of the underlying data. Two data points at unit Euclidean distance in the (untransformed) data will have larger Mahalanobis distance if they were aligned along a low-variance direction rather than a high-variance direction in the untransformed data. One can also use the *nonlinear* Mahalanobis method in conjunction with a kernel or spectral similarity matrix to make the distance function sensitive to arbitrary shapes in the

---

<sup>8</sup>As discussed in section 4.4.1, this intuition is directly used by locality-sensitive algorithms like the Local Outlier Factor (LOF) method. Therefore, by appropriately changing the similarity function in the exact  $k$ -nearest neighbor algorithm, one can achieve similar goals as locality-sensitive algorithms like LOF.

underlying data distribution [475].

Random forests can be used to compute data-dependent similarity [99, 491, 555] because of their ability [359] to define data locality in a distribution-dependent way. In the unsupervised setting, the basic idea is to generate synthetic data for the outlier class, and treat the provided data set as the normal class. As discussed in [491], the outliers are generated uniformly at random between the minimum and maximum range of each attribute. A random forest is constructed on the synthetically labeled data. The similarity between a pair of instances (say, A and B) can be defined as either (i) the number of trees in the random forest in which they occur in the same leaf node, or (ii) the average length of the common path traversed by the two instances A and B in each tree. The main advantage of this approach is that it is very efficient to compute the similarity between a pair of instances once the random forest has been constructed.

In principle, any hierarchical representation of the data that preserves its clustering structure can be used to measure similarity with the aforementioned approach as long as the clustering adjusts well to varying local density and correlations. Other unsupervised methods for creating random trees include the use of randomized hierarchical clustering methods [401, 555] and the isolation forest [548]. The former [401] also provides the option to create a “bag-of-words” representation of the data containing the identifiers of the leaf and/or internal tree nodes to which data points are assigned. The computation of hamming distance on this representation is almost equivalent to random-forest similarity measures. Data-dependent similarity measures for categorical data are discussed in section 8.4.2 of Chapter 8.

#### 4.3.4 ODIN: A Reverse Nearest Neighbor Approach

Most of the distance-based methods directly use the  $k$ -nearest neighbor distribution in order to define outliers. A different approach is to use the *number of reverse  $k$ -nearest neighbors* in order to define outliers [248]. Therefore, the concept of a reverse  $k$ -nearest neighbor is defined as follows:

**Definition 4.3.7** *A data point  $p$  is a reverse  $k$ -nearest neighbor of  $q$ , if and only if  $q$  is among the  $k$ -nearest neighbors of  $p$ .*

Data points that have large  $k$ -nearest neighbor distances, will also have few reverse neighbors, because they will lie among the  $k$ -nearest neighbors of very few data points. Thus, an outlier is defined as a point for which the number of reverse  $k$ -nearest neighbors is less than a pre-defined user-threshold.

The reverse nearest neighbor approach can also be easily understood in terms of the underlying  $k$ -nearest neighbor graph. Consider a graph in which the nodes correspond to the data points. A directed edge  $(p, q)$  is added to the graph if and only if  $q$  is among the  $k$ -nearest neighbors of  $p$ . Thus, every node has an outdegree of  $k$  in this graph. However, the in-degree of the nodes may vary, and is equal to the number of reverse  $k$ -nearest neighbors. The nodes with few reverse  $k$ -nearest neighbors are declared outliers. Alternatively, the number of reverse  $k$ -nearest neighbors may be reported as the outlier *score*. Smaller values are more indicative of a greater degree of outlierness. The reverse nearest-neighbor method is also referred to as *Outlier Detection using In-degree Number (ODIN)*. Like the shared nearest neighbor similarity in the previous section, all methods that use the  $k$ -nearest neighbor graph [100, 248] are locality sensitive.

This approach requires the determination of all the  $k$ -nearest neighbors of each node. Furthermore, distance-based pruning is no longer possible since the nearest neighbors of

Table 4.1: Example of Outliers in NHL Player Statistics [318]

Player Name	Short-Handed Goals	Power-Play Goals	Game-Winning Goals	Game-Tying Goals	Games Played
Mario Lemieux	31	8	8	0	70
Jaromir Jagr	20	1	12	1	82
John Leclair	19	0	10	2	82
R. Brind'Amor	4	4	5	4	82

each node need to be determined explicitly. Thus, the approach may potentially require  $O(N^2)$  time for construction of the  $k$ -nearest neighbor graph. The other problem with the approach is that many data points might be ties in terms of the number of reverse nearest neighbors (outlier score). Both these problems can be addressed with ensemble methods [32]. Data points are repeatedly scored with a subsample of  $s$  points, where  $s$  is chosen to be a random value in a constant range. The average score over various samples is reported as the outlier score.

### 4.3.5 Intensional Knowledge of Distance-Based Outliers

An important issue in outlier analysis is to retain a high level of interpretability for providing intuitive explanations and insights. This is very important in many application-driven scenarios. The concept of *intensional knowledge* of distance-based outliers was first proposed in [318]. The idea is to explain the outlier behavior of objects in terms of subsets of attributes. Thus, in this case, a *minimal* bounding box on the subsets of attributes is presented in order to explain the outlier behavior of the data points. For example, consider the case of National Hockey League (NHL) player statistics, which was first presented in [318]. An example set of statistics is illustrated in Table 4.1. The sample output from [318], which explains these outliers is as follows:

<i>Mario Lemieux</i>	An outlier in the 1-d space of power play goals An outlier in the 2-d space of short-handed goals and game-winning goals
<i>R. Brind'Amor</i>	An outlier in the 1-d space of game-tying goals.

Several notions are defined in [318] in order to understand the importance of an outlier:

1. Is a particular set of attributes the minimal set of attributes in which an outlier exists?
2. Is an outlier dominated by other outliers in the data?

The intensional knowledge can be directly characterized in terms of cells, because they define the bounding rectangles along different attributes. The work in [318] proposes a number of roll-up and drill-down methods in order to define the interesting combinations of attributes for intensional knowledge. The concept of *strong* and *weak* outliers is also defined. Outliers that are defined by minimal combinations of attributes are generally considered

stronger from an intensional perspective. It should be emphasized that this definition of strong and weak outliers is specific to an intensional knowledge-based approach and is different from the more general form in which this book uses these terms (as the outlier tendency of an object).

### 4.3.6 Discussion of Distance-Based Methods

Distance-based methods have a number of qualitative advantages over clustering-based techniques because of the more detailed granularity of the analysis. For example, distance-based algorithms can distinguish between noise and anomalies much better than cluster-based techniques. Furthermore, distance-based methods can also find isolated groups of outliers just like clustering methods. On the other hand, clustering methods have the advantage that they can provide insights about the *local* distributions of data points for defining distances. For example, in the case of Figure 4.1, the local cluster structure can be used in order to define a *locally sensitive* Mahalanobis distance, which is much more effective at identifying outliers, than a blind application of the Euclidean metric. However, it is also possible to design a distance-based algorithm based on the Mahalanobis distance, which is also referred to as *instance-specific Mahalanobis method* [33]. These correspond to the data-dependent similarity measures discussed in section 4.3.3.

Although the density-based methods explained later in this chapter do incorporate some notions of locality, they are still unable to provide the detailed level of local insights that an effective combination of a clustering- and distance-based approach can provide. In this context, some recent research has incorporated local clustering insights into distance-based methods [7, 153, 475]. Furthermore, the efficiency advantages of clustering methods should be incorporated into generalized distance-based methods in order to obtain the best results.

## 4.4 Density-Based Outliers

---

Density-based methods use the number of points in specific regions of the space to define outliers. They are very closely related to clustering and distance-based methods. In fact, some of these algorithms can be more easily considered clustering or distance-based methods, depending on how they are presented. This is because the notions of distances, clustering, and density are closely related and inter-dependent. Many of these algorithms discover *locality-sensitive* outliers. This section discusses both local and global algorithms.

The sensitivity of distance-based outliers to data locality was first noticed in [96]. While the Figure 4.1 illustrates the general effect of data locality on both data density and cluster orientation, the work in [96, 97] specifically addresses the issue of varying local density. In order to understand this specific issue, consider the specific example of a data set with varying density in Figure 4.6. The figure contains two outliers labeled ‘A’ and ‘B.’ Furthermore, the figure contains two clusters, one of which is much sparser than the other. It is evident that the outlier ‘A’ cannot be discovered by a distance-based algorithm unless a smaller distance-threshold is used by the algorithm. However, if a smaller distance threshold is used, then many data points in the sparser cluster may be incorrectly declared as outliers. This also means that the *ranking returned by a distance-based algorithm is incorrect when there is significant heterogeneity in the local distributions of the data*. This observation was also noted (more generally) in section 4.2, where it was shown that the outliers in Figure 4.1 are sensitive to both the local cluster density and the cluster orientation (local attribute correlation). However, much of the work in density-based clustering has generally focused

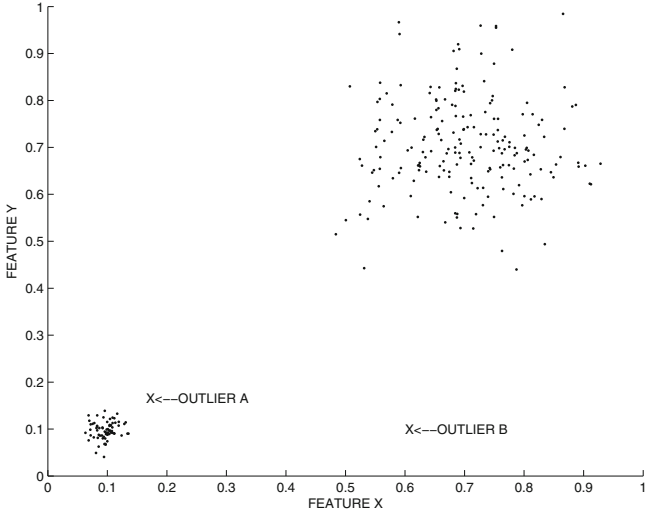


Figure 4.6: Impact of local density on outliers

on issues of varying data density rather than the varying shape and orientation of clusters.

The  $k$ -nearest neighbor algorithm can be paired with the data-dependent similarity measures of section 4.3.3 to address problems caused by local density variations. However, the LOF method [96] uses a different approach by defining the outlier score in a locally-adjusted manner.

**4.4.1 LOF: Local Outlier Factor**

The Local Outlier Factor (LOF) is a quantification of the outlierness of the data points, which is able to adjust for the variations in the different local densities. For a given data point  $\bar{X}$ , let  $D^k(\bar{X})$  be its distance to the  $k$ -nearest neighbor of  $X$ , and let  $L_k(\bar{X})$  be the set of points within the  $k$ -nearest neighbor distance of  $\bar{X}$ . Note that  $L_k(\bar{X})$  will typically contain  $k$  points, but may sometimes contain more than  $k$  points because of ties in the  $k$ -nearest neighbor distance.

Then, the reachability distance  $R_k(\bar{X}, \bar{Y})$  of object  $\bar{X}$  with respect to  $\bar{Y}$  is defined as the maximum of  $dist(\bar{X}, \bar{Y})$  and the  $k$ -nearest neighbor distance of  $\bar{Y}$ :

$$R_k(\bar{X}, \bar{Y}) = \max\{dist(\bar{X}, \bar{Y}), D^k(\bar{Y})\}$$

The reachability distance is not symmetric between  $\bar{X}$  and  $\bar{Y}$ . Intuitively, when  $\bar{Y}$  is in a dense region and the distance between  $\bar{X}$  and  $\bar{Y}$  is large, the reachability distance of  $\bar{X}$  with respect to it is equal to the true distance  $dist(\bar{X}, \bar{Y})$ . On the other hand, when the distances between  $\bar{X}$  and  $\bar{Y}$  are small, then the reachability distance is smoothed out by the  $k$ -nearest neighbor distance of  $\bar{Y}$ . The larger the value of  $k$  is, the greater the smoothing will be. Correspondingly, the reachability distances with respect to different points will also become more similar.

Then, the *average reachability distance*  $AR_k(\bar{X})$  of data point  $\bar{X}$  is defined as the average of its reachability distances to all objects in its neighborhood  $L_k(\bar{X})$  :

$$AR_k(\bar{X}) = \text{MEAN}_{\bar{Y} \in L_k(\bar{X})} R_k(\bar{X}, \bar{Y})$$

Here, the MEAN function simply represents the mean of a set of values. The work in [96] also defines the reachability density as the inverse of this value, though this particular presentation omits this step, since the LOF values can be expressed more simply and intuitively in terms of the average reachability distance  $AR_k(\bar{X})$ . The *Local Outlier Factor* is then simply equal to the mean ratio of  $AR_k(\bar{X})$  to the corresponding values of all points in the  $k$ -neighborhood of  $\bar{X}$ :

$$LOF_k(\bar{X}) = \text{MEAN}_{\bar{Y} \in L_k(\bar{X})} \frac{AR_k(\bar{X})}{AR_k(\bar{Y})} \quad (4.2)$$

$$= AR_k(\bar{X}) \cdot \text{MEAN}_{\bar{Y} \in L_k(\bar{X})} \frac{1}{AR_k(\bar{Y})} \quad (4.3)$$

The use of distance ratios in the definition ensures that the local distance behavior is well accounted for in this definition. As a result, the LOF values for the objects in a cluster are often close to 1, when the data points in the cluster are homogeneously distributed. For example, in the case of Figure 4.6, the LOF values of data points in *both* clusters will be close to 1, even though the densities of the two clusters are different. On the other hand, the LOF values of *both* the outlying points will be much higher since they will be computed in terms of the ratios to the average neighbor reachability distances. The LOF scores can also be viewed as *normalized reachability distance* of a point, where the normalization factor is the *harmonic* mean of the reachability distances in its locality. For example, Equation 4.3 can be rewritten as follows:

$$LOF_k(\bar{X}) = \frac{AR_k(\bar{X})}{\text{HMEAN}_{\bar{Y} \in L_k(\bar{X})} AR_k(\bar{Y})} \quad (4.4)$$

Here, HMEAN denotes the harmonic mean of the reachability distances of all the points in its locality. In principle, we could use any type of mean in the denominator. For example, if we had used a *arithmetic* mean of the reachability distances in the denominator, the results would have a similar interpretation. One observation about the LOF method is that while it is popularly understood in the literature as a density-based approach, it can be more simply understood as a *relative* distance-based approach with smoothing. The relative distances are computed on the basis of the local distribution of reachability distances. The LOF method was originally presented in [96] as a density-based approach because of its ability to adjust to regions of varying density. The density is loosely defined as the inverse of the smoothed reachability distances of a point. This is, of course, not a precise definition of density, which is traditionally defined in terms of the number of data points within a specified area or volume. The presentation in this chapter omits this intermediate density variable, both for simplicity, and for a definition of LOF directly in terms of reachability distances. The real connection of LOF to data density lies in its insightful ability to *adjust* to varying data density with the use of *relative distances*. While this book has also classified this method as a density-based approach, it can be equivalently understood in terms of either a relaxed definition of density or distances. There are a small number of simplifications that one could make to LOF without affecting its performance very significantly:

1. Instead of using the reachability distances, one might use the raw distances.
2. Instead of using the harmonic mean in Equation 4.4, one can simply use the arithmetic mean. Another variant known as LDOF (local distance-based outlier factor) [610] uses the averaged *pairwise-distances* between points in  $L_k(\bar{X})$  instead of the harmonic mean of arithmetic means.

With such modifications, the relationship to distance-based methods is more obvious. It is also possible to use data-dependent similarity measures (cf. section 4.3.3) in combination with an exact  $k$ -nearest neighbor outlier detector to achieve similar goals as LOF.

#### 4.4.1.1 Handling Duplicate Points and Stability Issues

The use of the harmonic mean in Equation 4.4 has some interesting consequences for the stability of the algorithm because the harmonic mean is often not a very stable central representative of a set of values. The occurrence of a single value of 0 in a set of values will result in a harmonic mean of 0. This has consequences in a data set containing duplicate points (or points in dense regions that are very close to one another). For example, if *even one* of the reachability values in the denominator of Equation 4.4 is 0, the entire expression will be set to  $\infty$ . In other words, *all points in the immediate neighborhood of duplicate points are at risk of having their scores set to  $\infty$* . It is clear that such an outcome is not desirable. Note that this problem is reduced with the use of the arithmetic mean instead of the harmonic mean in Equation 4.4. Another possibility is to use  $k$ -distinct-distances, which is essentially equivalent to removing duplicates from the data set for model construction [96]. However, removing duplicates is beneficial only if these duplicates are a result of noise or errors in the data. If the duplicates represent true densities, then the computed LOF score will be biased. Therefore, it is hard to use the solution of  $k$ -distinct-distances without a deeper semantic understanding of the duplicates in the data set. Finally, a reasonable approach is to use regularization by modifying Equation 4.4 with a small value  $\alpha > 0$ :

$$LOF_k(\bar{X}) = \frac{\alpha + AR_k(\bar{X})}{\alpha + \text{HMEAN}_{\bar{Y} \in L_k(\bar{X})} AR_k(\bar{Y})} \quad (4.5)$$

Intuitively, the value of  $\alpha$  regulates the prior probability that a data point is a normal point.

Although the issue of duplicates can be effectively handled using the aforementioned methods, harmonic normalization does result in broader issues of stability with respect to the value of the parameter  $k$ . For example, it is recommended in [96] to use the maximum value of  $LOF_k(\bar{X})$  over a range of different values of  $k$  as the outlier score [96]. However, at small values of  $k$ , groups of points might be close to one another by chance, which will increase the outlier scores of points in their locality. This problem can be viewed as soft version of the duplicate problem discussed earlier. Because of the unstable nature of harmonic normalization, even a *single* tightly knit group can increase the outlier scores of many points in a given locality.

This type of instability tends to make the LOF detector more sensitive to the value of  $k$  than other detectors such as the average  $k$ -nearest neighbor method. Therefore, if one can determine the best value of  $k$  for LOF, it is possible to obtain better results for LOF than those obtained using the best value of  $k$  for other detectors like the exact or average  $k$ -nearest neighbor methods. However, one must be careful in interpreting such results, because it is impossible to know the correct value of  $k$  in a particular detector for unsupervised problems like outlier detection. Furthermore, the good results of (the relatively unstable) LOF at particular values of  $k$  might simply be a manifestation of overfitting. In practice, a better performance measure is to compute various measures of the average performance over a range of parameter choices. In such cases, LOF is often outperformed by simpler detectors such as the exact  $k$ -nearest neighbor detector and the average  $k$ -nearest neighbor detector [32, 35]. Indeed, the trusty old  $k$ -nearest neighbor detectors should never be underestimated.



### 4.4.2 LOCI: Local Correlation Integral

An interesting method proposed in [426] uses a local density-based method for outlier analysis. The LOCI method is truly a density-based method, since it defines the density  $M(\bar{X}, \epsilon)$  of a data point  $\bar{X}$  in terms of the number of data points within a pre-specified radius  $\epsilon$  around a point. This is referred to as the *counting neighborhood* of the data point  $\bar{X}$ . Correspondingly, the average density  $AM(\bar{X}, \epsilon, \delta)$  in the  $\delta$ -neighborhood of  $\bar{X}$  is defined as the mean value of  $M(\bar{X}, \epsilon)$  for all data points at a distance at most  $\delta$  from  $\bar{X}$ . The value of  $\delta$  is also referred to as the *sampling neighborhood* of  $\bar{X}$ , and is always larger than  $\epsilon$ . Furthermore, the value of  $\epsilon$  is always chosen as a constant fraction of  $\delta$ , no matter what value of  $\delta$  is used. The value of  $\delta$  is a critical parameter in the analysis, and multiple values of this parameter are used in order to provide analytical insights at different levels of granularity. The average density in the neighborhood of  $\bar{X}$  is formally defined as follows:

$$AM(\bar{X}, \epsilon, \delta) = \text{MEAN}_{[\bar{Y}: \text{dist}(\bar{X}, \bar{Y}) \leq \delta]} M(\bar{Y}, \epsilon) \quad (4.6)$$

Correspondingly, the *multi-granularity deviation factor*  $MDEF(\bar{X}, \epsilon, \delta)$  at level  $\delta$  is expressed in terms of the ratio of the densities at a point and the average density in its neighborhood:

$$MDEF(\bar{X}, \epsilon, \delta) = 1 - \frac{M(\bar{X}, \epsilon)}{AM(\bar{X}, \epsilon, \delta)} \quad (4.7)$$

Note the similarity to LOF in terms of using the local ratios while defining the outlier score of a data point. The larger the value of the MDEF is, the greater the outlier score. In order to convert the MDEF score into a binary label, the deviation  $\sigma(\bar{X}, \epsilon, \delta)$  of the different values of  $M(\bar{X}, \epsilon)$  within the sampling neighborhood of  $\bar{X}$  is computed.

$$\sigma(\bar{X}, \epsilon, \delta) = \frac{STD_{[\bar{Y}: \text{dist}(\bar{X}, \bar{Y}) \leq \delta]} M(\bar{Y}, \epsilon)}{AM(\bar{X}, \epsilon, \delta)}$$

Here, the function  $STD$  computes the standard deviation over the entire sampling neighborhood. The denominator accounts for the fact that the MDEF value of Equation 4.7 is scaled by the same expression in the denominator.

The value of  $\epsilon$  is always chosen to be half that of  $\delta$  in order to enable fast approximate computation. Therefore, throughout this presentation, it is assumed that the value of  $\epsilon$  is automatically decided by the choice of  $\delta$ . Multiple values of  $\delta$  are used in order to provide a multi-granularity approach for outlier analysis. These methods vary the sampling radius from a minimum radius containing at least 20 points to a maximum radius that spans most of the data. A data point is an outlier if its MDEF value is unusually large among *any* of the values computed at different granularity levels. Specifically, the value of the MDEF needs to be at least  $k \cdot \sigma(\bar{X}, \epsilon, \delta)$ , where  $k$  is chosen to be 3. This choice of  $k$  is common in statistical analysis with the use of the normal distribution assumption.

The algorithmic efficiency can be improved with the following modifications:

- Only a limited set of sampling neighborhoods need to be considered. In particular, if the sampling or counting neighborhoods do not change for small changes in  $\delta$ , then those neighborhoods do not need to be considered.
- Fast methods for approximating the neighbor counts are also provided in [426]. This provides high-quality approximations to MDEF. It has been shown in [426], that a box count of a grid-based division of the data provides a fast approximation, when  $L_\infty$  distances are used. This approximation is also referred to as the aLOCI algorithm.

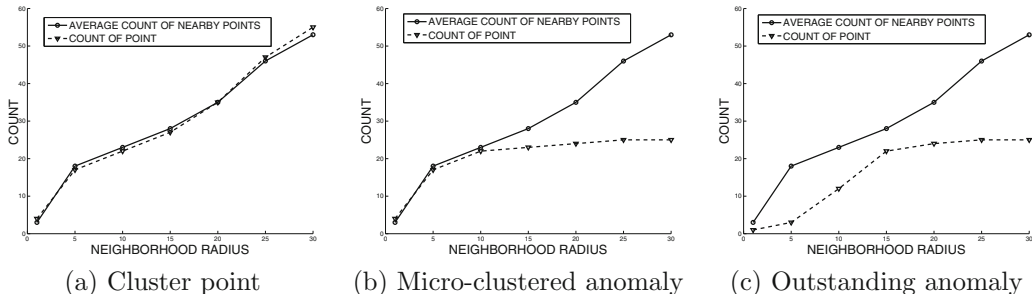


Figure 4.7: Illustrative example of a simplified version of the LOCI plot without ranges shown. The ranges can further help in distinguishing when the true density of a point falls below the average density of its neighborhood. The figure is drawn for illustrative purposes only to show the typical types of trends one would see in various settings.

#### 4.4.2.1 LOCI Plot

The LOCI plot compresses the information about a data point in a two dimensional representation, where the outlier behavior is visually interpretable from a multi-granular perspective. Since the value of  $MDEF(\bar{X}, \epsilon, \delta)$  is constructed by examining the relative behavior of  $M(\bar{X}, \epsilon)$  and  $AM(\bar{X}, \epsilon, \delta)$  over different values of  $\delta$ , it makes sense to visualize each of these quantities by separately plotting them against the sampling neighborhood  $\delta$ . Therefore, the LOCI plot shows the value of  $\delta$  on the  $X$ -axis against the following count-based quantities on the  $Y$ -axis:

- **Density behavior of point:** The value of  $M(\bar{X}, \epsilon) = M(\bar{X}, \delta/2)$  is plotted on the  $Y$ -axis. This shows the actual density behavior of the data point  $\bar{X}$  at different granularity levels.
- **Average density behavior of nearby points:** The values of  $AM(\bar{X}, \epsilon, \delta)$  and  $AM(\bar{X}, \epsilon, \delta) \pm STD_{[\bar{Y}:dist(\bar{X}, \bar{Y}) \leq \delta]} M(\bar{Y}, \epsilon)$  are plotted on the  $Y$ -axis. This shows the density behavior of the neighborhood of  $\bar{X}$  (along with statistical ranges) for different granularity levels. The ranges help in identifying cases in which the actual density of the point falls significantly below the true density of its neighborhood.

When a data point is an outlier, its density behavior will often fall below that of the point neighborhood, and it might even fall below the lower end of the neighborhood-density range.

The LOCI plot provides a *visual* understanding of how the deviations of the data point relate to extreme values of the deviation at different granularity levels, and it *explains* why a particular data point may have a high MDEF value. The use of different granularity levels is helpful in adjusting the algorithm to the vagaries of different data sets. For example, in the case of Figure 4.2, any distance-based or the LOF method would need to select the value of  $k$  (for the  $k$ -nearest neighbor method) very carefully in order to identify these data points as outliers. However, the LOCI plot would always visually enable a correct point-specific level of granularity of analysis. In some cases, specific data points may show up as outliers only at particular granularity levels, which would also show up in the LOCI plot. This type of visual insight is helpful for building intuition and interpretability in unsupervised problems like outlier detection.

An illustrative example of the LOCI plot is shown in Figure 4.7. Here, we have shown a simplified version of the plot without showing the upper and lower ranges of the aver-

age neighborhood density. In other words, we have shown only the neighborhood density  $AM(\bar{X}, \epsilon, \delta)$  without showing the ranges  $AM(\bar{X}, \epsilon, \delta) \pm STD_{[\bar{Y}:dist(\bar{X}, \bar{Y}) \leq \delta]}M(\bar{Y}, \epsilon)$ . These ranges are further helpful in distinguishing when the point density falls far below neighborhood density. Figure 4.7(a) shows the typical behavior of a clustered point in which the density of a point is roughly similar to the neighborhood density over all values of the radius. In the case of a clustered anomaly, which co-occurs with a small number of points, the two densities diverge only after sufficiently increasing the neighborhood size (see Figure 4.7(b)) beyond the size of the small cluster. Clustered anomalies tend to be common because of the propensity of outliers to occur in small groups. In the case of an outstanding outlier, as shown in Figure 4.7(c), the two densities diverge at the very beginning. Thus, these different plots provide insights about why specific points should be considered outliers. It is also noteworthy that the LOCI plot is specific to a single data point; therefore, it is usually leveraged only over a small set of interesting points identified by the LOCI algorithm.

### 4.4.3 Histogram-Based Techniques

Histograms use a space-partitioning methodology for density-based summarization. In the simplest case of univariate data, the data is discretized into bins of equal width between the minimum and maximum values, and the frequency of each bin is estimated. Data points that lie in bins with very low frequency are reported as outliers. In the context of multivariate data, the approach can be generalized in two different ways:

- The outlier scores are computed separately for each dimension, and then the scores can be aggregated.
- The discretization along each dimension can be generated at the same time, and a grid structure can be constructed. The distribution of the points in the grid structure can be used in order to create a model of the sparse regions. The data points in these sparse regions are the outliers.

In some cases, the histogram is constructed only on a sample of the points (for efficiency reasons), but all points are scored based on the frequency of a bin in which a point lies. Let  $f_1 \dots f_b$  be the (raw) frequencies of the  $b$  univariate or multivariate bins. These frequencies represent the outlier scores of the points inside these bins. Smaller values are more indicative of outlierness. In some cases, the frequency count (score) of a data point is adjusted by reducing it by 1. This is because the inclusion of the data point itself in the count can mask its outlierness during extreme value analysis. This adjustment is particularly important if a sample of the data is used to construct the histogram, but out-of-sample points are scored using the frequencies of their relevant bins in the constructed histogram. In such cases, the scores of only the in-sample points are reduced by 1. Therefore, the adjusted frequency  $F_j$  of the  $j$ th point (belonging to the  $i$ th bin with frequency  $f_i$ ) is given by the following:

$$F_j = f_i - I_j \quad (4.8)$$

Here,  $I_j \in \{0, 1\}$  is an indicator variable depending on whether or not the  $j$ th point is an in-sample point. Note that  $F_j$  is always nonnegative, because bins containing in-sample points always have a frequency of at least 1.

It is noteworthy the logarithm of the adjusted frequency  $F_j$  represents a log-likelihood score, which enables better extreme-value analysis for score to label conversion. For regularization purposes, we use  $\log_2(F_j + \alpha)$  as the outlier score of the  $j$ th point, where  $\alpha > 0$  is the

regularization parameter. To convert the scores to binary labels, a Student's  $t$ -distribution or normal distribution can be used to determine unusually low scores with extreme-value analysis. Such points are labeled as outliers. Histograms are very similar to clustering methods because they summarize the dense and sparse regions of the data in order to compute outlier scores; the main difference is that clustering methods partition the data *points*, whereas histogram methods tend to partition the data *space* into regions of equal size.

The major challenge with histogram-based techniques is that it is often hard to determine optimal histogram width well. Histograms that are too wide or too narrow will not model the frequency distribution at the level of granularity needed to optimally detect outliers. When the bins are too narrow, the normal data points falling in these bins will be identified as outliers. On the other hand, when the bins are too wide, anomalous data points may fall in high-frequency bins, and will therefore not be identified as outliers. In such a setting, it makes sense to vary the histogram width and obtain multiple scores for the same data point. These (log-likelihood) scores are then averaged over the different base detectors to obtain a final result. Like clustering, histogram-based methods tend to have a high variability in prediction depending on parameter choices, which makes them ideal candidates for ensemble methods. For example, the *RS-Hash* method (cf. section 5.2.5 of Chapter 5) varies the dimensionality and size of grid regions to score points as outliers. Similarly, some recent ensemble methods like isolation forests (cf. section 5.2.6) can be viewed as randomized histograms in which the grid regions of varying size and shapes are created in hierarchical and randomized fashion. Instead of measuring the number of points in a grid region of fixed size, an indirect measure of the expected size of the grid region required to isolate a single point is reported as the outlier score. This type of approach can avoid the problem of selecting a fixed grid size up front.

A second (related) challenge with the use of histogram-based techniques is that their space-partitioning approach makes them myopic to the presence of clustered anomalies. For example, in the case of Figure 4.2, a multivariate grid-based approach may not be able to classify an isolated group of data points as outliers, unless the resolution of the grid structure is calibrated carefully. This is because the density of the grid only depends on the data points inside it, and an isolated group of points may create an artificially dense grid cell, when the granularity of representation is high. This issue can also be partially resolved by varying the grid width and averaging the scores.

Histogram methods do not work very well in higher dimensionality because of the sparsity of the grid structure with increasing dimensionality, unless the outlier score is computed with respect to carefully chosen lower dimensional projections. For example, a  $d$ -dimensional space will contain at least  $2^d$  grid-cells, and therefore, the number of data points expected to populate each cell reduces exponentially with increasing dimensionality. As discussed in section 4.4.5, the use of some techniques like rotated bagging [32] and subspace histograms (cf. section 5.2.5 of Chapter 5) can partially address this issue. Although histogram-based techniques have significant potential, they should be used in combination with high-dimensional subspace ensembles (see Chapters 5 and 6) for best results.

#### 4.4.4 Kernel Density Estimation

Kernel-density estimation methods are similar to histogram techniques in terms of building density profiles. However, a smoother version of the density profile is constructed by using kernel functions rather than discrete counting. Because of the similarity between the two classes of methods, one tends to obtain similar results in the two cases, especially if ensemble methods are used to smooth out the implementation- and parameter-specific effects.

In kernel density estimation [496], which is also known as the Parzen-Rosenblatt method, a continuous estimate of the density is generated at a given point. The value of the density at a given point is estimated as the sum of the smoothed values of kernel functions  $K'_h(\cdot)$  associated with each point in the data set. Each kernel function is associated with a kernel width  $h$  that regulates the level of smoothing. The kernel density estimate,  $\hat{f}(\bar{X})$ , based on  $N$  data points and kernel function  $K'_h(\cdot)$  is defined as follows:

$$\hat{f}(\bar{X}) = \frac{1}{N} \cdot \sum_{i=1}^N K'_h(\bar{X} - \bar{X}_i) \quad (4.9)$$

Thus, each discrete point  $\bar{X}_i$  in the data set is replaced by a continuous function  $K'_h(\cdot)$  which peaks at  $\bar{X}_i$  and has a variance that is determined by the smoothing parameter  $h$ . An example of such a distribution would be the Gaussian kernel with width  $h$  for  $d$ -dimensional data:

$$K'_h(\bar{X} - \bar{X}_i) = \left( \frac{1}{h\sqrt{2\pi}} \right)^d \cdot e^{-\frac{\|\bar{X} - \bar{X}_i\|^2}{2h^2}} \quad (4.10)$$

The estimation error is defined by the kernel width  $h$ , which is chosen in a data-driven manner. It has been shown [496] that for most smooth functions  $K'_h(\cdot)$ , when the number of data points goes to infinity, the estimator  $\hat{f}(\bar{X})$  asymptotically converges to the true density function  $f(\bar{X})$ , provided that the width  $h$  is chosen appropriately. For a data set containing  $N$  points, the Silverman approximation rule [496] suggests a bandwidth of  $h = 1.06 \hat{\sigma} N^{-1/5}$ , where the  $\hat{\sigma}^2$  is the estimated sample variance. This choice is, however, only a rule-of-thumb; in general, the best choice of bandwidth is data-specific. The kernel density estimate at a given point is a data-driven estimate of the value of the probability density function of the generating data distribution. Good results have been shown in [184] with a density-estimation method proposed in [316]. One should exclude the test point from the summation of Equation 4.9 to avoid overfitting, which is particularly important with certain types of kernel functions.

One can view the kernel density estimate as a simple non-parametric alternative to the fit values computed by the expectation-maximization algorithm in Chapter 2. Therefore, the kernel density estimates can be viewed as outlier scores in which smaller values indicate a greater degree of outlierness. The data points with unusually low density are declared outliers with the use of a Student's  $t$ -distribution or normal distribution assumption. However, the logarithmic function should be applied to the scores before extreme-value analysis, because such analysis is more effective on the log-likelihood scores.

Density-based methods face similar implementation challenges as histogram techniques. Just as the choice of grid-width causes a dilemma in histogram methods, the proper choice of bandwidth in kernel density methods is often data-distribution specific, which is not known a priori. Furthermore, the use of a global bandwidth in order to estimate density may not work very well in cases where there are wide variations in local density such as Figures 4.2 and 4.6. In a sense, the bandwidth plays a similar role in kernel-density estimation as the grid width in histogram methods, because both regulate the locality size of the estimation process. Furthermore, these methods are not very effective for higher dimensionality, because the accuracy of the density estimation process degrades with increasing dimensionality.

#### 4.4.4.1 Connection with Harmonic $k$ -Nearest Neighbor Detector

The harmonic  $k$ -nearest neighbor detector of Definition 4.3.3 is a special case of density-based methods. Specifically, the kernel function is simply set to the inverse distance to the

target. We replicate Equation 4.9 here:

$$\begin{aligned}\hat{f}(\bar{X}) &= \frac{1}{N} \cdot \sum_{i=1}^N K'_h(\bar{X} - \bar{X}_i) \\ &\propto \frac{1}{N} \cdot \sum_{i=1}^N 1/|\bar{X} - \bar{X}_i|\end{aligned}$$

Note that the second expression is simply the inverse of the harmonic score when the value of  $k$  is set to  $N$  in Definition 4.3.3. It is important to remove any training point  $\bar{X}_i$  that is identical to  $\bar{X}$  to avoid infinity scores caused by overfitting. Note that this kernel function does not use the bandwidth, and it is therefore a parameter-free detector. One can also use only the nearest  $k < N$  points for the estimation in lieu of the bandwidth, although reasonably good results are often obtained at  $k = N$ . It is noteworthy that an (arithmetically) averaged  $k$ -nearest neighbor detector cannot discover isolated points near the center of the data distribution when  $k$  is set to  $N$ . For example, a single outlier at the center of a ring of points would receive the most *inlier-like* score in an average  $N$ -nearest neighbor detector. However, a harmonic  $N$ -nearest neighbor detector will do an excellent job in scoring the outlier appropriately, particularly if the value of  $N$  is large. This is because kernel densities are always estimated more accurately with larger data sets.

With an increasing number of points, this approach will indeed become a good *heuristic* estimate of the relative density. However, in many cases, one can obtain better results by repeatedly estimating  $\hat{f}(\bar{X})$  with different samples of the data and averaging the scores to create an ensemble method. As in the case of all density-based methods, a logarithm function should be applied to  $\hat{f}(\bar{X})$  (to convert it into a log-likelihood) before applying extreme-value analysis or ensemble methods.

#### 4.4.4.2 Local Variations of Kernel Methods

It is possible to make kernel methods more locality sensitive by using a variety of tricks either in isolation or in combination:

1. The bandwidth for estimating the density at a point can be computed *locally* by using its  $k$ -nearest neighbor distance rather than using a global value.
2. The kernel density at a point can be normalized using the average kernel densities at its neighboring points.

These two tricks were used in combination in [342]. In addition, the distance in the exponent of the kernel function was replaced with the reachability distance (as defined for the LOF algorithm).

#### 4.4.5 Ensemble-Based Implementations of Histograms and Kernel Methods

Histogram- and kernel-based techniques can be significantly strengthened using ensemble methods. The main problem in histogram-based techniques is to use them effectively in the context of high-dimensional data. In this context, it is possible to design effective implementations for high-dimensional data with the use of *rotated bagging* [32], because rotated bagging drastically reduces the dimensionality of the data from  $O(d)$  to  $O(\sqrt{d})$ . The basic

idea is to generate a randomly rotated subspace of dimensionality  $2 + \lceil \sqrt{d}/2 \rceil$  and project the data points in this subspace before applying a multidimensional histogram-based method (or any other detector). One might choose even less than  $2 + \lceil \sqrt{d}/2 \rceil$  dimensions to ameliorate the effects of high dimensions. The specific details of the rotation process are described in section 5.3.3 of Chapter 5 and in section 6.4.4 of Chapter 6.

This approach is repeated multiple times with different random projections, and the scores of a point over different ensemble components are averaged in order to compute the final outlier score. The natural ensemble-based approach of rotated bagging implicitly constructs a more complex model than the simple histogram constructed by the base detectors. As a result, accuracy can be greatly improved. Note that rotated bagging was designed [32] as a meta-algorithm for *any* base detector and not just histogram-based methods.

A related method, referred to as *LODA* [436], shows how one can use 1-dimensional random projections with  $\sqrt{d}$  non-zero elements in order to score the data points effectively. The histograms are built on 1-dimensional projections containing vectors of  $\sqrt{d}$  non-zero elements, each of which is drawn from a standard normal distribution. The histograms are used to compute a log-likelihood score of each data point because it is easy to obtain probabilistic scores from histogram- and kernel density-based methods. The log-likelihood scores of each point from the different ensemble components are averaged in order to compute the final outlier score. The basic idea here is that even though 1-dimensional histograms are very weak detectors, it is often possible to combine such weak detectors in an ensemble in order to create a very strong outlier detector. The *LODA* method can be viewed as a special case of rotated bagging [32] in which 1-dimensional projections are combined with histogram-based detectors in order to obtain high-quality results. The *RS-Hash* method samples axis-parallel subspaces of varying dimensionality to score data points as outliers. When histogram and kernel methods are used in the ensemble-centric setting, it is important to apply the logarithm function to the estimated density before averaging the scores.

## 4.5 Limitations of Proximity-Based Detection

---

Most proximity-based methods use distances in order to define outliers at varying levels of granularity. Typically, higher levels of granularity are required for greater accuracy. In particular, methods that abstract the data by various forms of summarization do not distinguish well between true anomalies and noisy regions of low density. This weakness is manifested in the high variability in the prediction with different types of summarizations. In such cases, the variabilities in the predictions arising from the vagaries of the summarization process need to be ameliorated by ensemble methods. Furthermore, these methods need to combine global and local analysis carefully in order to find the true outliers in the data. A fully global analysis may miss important outliers as indicated in Figures 4.1 and 4.6, whereas a fully local analysis may miss small clustered groups of outliers as illustrated in Figure 4.2. At the same time, increasing the granularity of analysis can make the algorithms inefficient. In the worst-case, a distance-based algorithm with full granularity can require  $O(N^2)$  distance computations in a data set containing  $N$  records. While indexing methods can be used in order to incorporate pruning into the outlier search, the effectiveness of pruning methods reduces with increasing dimensionality because of data sparsity.

An even more fundamental limitation in the context of high dimensional data is not one of *efficiency*, but that of the *quality* of the outliers found. In the high-dimensional case, all points become almost equidistant from one another, and therefore the contrast in the distances is lost [25, 263]. This is also referred to as the curse of dimensionality, which arises

from data sparsity, and it negatively impacts many high dimensional applications [8]. With increasing dimensionality, most of the features are not informative for outlier detection, and the noise effects of these dimensions will impact proximity-based methods in a very negative way. In such cases, the outliers can be masked by the noise in the features, unless the relevant dimensions can be explicitly discovered by an outlier detection method. Since proximity-based methods are naturally designed to use all the features in the data, their quality will naturally degrade with increasing dimensionality. Some methods do exist for improving the effectiveness of such methods in increasing dimensionality with subspace methods. These methods will be discussed in Chapter 5.

## 4.6 Conclusions and Summary

---

This chapter provides an overview of the key proximity-based techniques for outlier analysis. All these methods determine the outliers in the data with the use of proximity information between data points. These methods are closely related to clustering techniques in a complementary sense; while the former finds outlier points in sparse data localities, the latter tries to determine dense data localities. Therefore, clustering is itself a common method used in proximity-based outlier analysis. With the right choice of clustering method, it is also possible to make the approach adapt to patterns of arbitrary shape in the data. Such methods can also be extended to arbitrary data types.

Proximity-based methods enjoy wide popularity in the literature because of ease of implementation and interpretability. A major challenge in using such methods is that they are typically computationally intensive, and most of the high-quality methods require  $O(N^2)$  distance computations in the worst-case. Various pruning methods can be used to improve their efficiency when binary labels are required instead of scores. Among the various pruning methods, sampling methods are among the most effective ones. Local normalization is a principle used to improve the effectiveness of various outlier detection algorithms. Two common local algorithms include the LOF and LOCI algorithms. Finally, histogram-based and kernel density estimation techniques have also been explored in literature. Although these methods have not found much popularity, they have significant potential when used in combination with ensemble methods.

## 4.7 Bibliographic Survey

---

The traditional definition of multivariate outliers often assumed them to be side products of clustering algorithms. Outliers were therefore defined as data points that do not naturally fit into any cluster. However, the non-membership of a data point in a cluster is not able to distinguish between noise and anomalies. A detailed discussion of different clustering algorithms and their use for outlier analysis may be found in [283, 307]. Many of the clustering algorithms explicitly label points that do not fit within clusters as outliers [594]. However, in some sparse high-dimensional domains, such as transaction data, (subspace)-clustering are among the few feasible methods for identifying outliers [254].

In order to improve the quality of clustering-based outlier scores, one can use the distance of data points to cluster centroids, rather than using only the membership of data points in clusters. The work in [499] investigates a number of deterministic and probabilistic methods for clustering in order to detect anomalies. These techniques were designed in the context of intrusion detection. One challenge in such methods is to prevent the clustering methods from quality-degradation from noise and anomalies already present in the data. This is



because if the discovered clusters are already biased by noise and anomalies, it will also prevent the effective identification of outliers. Such techniques have been used often in the context of intrusion-detection applications [70, 499]. The work in [70] uses a first phase in which the normal data is identified, by using data points matching frequent patterns in the data. These normal data are subsequently used in order to perform robust clustering. The outliers are then determined as points which lie at a significant distance to these clusters. These methods can be considered a kind of sequential ensemble approach.

A number of outlier detection methods have also been proposed for cases where the anomalies may lie in small clusters [188, 253, 291, 420, 445, 446]. Many of these techniques work by using distance-thresholding in order to regulate the creation of new clusters. When a data point does not lie within a specified threshold distance of the nearest cluster centroid, a new cluster is created containing a single instance. This results in clusters of varying size, since some of the newly created clusters do not get a sufficient number of points added to them. Then, the outlieriness of a data point may be decided both by the number of points in its cluster, and the distance of its cluster to the other clusters. A number of indexing techniques have also been proposed in order to speed to the partitioning of the data points into clusters [131, 518]. Biased sampling [321] has also been shown to be an effective and efficient method for clustering-based outlier detection. A facility-location model for integrated clustering and outlier detection is proposed in [422]. The work in [475] shows how to construct robust spectral embeddings for outlier detection. Many of these methods can also be extended to arbitrary data types.

Distance-based methods have been extremely popular in the literature because of their ability to perform the analysis at a higher level of granularity than clustering methods. Furthermore, such methods are intuitive and extremely easy to understand and implement. The first distance-based method was proposed in [317]. The ideas in this work were extended to finding intensional knowledge in [318]. Subsequently, indexing methods were designed to improve the efficiency of this method in [456]. The work in [58] uses linearization in which the multidimensional space is populated with Hilbert space-filling curves. This 1-d representation has the advantage that the  $k$ -nearest neighbors can be determined very *fast* by examining the predecessors and successors of a data point on the space-filling curve. The sum of the  $k$ -nearest neighbor distances on the linearized representation is used in order to generate the outlier score of a data object. While the use of the *sum* of the  $k$ -nearest neighbor distances has some advantages over the  $k$ -nearest neighbor distance in differentiating between sparsely populated data and clustered data, it has the disadvantage of (sometimes) not being able to detect groups of isolated anomalies as illustrated in Figure 4.2. One challenge with the use of space-filling curves is that they map the data into a hypercube with  $d$  dimensions, and the number of corners of this hypercube increases exponentially with  $d$ . In such cases, the sparsity of the data in high dimensions may result in a degradation of the locality behavior of the space-filling curve. In order to address this issue, the work in [58] uses data-shifting techniques in order to improve locality. An iterative technique was designed, which requires  $d + 1$  scans of the data set.

The work in [75] designs a simple sampling-based pruning technique in order to improve the efficiency of a  $k$ -nearest neighbor-based outlier detection technique. The core idea is similar to the pruning rule used in [456]. The idea is that if the outlier score for an object is less than the  $k$ -nearest neighbor distance of the  $r$ th best outlier, that data point cannot possibly be an outlier and is pruned from further consideration. This simple pruning rule has been shown in [75] to work well with randomized data. The randomization itself can be done in linear time by using a disk-based shuffling technique. The work in [572] performs the nearest neighbor computations on a smaller sample of the data set in order to improve

the efficiency. Theoretical guarantees are provided in order to bound the loss in accuracy resulting from the sampling process.

The effectiveness of pruning methods is clearly dependent on the ability to generate a good bound on the  $k$ -nearest neighbor distances in an efficient way. Therefore, the work in [219] partitions the data into small clusters. The  $k$ -nearest neighbor distance of a data point within a cluster is used in order to generate an upper bound on the  $k$ -nearest neighbor distance of that point. If this upper bound is less than the scores of the set of outliers already found, then the point can be pruned from consideration. The work in [456] also uses clustering techniques for pruning. The method in [219] uses recursive hierarchical partitioning in order to ensure that each cluster is assigned a similar number of data points. The ordering of the data points along the principal component of largest variance is used in order to provide a quick estimate of the  $k$ -nearest neighbor distance. In some cases, more accurate outlier detection can be performed by using a data-dependent similarity measure in  $k$ -nearest neighbor methods.

A discussion of data-dependent similarity functions is provided in [33, 491, 548]. Methods for data-dependent similarity include the shared-nearest neighbor measure [287], the pairwise global/local Mahalanobis adaptations (cf. Chapter 3 [33]), random forests [99, 491], randomized clustering trees [401, 555], and isolation forests [548]. It is noteworthy that isolation forests can be viewed as adaptations of extremely randomized clustering forests (ERC-Forests) [401] with a single trial at each node and growing the tree to full height with singleton points at the leaves.

A resolution-based method was proposed in [190]. According to this method, whether a point belongs to a cluster or whether it is an outlier depends on the distance threshold. At the highest resolution level, all points are in individual clusters of their own and are therefore outliers. As the resolution is slowly reduced, more and more data points join clusters. In each step, each point changes from being an outlier to a cluster. Based on this, a *Resolution Outlier Factor (ROF)* value was defined in [190]. This was shown to provide effective results for outlier analysis.

Most of the distance-based algorithms are designed with the use of Euclidean distances. In practice, the Euclidean function may not be optimal for finding the outliers. In fact, for many other domains of data, the distance functions are often defined in a fairly complex way, and many of the pruning techniques designed for Euclidean spaces will not work well in arbitrary spaces. In this context, an efficient algorithm was designed for outlier detection in arbitrary metric spaces [533], which requires at most three passes over the data.

A method to improve the efficiency of distance-based algorithms with the use of reference points was proposed in [430]. The core idea in this work is to rank the data points on the basis of their relative degree of density with respect to a fixed set of  $R$  reference points. Each data point is transformed to a 1-dimensional space in  $R$  possible ways on the basis of their distance to a reference point. For each of these  $R$  1-dimensional data sets, the relative degree of density of each data point with respect to the corresponding reference point is computed. The overall relative degree of density of a data point is defined as the minimum relative degree of density over all the reference points. This relative degree of density provides a way to rank the different data points. Distributed algorithms for speeding up outlier detection are proposed in [83].

Scalability is a significant issue in the context of the data streams. Typically, in the case of data streams, a past window of history is used in order to determine outliers. Data points whose  $k$ -nearest neighbor values are large in a specific sliding window history are declared outliers [60, 322]. Stream-clustering methods such as those in [28] can also be used in order to speed up the outlier analysis process. Such an approach has been discussed in [322].

The issue of local density in the context of outlier analysis was first addressed in [96, 97]. We note that the reverse nearest neighbor approach [100, 248] presented in this chapter shares some similarities to LOF in terms of adjusting to local densities with the use of a reverse nearest-neighbor approach. The variations in local density may result in poor ranking of outliers by global distance-based methods. Therefore, the concept of Local Outlier Factor (LOF) was proposed in [96]. These methods adjust the outlier score of an object on the basis of the local density. It should be mentioned that the concept of density is really loosely defined in LOF as an inverse of averaged distances. A true definition of density should really count the number of data points in a specific volume. Data points in local regions of high density are given a higher outlier score, even if they are slightly isolated from the other points in their locality. A comparison of the LOF algorithm with respect to the average  $k$ -nearest neighbor algorithm is provided in [32]. It is shown that the average  $k$ -nearest neighbor algorithm is more robust.

Many different variants of the broad LOF approach were subsequently proposed. For example, the work in [293] proposed the concept of top- $n$  local outliers, where the top- $n$  outliers were determined on the basis of the density. Pruning techniques were used to improve the running time by partitioning the data into clusters, and computing bounds on the LOF values of the points in each cluster. Thus, entire clusters can be pruned if they are guaranteed to contain only points that have lower LOF values than the weakest of the current top- $n$  outliers. Other methods for improving the effectiveness of top- $n$  local outlier detection with the use of cluster-pruning were proposed in [144].

One issue with LOF is that it can sometimes be ineffective when regions of different density are not clearly separated. Therefore, the INFLO technique of [294] proposed a modification of LOF, which uses a symmetric nearest-neighbor relationship based on the nearest-neighbor distance as well as the reverse nearest-neighbor distance to order to define the local outliers. The concept of *connectivity-based outlier factor (COF)* was proposed in [534], which is also able to find outliers in low density or arbitrarily shaped regions effectively. The main difference between COF and LOF is the way in which COF defines the neighborhood of a data point. Specifically, the neighborhood is defined incrementally by adding the closest point to the current neighborhood set. This approach is based on a similar motivation as single-linkage clustering, because it merges the neighborhood-set (viewed as a cluster) and points (viewed as singleton clusters) based on exactly the same criterion as single-linkage clustering. Therefore, it can define arbitrarily shaped neighborhoods effectively when the points are distributed on arbitrary lower dimensional manifolds of the data. The LOF approach has also been combined with other clustering techniques. For example, the work in [253, 255] defines a score called *Cluster-Based Local Outlier Factor (CBLOF)* in which anomalies are defined as a combination of local distances to nearby clusters and the size of the clusters to which the data point belongs. Data points in small clusters that are at a large distance to nearby clusters are flagged as outliers.

The LOF scheme has also been extended to the case of spatial data with non-spatial attributes [523]. For example, the sea-surface temperature is a non-spatial attribute in the context of spatial location. Such data are known to exhibit *spatial auto-correlations*, in which the value of an element is affected by its immediate neighbors (e.g., spatial temperature locality). Furthermore, the data shows *spatial heteroscedasticity*, in which the variance of a data point is based on its location. For example, “normal” temperature variations are clearly based on geographical location. We note that spatial data share some similarities with temporal data from the perspective of *spatial continuity*, which is analogous to *temporal continuity*. Correspondingly, the work in [523] defines a local outlier measure, known as the *Spatial Local Outlier Measure (SLOM)*, which is specially suited to spatial outlier detection.

The generalization of the LOF method to the streaming scenario is discussed in [443].

The LOCI method [426] is also a locally sensitive method, which uses the number of points in a circular neighborhood around a point, rather than the inverse of the  $k$ -nearest neighbor distances for local density computation. Thus, it is truly a density-based method from an intuitive perspective. Furthermore, the approach is tested over different levels of granularity in order to reduce the parameter choices and remove the need for *some* of the input parameters during the outlier detection process. An approximate version of the algorithm can be implemented in almost linear time. An interesting contribution of this work is the introduction of LOCI plots, which provide an intuitive understanding of the outliers in the data with a visual plot. The LOCI plot provides an understanding of how different sizes of neighborhoods may correspond to the outlier score of a data point.

The traditional methods for density-based outlier analysis involve the use of discretization, grid-based methods, and kernel-density based methods. The first two belong in the general category of histogram-based methods [260, 288]. Histogram-based techniques find wide applicability in intrusion-detection techniques, in which it is natural to construct frequency-based profiles of various events. The main challenge in histogram-based methods is that the bucket-size along each dimension can sometimes be hard to pick correctly. The work in [476] proposes the *RS-Hash* method to perform subspace outlier detection in linear time by randomly varying the sizes and dimensions of the different grid regions in an ensemble-centric approach. The approach has also been extended to data streams in the same work. A closely related method is that of kernel-density estimation [262, 496]. Local variations of this approach are discussed in [342, 483]. Kernel density-estimation is a continuous variation of grid-based methods, in which a smooth kernel function is used for the estimation process. A particularly robust form of kernel density estimation, which has been shown [184] to achieve good results for anomaly detection, is provided in [316].

## 4.8 Exercises

---

1. Consider a data set with the following observations:  $\{ (1, 3), (1.01, 3.01), (0.99, 3.01), (0.99, 3), (0.99, 2.99), (3, 1) \}$ .
  - What are the results of linear modeling on this data set with 1-dimensional PCA for finding outliers?
  - How well does a 1-NN technique work for finding outliers in this case?
  - Discuss the main challenge in using PCA in this case. Is there any way that you can improve the scores from PCA?
2. Consider a data set containing a single cluster with the points  $\{ (1, 1), (0, 0), (2, 2.1), (3, 3.1), (4, 4), (5.1, 5) \}$ .
  - Consider the two points  $(6.5, 6.5)$ , and  $(1, 2.1)$ . Draw the points on a piece of paper. Which of the two data points seems more like an outlier?
  - Which point does a 1-NN algorithm set as the highest outlier score with the Euclidean metric?
  - Which point does a 1-NN algorithm set as the lowest outlier score with the Euclidean metric?
  - Which data point does a rank-1 PCA-based algorithm set at the highest outlier rank, when the residuals are the outlier scores?

- Would you recommend changing the distance function from the Euclidean metric? How?
3. Download the Ionosphere data set from the UCI Machine Learning Repository [203].
    - Rank the data points based on their residual scores in a PCA approach, when only the top 3 eigenvectors are used.
    - Rank the data points based on their  $k$ -nearest neighbor scores, for values of  $k$  ranging from 1 through 5.
    - Normalize the data, so that the variance along each dimension is 1. Rank the data points based on their  $k$ -nearest neighbor scores, for values of  $k$  ranging from 1 through 5.
    - How many data points are common among the top 5 ranked outliers using different methods?
    - Now use a voting scheme, which adds up the ranks of the outliers in different schemes. Which are the top 5 outliers now?
    - Does this ensemble approach provide more robust outliers?
  4. Repeat Exercise 3 with the network intrusion data set from the UCI Machine Learning Repository.
  5. A manufacturing company produces 2-dimensional square widgets, which are normally distributed with a length of 1 meter on each side, and a standard deviation of 0.01 meters.
    - Generate a data set with 100,000 widgets from this distribution.
    - The company produced 5 anomalous widgets, due a defect in the manufacturing process. Each such widget had a square length of 0.1 meters, and standard deviation of 0.001 meters. Generate these 5 anomalous points using the normal distribution assumption.
    - Does a 1-NN approach find the anomalous widgets?
    - Does a 10-NN approach find the anomalous widgets?
  6. Apply a  $k$ -means clustering approach to the data set in Exercise 5, where 5 cluster centroids are used. As a post-processing step, remove any clusters with 10 or less data points. Score the data points by their distance to their closest cluster centroids. Which data points have the highest outlier scores?
  7. Apply the reverse 1-NN algorithm to the case of Exercise 5. Which data points have the highest outlier scores? Which data points have the highest outlier scores with the reverse 10-NN algorithm? With the reverse 100-NN algorithm?
  8. Repeat Exercises 3 and 4 with the use of the LOF method and determine the ranking of the outliers. Are the outliers same in this case as those found in Exercises 3 and 4?
  9. Repeat Exercise 8 with the use of the LOCI method. Are the outliers found to be the same?

---

---

## Chapter 5

# High-Dimensional Outlier Detection: The Subspace Method

---

---

“In view of all that we have said in the foregoing sections, the many obstacles we appear to have surmounted, what casts the pall over our victory celebration? It is the curse of dimensionality, a malediction that has plagued the scientist from the earliest days.”– Richard Bellman

### 5.1 Introduction

---

Many real data sets are very high dimensional. In some scenarios, real data sets may contain hundreds or thousands of dimensions. With increasing dimensionality, many of the conventional outlier detection methods do not work very effectively. This is an artifact of the well-known *curse of dimensionality*. In high-dimensional space, the data becomes sparse, and the true outliers become masked by the noise effects of multiple irrelevant dimensions, when analyzed in *full dimensionality*.

A main cause of the dimensionality curse is the difficulty in defining the *relevant* locality of a point in the high-dimensional case. For example, proximity-based methods define locality with the use of distance functions on all the dimensions. On the other hand, all the dimensions may not be relevant for a specific test point, which also affects the quality of the underlying distance functions [263]. For example, all pairs of points are almost equidistant in high-dimensional space. This phenomenon is referred to as *data sparsity* or *distance concentration*. Since outliers are defined as data points in sparse regions, this results in a poorly discriminative situation where all data points are situated in almost equally sparse regions in full dimensionality. The challenges arising from the dimensionality curse are not specific to outlier detection. It is well known that many problems such as clustering and similarity search experience qualitative challenges with increasing dimensionality [5, 7, 121, 263]. In fact, it has been suggested that almost any algorithm that is based on the notion of proximity would degrade qualitatively in higher-dimensional space, and would therefore need to

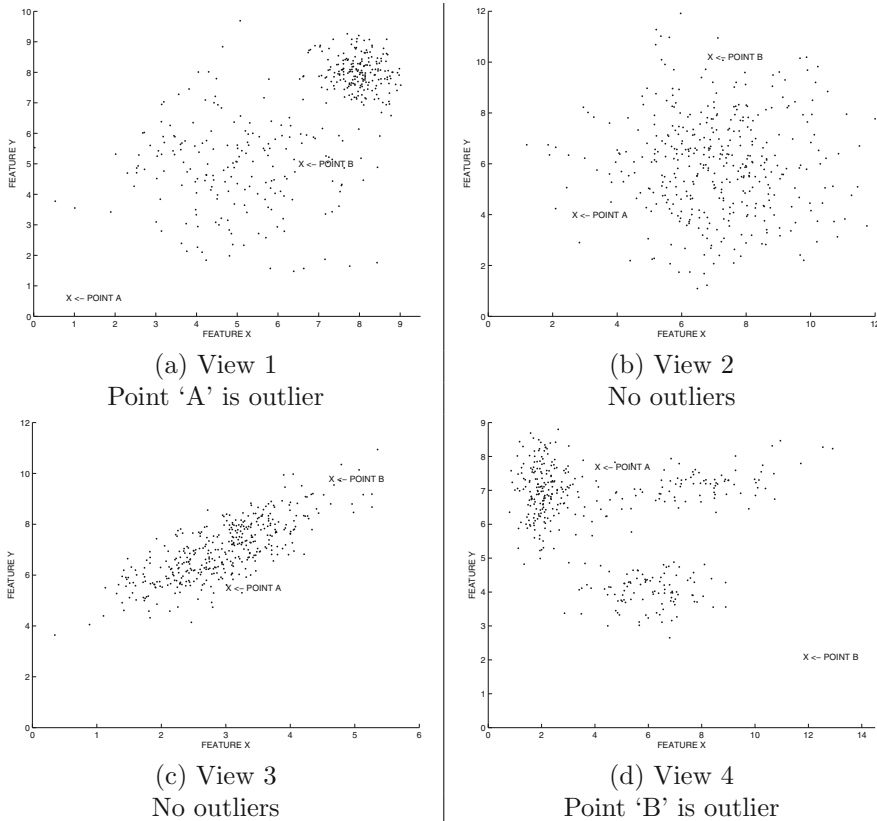


Figure 5.1: The outlier behavior is masked by the irrelevant attributes in high dimensions.

be re-defined in a more meaningful way [8]. The impact of the dimensionality curse on the outlier detection problem was first noted in [4].

In order to further explain the causes of the ineffectiveness of full-dimensional outlier analysis algorithms, a motivating example will be presented. In Figure 5.1, four different 2-dimensional views of a hypothetical data set have been illustrated. Each of these views corresponds to a disjoint set of dimensions. It is evident that point ‘A’ is exposed as an outlier in the first view of the data set, whereas point ‘B’ is exposed as an outlier in the fourth view of the data set. However, neither of the data points ‘A’ and ‘B’ are exposed as outliers in the second and third views of the data set. These views are therefore *noisy* from the perspective of measuring the outlierness of ‘A’ and ‘B.’ In this case, three of the four views are quite non-informative and noisy for exposing any *particular* outlier ‘A’ or ‘B.’ In such cases, the outliers are lost in the random distributions within these views, when the distance measurements are performed in *full* dimensionality. This situation is often naturally magnified with increasing dimensionality. For data sets of very high dimensionality, it is possible that only a very small fraction of the views may be informative for the outlier analysis process.

What does the aforementioned pictorial illustration tell us about the issue of locally relevant dimensions? The physical interpretation of this situation is quite intuitive in practical scenarios. An object may have several measured quantities, and significantly abnormal behavior of this object may be reflected only in a small subset of these quantities. For ex-

ample, consider an airplane mechanical fault-detection scenario in which the results from different tests are represented in different dimensions. The results of thousands of different airframe tests on the same plane may mostly be normal, with some noisy variations, which are not significant. On the other hand, some deviations in a small subset of tests may be significant enough to be indicative of anomalous behavior. When the data from the tests are represented in full dimensionality, the anomalous data points will appear normal in virtually all views of the data except for a very small fraction of the dimensions. Therefore, aggregate proximity measures are unlikely to expose the outliers, since the noisy variations of the vast number of normal tests will mask the outliers. Furthermore, when different objects (instances of different airframes) are tested, different tests (subsets of dimensions) may be relevant for identifying the outliers. In other words, the outliers are often embedded in *locally* relevant subspaces.

What does this mean for full-dimensional analysis in such scenarios? When full-dimensional distances are used in order to measure deviations, the dilution effects of the vast number of “normally noisy” dimensions will make the detection of outliers difficult. In most cases, this will show up as distance-concentration effects from the noise in the other dimensions. This may make the computations more erroneous. Furthermore, the additive effects of the noise present in the large number of different dimensions will interfere with the detection of actual deviations. Simply speaking, *outliers are lost in low-dimensional subspaces, when full-dimensional analysis is used, because of the masking and dilution effects of the noise in full dimensional computations* [4].

Similar effects are also experienced for other distance-based methods such as clustering and similarity search. For these problems, it has been shown [5, 7, 263] that by examining the behavior of the data in subspaces, it is possible to design more meaningful clusters that are specific to the particular subspace in question. This broad observation is generally true of the outlier detection problem as well. Since the outliers may only be discovered in low-dimensional subspaces of the data, it makes sense to explore the lower dimensional subspaces for deviations of interest. Such an approach filters out the additive noise effects of the large number of dimensions and results in more robust outliers. An interesting observation is that such lower-dimensional projections can often be identified even in data sets with missing attribute values. This is quite useful for many real applications, in which feature extraction is a difficult process and full feature descriptions often do not exist. For example, in the airframe fault-detection scenario, it is possible that only a subset of tests may have been applied, and therefore the values in only a subset of the dimensions may be available for outlier analysis. This model is referred to as *projected outlier detection* or, alternatively, *subspace outlier detection* [4].

The identification of relevant subspaces is an extraordinarily challenging problem. This is because the number of possible projections of high-dimensional data is exponentially related to the dimensionality of the data. An effective outlier detection method would need to search the data points and dimensions in *an integrated way*, so as to reveal the most relevant outliers. This is because different subsets of dimensions may be relevant to different outliers, as is evident from the example in Figure 5.1. This further adds to the computational complexity.

An important observation is that subspace analysis is generally more difficult in the context of the outlier detection problem than in the case of problems such as clustering. This is because problems like clustering are based on aggregate behavior, whereas outliers, by definition, are rare. Therefore, in the case of outlier analysis, statistical aggregates on individual dimensions in a given locality often provide *very weak* hints for the subspace exploration process as compared to aggregation-based problems like clustering. When such



weak hints result in the omission of relevant dimensions, the effects can be much more drastic than the inclusion of irrelevant dimensions, especially in the interesting cases when the number of locally relevant dimensions is a small fraction of the full data dimensionality. A common mistake is to assume that the complementarity relationship between clustering and outlier analysis can be extended to the problem of local subspace selection. In particular, blind adaptations of dimension selection methods from earlier subspace clustering methods, which are unaware of the nuances of subspace analysis principles across different problems, may sometimes miss important outliers. In this context, it is also crucial to recognize the difficulty in identifying relevant subspaces for outlier analysis. In general, selecting a single relevant subspace for each data point can cause unpredictable results, and therefore it is important to combine the results from *multiple* subspaces. In other words, subspace outlier detection is inherently posed as an ensemble-centric problem.

Several classes of methods are commonly used:

- **Rarity-based:** These methods attempt to discover the subspaces based on rarity of the underlying distribution. The major challenge here is computational, since the number of rare subspaces is far larger than the number of dense subspaces in high dimensionality.
- **Unbiased:** In these methods, the subspaces are sampled in an unbiased way, and scores are combined across the sampled subspaces. When subspaces are sampled from the original set of attributes, the approach is referred to as *feature bagging* [344]. In cases in which arbitrarily oriented subspaces are sampled, the approach is referred to as *rotated bagging* [32] or *rotated subspace sampling*. In spite of their extraordinary simplicity, these methods often work well.
- **Aggregation-based:** In these methods, aggregate statistics such as cluster statistics, variance statistics, or non-uniformity statistics of global or local subsets of the data are used to quantify the relevance of subspaces. Unlike rarity-based statistics, these methods quantify the statistical properties of global or local reference sets of points instead of trying to identify rarely populated subspaces directly. Since such methods only provide weak (and error-prone) *hints* for identifying relevant subspaces, multiple subspace sampling is crucial.

This chapter is organized as follows. Axis-parallel methods for subspace outlier detection are studied in section 5.2. The underlying techniques discuss how multiple subspaces may be combined to discover outliers. The problem of identifying outliers in generalized subspaces (i.e., arbitrarily oriented subspaces) is discussed in section 5.3. Recent methods for finding outliers in nonlinear subspaces are also discussed in this section. The limitations of subspace analysis are discussed in section 5.4. The conclusions and summary are presented in section 5.5.

## 5.2 Axis-Parallel Subspaces

---

The first work on subspace outlier detection [4] proposed a model in which outliers were defined by axis-parallel subspaces. In these methods, an outlier is defined in a subset of features from the original data. Clearly, careful quantification is required for comparing the scores from various subspaces, especially if they are of different dimensionality and use different scales of reference. Furthermore, methods are required for quantifying the

effectiveness of various subspaces in exposing outliers. There are two major variations in the approaches used by axis-parallel methods:

- In one class of methods, points are examined one by one and their relevant outlying subspaces are identified. This is inherently an instance-based method. This type of approach is computationally expensive because a significant amount of computational time may be required for determining the outlier subspaces of each point. However, the approach provides a more fine-grained analysis, and it is also useful for providing *intensional knowledge*. Such intensional knowledge is useful for describing *why* a specific data point is an outlier.
- In the second class of methods, outliers are identified by building a subspace model up front. Each point is scored with respect to the model. In some cases, each model may correspond to a single subspace. Points are typically scored by using an ensemble score of the results obtained from different models. Even in cases in which a single (global) subspace is used in a model for scoring all the points, the combination score often enhances the local subspace properties of the scores because of the ability of ensemble methods to reduce *representational bias* [170] (cf. section 6.4.3 of Chapter 6).

The fine-grained analysis of the first class of methods is often computationally expensive. Because of the computationally intensive and fine-grained nature of this analysis, it is often harder to fully explore the use of multiple subspaces for analysis. This can sometimes have a detrimental effect on the accuracy as well. The second class of methods has clear computational benefits. This computational efficiency can be leveraged to explore a larger number of subspaces and provide more robust results. Many of the methods belonging to the second category, such as feature bagging, rotated bagging, subspace histograms, and isolation forests, are among the more successful and accurate methods for subspace outlier detection.

The advantages of ensemble-based analysis are very significant in the context of subspace analysis [31]. Since the outlier scores from different subspaces may be very different, it is often difficult to fully trust the score from a single subspace, and the combination of scores is crucial. This chapter will explore several methods that leverage the advantages of combining multiple subspaces.

## 5.2.1 Genetic Algorithms for Outlier Detection

The first approach for subspace outlier detection [4] was a genetic algorithm. Subspace outliers are identified by finding *localized regions of the data in low-dimensional space* that have abnormally low density. A genetic algorithm is employed to discover such local subspace regions. The outliers are then defined by their membership in such regions.

### 5.2.1.1 Defining Abnormal Lower-Dimensional Projections

In order to identify abnormal lower-dimensional projections, it is important to provide a proper statistical definition of an abnormal lower-dimensional projection. An abnormal lower-dimensional projection is one in which the density of the data is exceptionally lower than average. In this context, the methods for extreme-value analysis introduced in Chapter 2 are useful.

A grid-based approach is used in order to identify rarely populated local subspace regions. The first step is to create grid regions with data discretization. Each attribute is

divided into  $\phi$  ranges. These ranges are created on an equi-depth basis. Thus, each range contains a fraction  $f = 1/\phi$  of the records. The reason for using equi-depth ranges as opposed to equi-width ranges is that different localities of the data may have different densities. Therefore, such an approach partially adjusts for the local variations in data density during the initial phase. These ranges form the units of locality that are used in order to define sparse subspace regions.

Consider a  $k$ -dimensional cube that is created by selecting grid ranges from  $k$  different dimensions. If the attributes are statistically independent, the expected fraction of the records in that  $k$ -dimensional region is  $f^k$ . Of course, real-world data is usually far from statistically independent and therefore the actual distribution of points in a cube would differ significantly from this expected value. Many of the local regions may contain very few data points and most of them will be empty with increasing values of  $k$ . In cases where these abnormally sparse regions are non-empty, the data points inside them might be outliers.

It is assumed that the total number of points in the database is denoted by  $N$ . Under the aforementioned independence assumption, the presence or absence of any point in a  $k$ -dimensional cube is a Bernoulli random variable with probability  $f^k$ . Then, from the properties of Bernoulli random variables, we know that the expected fraction and standard deviation of the points in a  $k$ -dimensional cube is given by  $N \cdot f^k$  and  $\sqrt{N \cdot f^k \cdot (1 - f^k)}$ , respectively. Furthermore, if the number of data points  $N$  is large, the central limit theorem can be used to *approximate* the number of points in a cube by a normal distribution. Such an assumption can help in creating meaningful measures of abnormality (sparsity) of the cube. Let  $n(\mathcal{D})$  be the number of points in a  $k$ -dimensional cube  $\mathcal{D}$ . The sparsity coefficient  $S(\mathcal{D})$  of the data set  $\mathcal{D}$  can be computed as follows:

$$S(\mathcal{D}) = \frac{n(\mathcal{D}) - N \cdot f^k}{\sqrt{N \cdot f^k \cdot (1 - f^k)}} \quad (5.1)$$

Only sparsity coefficients that are negative are indicative of local projected regions for which the density is lower than expectation. Since  $n(\mathcal{D})$  is assumed to fit a normal distribution, the normal distribution tables can be used to quantify the probabilistic level of significance of its deviation. Although the independence assumption is never really true, it provides a good practical *heuristic* for estimating the point-specific abnormality.

### 5.2.1.2 Defining Genetic Operators for Subspace Search

An exhaustive search of all the subspaces is impractical because of exponential computational complexity. Therefore, a selective search method, which prunes most of the subspaces, is required. The nature of this problem is such that there are no upward- or downward-closed properties<sup>1</sup> on the grid-based subspaces satisfying the sparsity condition. Such properties are often leveraged in other problems like frequent pattern mining [36]. However, unlike frequent pattern mining, in which one is looking for patterns with high frequency, the problem of finding sparsely-populated subsets of dimensions has the flavor of finding a needle in haystack. Furthermore, even though particular regions may be well populated on certain subsets of dimensions, it is possible for them to be very sparsely populated when such dimensions are combined. For example, in a given data set, there may be a large number of individuals clustered at the age of 20, and a modest number of individuals with high levels of severity of Alzheimer's disease. However, *very rare* individuals would satisfy

<sup>1</sup>An upward-closed pattern is one in which all supersets of the pattern are also valid patterns. A downward-closed set of patterns is one in which all subsets of the pattern are also members of the set.

both criteria, because the disease does not affect young individuals. From the perspective of outlier detection, a 20-year old with early-onset Alzheimer is a very interesting record. However, the interestingness of the pattern is not even hinted at by its lower-dimensional projections. Therefore, the best projections are often created by an unknown combination of dimensions, whose lower-dimensional projections may contain very few hints for guiding subspace exploration. One solution is to change the measure in order to force better closure or pruning properties; however, forcing the choice of the measure to be driven by algorithmic considerations is often a recipe for poor results. In general, it is not possible to predict the effect of combining two sets of dimensions on the outlier scores. Therefore, a natural option is to develop search methods that can identify such hidden combinations of dimensions. An important observation is that one can view the problem of finding sparse subspaces as an optimization problem of minimizing the count of the number of points in the identified subspaces. However, since the number of subspaces increases exponentially with data dimensionality, the work in [4] uses genetic algorithms, which are also referred to as evolutionary search methods. Such optimization methods are particularly useful in unstructured settings where there are few hard rules to guide the search process.

Genetic algorithms, also known as evolutionary algorithms [273], are methods that imitate the process of organic evolution in order to solve poorly structured optimization problems. In evolutionary methods, every solution to an optimization problem can be represented as an individual in an evolutionary system. The measure of fitness of this “individual” is equal to the objective function value of the corresponding solution. As in biological evolution, an individual has to compete with other individuals that are alternative solutions to the optimization problem. Therefore, one always works with a multitude (i.e., *population*) of solutions at any given time, rather than a single solution. Furthermore, new solutions can be created by recombination of the properties of older solutions, which is the analog of the process of biological reproduction. Therefore, appropriate operations are defined in order to imitate the recombination and mutation processes in order to complete the simulation. A mutation can be viewed as a way of exploring closely related solutions for possible improvement, much as one would do in a hill-climbing approach.

Clearly, in order to simulate this biological process, we need some type of concise representation of the solutions to the optimization problem. This representation enables a concrete algorithm for simulating the algorithmic processes of recombination and mutation. Each feasible solution is represented as a string, which can be viewed as the chromosome representation of the solution. The process of conversion of feasible solutions into strings is referred to as its *encoding*. The effectiveness of the evolutionary algorithm often depends crucially on the choice of encoding because it implicitly defines all the operations used for search-space exploration. The measure of fitness of a string is evaluated by the *fitness function*. This is equivalent to an evaluation of the objective function of the optimization problem. Therefore, a solution with a better objective function value can be viewed as the analog of a *fitter* individual in the biological setting. When evolutionary algorithms simulate the process of biological evolution, it generally leads to an improvement in the average objective function of all the solutions (population) at hand much as the biological evolution process improves fitness over time. Furthermore, because of the perpetual (selection) bias towards fitter individuals, diversity in the population of solutions is lost. This loss of diversity resembles the way in which convergence works in other types of iterative optimization algorithms. De Jong [163] defined convergence of a particular position in the string as the stage at which 95% of the population had the same value for that position. The population is said to have converged when all positions in the string representation have converged.

The evolutionary algorithm views subspace projections as possible solutions to the op-

timization problem. Such projections can be easily represented as strings. Since the data is discretized into a grid structure, we can assume that the identifiers of the various grid intervals in any dimension range from 1 to  $\phi$ . Consider a  $d$ -dimensional data point for which the grid intervals for the  $d$  different dimensions are denoted by  $(m_1, \dots, m_d)$ . The value of each  $m_i$  can take on any of the value from 1 through  $\phi$ , or it can take on the value \*, which indicates a “don’t care” value. Thus, there are a total of  $\phi+1$  possible values of  $m_i$ . Consider a 4-dimensional data set with  $\phi = 10$ . Then, one possible example of a solution to the problem is given by the string \*3\*9. In this case, the ranges for the second and fourth dimension are identified, whereas the first and third are left as “don’t cares.” The evolutionary algorithm uses the dimensionality of the projection  $k$  as an input parameter. Therefore, for a  $d$ -dimensional data set, the string of length  $d$  will contain  $k$  specified positions and  $(d - k)$  “don’t care” positions. This represents the string encoding of the  $k$ -dimensional subspace. The fitness for the corresponding solution may be computed using the sparsity coefficient discussed earlier. The evolutionary search technique starts with a population of  $p$  random solutions and iteratively uses the processes of selection, crossover, and mutation in order to perform a combination of hill climbing, solution recombination and random search over the space of possible projections. The process is continued until the population converges to a global optimum according to the *De Jong convergence criterion* [163]. At each stage of the algorithm, the  $m$  best projection solutions (most negative sparsity coefficients) are tracked in running fashion. At the end of the algorithm, these solutions are reported as the best projections in the data. The following operators are defined for selection, crossover, and mutation:

- **Selection:** The copies of a solution are replicated by ordering them by rank and biasing them in the population in the favor of higher ranked solutions. This is referred to as *rank selection*.
- **Crossover:** The crossover technique is key to the success of the algorithm, since it implicitly defines the subspace exploration process. One solution is to use a uniform two-point crossover in order to create the recombinant children strings. The two-point crossover mechanism works by determining a point in the string at random called the crossover point, and exchanging the segments to the right of this point. However, such a blind recombination process may create poor solutions too often. Therefore, an optimized crossover mechanism is defined. In this case, it is guaranteed that both children solutions correspond to a  $k$ -dimensional projection as the parents, and the children typically have high fitness values. This is achieved by examining a subset of the different possibilities for recombination and selecting the best among them. The basic idea is to select  $k$  dimensions greedily from the space of (at most)  $2 \cdot k$  distinct dimensions included in the two parents. A detailed description of this optimized crossover process is provided in [4].
- **Mutation:** In this case, random positions in the string are flipped with a predefined mutation probability. Care must be taken to ensure that the dimensionality of the projection does not change after the flipping process.

At termination, the algorithm is followed by a postprocessing phase. In the postprocessing phase, all data points containing the abnormal projections are reported by the algorithm as the outliers. The approach also provides the relevant projections which provide the *causality* for the outlier behavior of a data point. Thus, this approach has a high degree of interpretability.

### 5.2.2 Finding Distance-Based Outlying Subspaces

After the initial proposal of the basic subspace outlier detection framework [4], one of the earliest methods along this line was the *HOS-Miner* approach. Several different aspects of the broader ideas associated with *HOS-Miner* are discussed in [605, 606, 607]. A first discussion of the *HOS-Miner* approach was presented in [605]. According to this work, the definition of the outlying subspace for a given data point  $\bar{X}$  is as follows:

**Definition 5.2.1** *For a given data point  $\bar{X}$ , determine the set of subspaces such that the sum of its  $k$ -nearest neighbor distances in that subspace is at least  $\delta$ .*

This approach does not normalize the distances with the number of dimensions. Therefore, a subspace becomes more likely to be outlying with increasing dimensionality. This definition also exhibits closure properties in which any subspace of a non-outlying subspace is also not outlying. Similarly, every superset of an outlying subspace is also outlying. Clearly, only *minimal* subspaces that are outliers are interesting. The method in [605] uses these closure properties to prune irrelevant or uninteresting subspaces. Although the aforementioned definition has desirable closure properties, the use of a fixed threshold  $\delta$  across subspaces of different dimensionalities seems unreasonable. Selecting a definition based on algorithmic convenience can often cause poor results. As illustrated by the earlier example of the young Alzheimer patient, true outliers are often hidden in subspaces of the data, which cannot be inferred from their lower- or higher-dimensional projections.

An X-Tree is used in order to perform the indexing for performing the  $k$ -nearest neighbor queries in different subspaces efficiently. In order to further improve the efficiency of the learning process, the work in [605] uses a random sample of the data in order to learn about the subspaces before starting the subspace exploration process. This is achieved by estimating a quantity called the *Total Savings Factor (TSF)* of the outlying subspaces. These are used to regulate the search process for specific query points and prune the different subspaces in an ordered way. Furthermore, the TSF values of different subspaces are dynamically updated as the search proceeds. It has been shown in [605] that such an approach can be used in order to determine the outlying subspaces of specific data points efficiently. Numerous methods for using different kinds of pruning properties and genetic algorithms for finding outlying subspaces are presented in [606, 607].

### 5.2.3 Feature Bagging: A Subspace Sampling Perspective

The simplest method for combining outliers from multiple subspaces is the use of *feature bagging* [344], which is an ensemble method. Each base component of the ensemble uses the following steps:

- Randomly select an integer  $r$  from  $\lfloor d/2 \rfloor$  to  $(d - 1)$ .
- Randomly select  $r$  features (without replacement) from the underlying data set in iteration  $t$  in order to create an  $r$ -dimensional data set  $D_t$  in the  $t$ th iteration.
- Apply the outlier detection algorithm  $O_t$  on the data set  $D_t$  in order to compute the score of each data point.

In principle, one could use a different outlier detection algorithm in each iteration, provided that the scores are normalized to Z-values after the process. The normalization is also necessary to account for the fact that different subspace samples contain a different number of features. However, the work in [344] uses the LOF algorithm for all the iterations. Since the

LOF algorithm returns inherently normalized scores, such a normalization is not necessary. At the end of the process, the outlier scores from the different algorithms are combined in one of two possible ways:

- *Breadth-first approach*: In this approach, the ranking of the algorithms is used for combination purposes. The top-ranked outliers over all the different executions are ranked first, followed by the second-ranked outliers (with repetitions removed), and so on. Minor variations could exist because of tie-breaking between the outliers within a particular rank.
- *Cumulative-sum approach*: The outlier scores over the different algorithm executions are summed up. The top ranked outliers are reported on this basis. One can also view this process as equivalent to the averaging combination function in an ensemble method (cf. Chapter 6).

It was experimentally shown in [344] that such methods are able to ameliorate the effects of irrelevant attributes. In such cases, full-dimensional algorithms are unable to distinguish the true outliers from the normal data, because of the additional noise.

At first sight, it would seem that random subspace sampling [344] does not attempt to optimize the discovery of relevant subspaces at all. Nevertheless, it does have the paradoxical merit that it is relatively efficient to sample subspaces, and therefore a large number of subspaces can be sampled in order to improve robustness. Even though each detector selects a global subspace, the *ensemble-based combined score* of a given point is able to implicitly benefit from the locally-optimized subspaces. This is because different points may obtain favorable scores in different subspace samples, and the ensemble combination is often able to identify all the points that are favored in a sufficient number of the subspaces. This phenomenon can also be formally explained in terms of the notion of how ensemble methods reduce representational bias [170] (cf. section 6.4.3.1 of Chapter 6). In other words, ensemble methods provide an implicit route for converting global subspace exploration into local subspace selection and are therefore inherently more powerful than their individual components. An ensemble-centric perspective on feature bagging is provided in section 6.4.3.1.

The robustness resulting from multiple subspace sampling is clearly a very desirable quality, as long as the combination function at the end recognizes the differential behavior of different subspace samples for a given data point. In a sense, this approach implicitly recognizes the difficulty of detecting relevant and rare subspaces, and therefore samples as many subspaces as possible in order to reveal the rare behavior. From a conceptual perspective, this approach is similar to that of harnessing the power of many weak learners to create a single strong learner in classification problems. The approach has been shown to show consistent performance improvement over full-dimensional methods for many real data sets in [344]. This approach may also be referred to as the *feature bagging method* or *random subspace ensemble method*. Even though the original work [344] uses LOF as the base detector, the average  $k$ -nearest neighbor detector has also been shown to work [32].

## 5.2.4 Projected Clustering Ensembles

Projected clustering methods define clusters as sets of points together with sets of dimensions in which these points cluster well. As discussed in Chapter 4, clustering and outlier detection are complementary problems. Therefore, it is natural to investigate whether projected or subspace clustering methods can also be used for outlier detection. Although the

relevant subspaces for clusters are not always relevant for outlier detection, there is still a weak relationship between the two. By using ensembles, it is possible to strengthen the types of outliers discovered using this approach.

As shown in the *OutRank* work [406], one can use ensembles of projected clustering algorithms [5] for subspace outlier detection. In this light, it has been emphasized in [406] that the use of *multiple* projected clusterings is essential because the use of a single projected clustering algorithm provides very poor results. The basic idea in *OutRank* is to use the following procedure repeatedly:

- Use a randomized projected clustering method like PROCLUS [5] on the data set to create a set of projected clusters.
- Quantify the outlier score of each point based on its similarity to the cluster to which it belongs. Examples of relevant scores include the size, dimensionality, (projected) distance to cluster centroid, or a combination of these factors. The proper choice of measure is sensitive to the specific clustering algorithm that is used.

This process is applied repeatedly, and the scores are averaged in order to yield the final result. The use of a sufficiently randomized clustering method in the first step is crucial for obtaining good results with this ensemble-centric approach.

There are several variations one might use for the scoring step. For a distance-based algorithm like PROCLUS, it makes sense to use the same distance measure to quantify the outlier score as was used for clustering. For example, one can use the Manhattan segmental distance of a data point from its nearest cluster centroid in the case of PROCLUS. The Manhattan segmental distance is estimated by first computing the Manhattan distance of the point to the centroid of the cluster in its relevant subspace and then dividing by the number of dimensions in that subspace. However, this measure ignores the number of points and dimensionality of the clusters; furthermore, it is not applicable for pattern-based methods with overlapping clusters. A natural approach is the *individual weighting* measure. For a point, the fraction of the number of points in its cluster to maximum cluster size is computed, and the fraction of the number of dimensions in its cluster subspace to the maximum cluster-subspace dimensionality is also computed. A simple outlier score is to add these two fractions over all clusters in which the point occurs and then divide by the total number of clusters in the data. A point that is included in many large and high-dimensional subspace clusters is unlikely to be an outlier. Therefore, points with smaller scores are deemed as outliers. A similar method for binary and categorical data is discussed in section 8.5.1 of Chapter 8. Two other measures, referred to as cluster coverage and subspace similarity, are proposed in [406]. The work in [406] experimented with a number of different clustering algorithms and found that using multiple randomized runs of PROCLUS yields the best results. This variation is referred to as *Multiple-Proclus*. The cluster coverage and subspace similarity measures were used to achieve these results, although reasonable results were also achieved with the individual weighting measure.

The key point to understand about such clustering-based methods is that the type of outlier discovered is sensitive to the underlying clustering, although an ensemble-centric approach is essential for success. Therefore, a locality-sensitive clustering ensemble will make the outliers locality-sensitive; a subspace-sensitive clustering ensemble will make the outliers subspace-sensitive, and a correlation-sensitive clustering ensemble will make the outliers correlation-sensitive.



### 5.2.5 Subspace Histograms in Linear Time

A linear-time implementation of subspace histograms with hashing is provided in [476] and is referred to as *RS-Hash*. The basic idea is to repeatedly construct grid-based histograms on data samples of size  $s$  and combine the scores in an ensemble-centric approach. Each histogram is constructed on a randomly chosen subspace of the data. The dimensionality of the subspace and size of the grid region is specific to its ensemble component. In the testing phase of the ensemble component, all  $N$  points in the data are scored based on the logarithm of the number of points (from the training sample) in its grid region. The approach is repeated with multiple samples of size  $s$ . The point-specific scores are averaged over different ensemble components to create a final result, which is highly robust. The variation in the dimensionality and size of the grid regions is controlled with an integer dimensionality parameter  $r$  and a fractional grid-size parameter  $f \in (0, 1)$ , which vary randomly over different ensemble components. We will describe the process of random selection of these parameters later. For now, we assume (for simplicity) that the values of these parameters are fixed.

The sample  $S$  of size  $s \ll N$  may be viewed as the training data of a single ensemble component, and each of the  $N$  points is scored in this component by constructing a subspace histogram on this training sample. First, a set  $V$  of  $r$  dimensions is randomly sampled from the  $d$  dimensions, and all the scoring is done on histograms built in this  $r$ -dimensional subspace. The minimum value  $\min_j$  and the maximum value  $\max_j$  of the  $j$ th dimension are determined from this sample. Let  $x_{ij}$  denote the  $j$ th dimension of the  $i$ th point. All  $s \cdot r$  values  $x_{ij}$  in the training sample, such that  $j \in V$ , are normalized as follows:

$$x'_{ij} \leftarrow \frac{x_{ij} - \min_j}{\max_j - \min_j} \quad (5.2)$$

One can even use  $x'_{ij} \leftarrow x_{ij}/(\max_j - \min_j)$  for implementation simplicity. At the time of normalization, we also create the following  $r$ -dimensional *discretized* representation of the training points, where for each of the  $r$  dimensions in  $V$ , we use a grid-size of width  $f \in (0, 1)$ . Furthermore, to induce diversity across ensemble components, the placement of the grid partitioning points is varied across ensemble components. In a given ensemble component, a grid partitioning point is not placed at 0, but at a value of  $-\alpha_j$  for dimension  $j$ . This can be achieved by setting the discretized identifier of point  $i$  and dimension  $j$  to  $\lfloor (x'_{ij} + \alpha_j)/f \rfloor$ . The value of  $\alpha_j$  is fixed within an ensemble component up front at a value randomly chosen from  $(0, f)$ . This  $r$ -dimensional discretized representation provides the identity of the  $r$ -dimensional bounding box of that point. A hash table maintains a count of each of the bounding boxes encountered in the training sample. For each of the  $s$  training points, the count of its bounding box is incremented by hashing this  $r$ -dimensional discrete representation, which requires constant time. In the testing phase, the discretized representation of each of the  $N$  points is again constructed using the aforementioned process, and its count is retrieved from the hash table constructed on the training sample. Let  $n_i \leq s$  denote this count of the  $i$ th point. For points included in the training sample  $S$ , the outlier score of the  $i$ th point is  $\log_2(n_i)$ , whereas for points not included in the training sample  $S$ , the score is  $\log_2(n_i + 1)$ . This process is repeated over multiple ensemble components (typically 100), and the average score of each point is returned. Low scores represent outliers.

It is noteworthy that the values of  $\min_j$  and  $\max_j$  used in the testing phase of an ensemble component are the same as those estimated from the training sample. Thus, the training phase requires only  $O(s)$  time, and the value of  $s$  is typically a small constant such as 1000. The testing phase of each ensemble component requires  $O(N)$  time, and the overall

algorithm requires  $O(N + s)$  time. The constant factors also tend to be very small and the approach is extremely fast.

We now describe the process of randomly selecting  $f$ ,  $r$ , and  $\alpha_j$  up front in each ensemble component. The value of  $f$  is selected uniformly at random from  $(1/\sqrt{s}, 1 - 1/\sqrt{s})$ . Subsequently, the value of  $r$  is set uniformly at random to an integer between  $1 + 0.5 \cdot \lceil \log_{\max\{2, 1/f\}}(s) \rceil$  and  $\log_{\max\{2, 1/f\}}(s)$ . The value of each  $\alpha_j$  is selected uniformly at random from  $(0, f)$ . This variation of grid size (with  $f$ ), dimensionality (with  $r$ ), and placement of grid regions (with  $\alpha_j$ ) provides additional diversity, which is helpful to the overall ensemble result. The work in [476] has also shown how the approach may be extended to data streams. This variant is referred to as *RS-Stream*. The streaming variant requires greater sophistication in the hash-table design and maintenance, although the overall approach is quite similar.

### 5.2.6 Isolation Forests

The work in [367] proposes a model called *isolation forests*, which shares some intuitive similarity with another ensemble technique known as *random forests*. Random forests are among the most successful models used in classification and are known to outperform the majority of classifiers in a variety of problem domains [195]. However, the unsupervised way in which an isolation forest is constructed is quite different, especially in terms of how a data point is scored. As discussed in section 5.2.6.3, the isolation forest is also a special case of the *extremely randomized clustering forest (ERC-Forest)* for clustering [401].

An isolation forest is an ensemble combination of a set of *isolation trees*. In an isolation tree, the data is recursively partitioned with axis-parallel cuts at randomly chosen partition points in randomly selected attributes, so as to isolate the instances into nodes with fewer and fewer instances until the points are isolated into singleton nodes containing one instance. In such cases, the tree branches containing outliers are noticeably less deep, because these data points are located in sparse regions. Therefore, the distance of the leaf to the root is used as the outlier score. The final combination step is performed by averaging the path lengths of the data points in the different trees of the isolation forest.

Isolation forests are intimately related to subspace outlier detection. The different branches correspond to different local subspace regions of the data, depending on how the attributes are selected for splitting purposes. The smaller paths correspond to lower dimensionality<sup>2</sup> of the subspaces in which the outliers have been isolated. The less the dimensionality that is required to isolate a point, the stronger the outlier that point is likely to be. In other words, isolation forests work under the implicit assumption that it is more likely to be able to isolate outliers in subspaces of lower dimensionality created by random splits. For instance, in our earlier example on Alzheimer’s patients, a *short* sequence of splits such as  $Age \leq 30$ ,  $Alzheimer = 1$  is likely to isolate a rare individual with early-onset Alzheimer’s disease.

The training phase of an isolation forest constructs multiple isolation trees, which are unsupervised equivalents of decision trees. Each tree is binary, and has at most  $N$  leaf nodes for a data set containing  $N$  points. This is because each leaf node contains exactly one data point by default, but early termination is possible by parameterizing the approach with

---

<sup>2</sup>Although it is possible to repeatedly cut along the same dimension, this becomes less likely in higher-dimensional data sets. In general, the path length is highly correlated with the dimensionality of the subspace used for isolation. For a data set containing  $2^8 = 256$  points (which is the recommended subsample size), the average depth of the tree will be 8, but an outlier might often be isolated in less than three or four splits (dimensions).

a height parameter. In order to construct the isolation tree from a data set containing  $N$  points, the approach creates a root node containing all the points. This is the initial state of the isolation tree  $\mathcal{T}$ . A candidate list  $\mathcal{C}$  (for further splitting) of nodes is initialized as a singleton list containing the root node. Then, the following steps are repeated to create the isolation tree  $\mathcal{T}$  until the candidate list  $\mathcal{C}$  is empty:

1. Select a node  $R$  from  $\mathcal{C}$  randomly and remove from  $\mathcal{C}$ .
2. Select a random attribute  $i$  and split the data in  $R$  into two sets  $R_1$  and  $R_2$  at a random value  $a$  chosen along that attribute. Therefore, all data points in  $R_1$  satisfy  $x_i \leq a$  and all data points in  $R_2$  satisfy  $x_i > a$ . The random value  $a$  is chosen uniformly at random between the minimum and maximum values of the  $i$ th attribute among data points in node  $R$ . The nodes  $R_1$  and  $R_2$  are children of  $R$  in  $\mathcal{T}$ .
3. **(Step performed for each  $i \in \{1, 2\}$ ):** If  $R_i$  contains more than one point then add it to  $\mathcal{C}$ . Otherwise, designate the node as an isolation-tree leaf.

This process will result in the creation of a binary tree that is typically not balanced. Outlier nodes will tend to be isolated more quickly than non-outlier nodes. For example, a point that is located very far away from the remaining data along all dimensions would likely be separated as an isolated leaf in the very first iteration. Therefore, the length of the path from the root to the leaf is used as the outlier score. Note that the path from the root to the leaf defines a subspace of varying dimensionality. Outliers can typically be isolated in much lower-dimensional subspaces than normal points. Therefore, the number of edges from the root to a node is equal to its outlier score. Smaller scores correspond to outliers. This inherently randomized approach is repeated multiple times and the scores are averaged to create the final result. The ensemble-like approach is particularly effective and it often provides results of high quality. The basic version of the isolation tree, when grown to full height, is *parameter-free*. This characteristic is always a significant advantage in unsupervised problems like outlier detection. The *average-case* computational complexity of the approach is  $\theta(N \log(N))$ , and the space complexity is  $O(N)$  for each isolation tree.

One can always improve computational efficiency and often improve accuracy with subsampling. The subsampling approach induces further diversity, and gains some of the advantages inherent in outlier ensembles [31]. In this case, the isolation tree is constructed using a subsample in the *training phase*, although all points are scored against the subsample in the *testing phase*. The training phase returns the tree structure together with the split conditions (such as  $x_i \leq a$  and  $x_i > a$ ) at each node. Out-of-sample points are scored in the testing phase by using the split conditions computed during the training phase. Much like the testing phase of a decision tree, the appropriate leaf node for an out-of-sample point is identified by traversing the appropriate path from the root to the leaf with the use of the split conditions, which are simple univariate inequalities.

The use of a subsample results in better computational efficiency and better diversity. It is stated in [367] that a subsample size of 256 works well in practice, although this value might vary somewhat with the data set at hand. Note that this results in *constant* computational and memory requirements for building the tree, irrespective of data set size. The testing phase requires average-case complexity of  $\theta(\log(N))$  for each data point if the entire data set is used. On the other hand, the testing phase requires constant time for *each point*, if subsamples of size 256 are used. Therefore, if subsampling is used with a constant number of samples and a constant number of trials, the running time is constant for the training phase,  $O(N)$  for the testing phase, and the space complexity is constant as well.

The isolation forest is an efficient method, which is noteworthy considering the fact that most subspace methods are computationally intensive.

The final combination step is performed by averaging the path lengths of a data point in the different trees of the isolation forest. A detailed discussion of this approach from an ensemble-centric point of view is provided in section 6.4.5 of Chapter 6. Practical implementations of this approach are available on the Python library *scikit-learn* [630] and R library *SourceForge* [631].

### 5.2.6.1 Further Enhancements for Subspace Selection

The approach is enhanced by additional pre-selection of feature with a *Kurtosis measure*. The Kurtosis of a set of feature values  $x_1 \dots x_N$  is computed by first standardizing them to  $z_1 \dots z_N$  with zero mean and unit standard deviation:

$$z_i = \frac{x_i - \mu}{\sigma} \quad (5.3)$$

Here,  $\mu$  is the mean and  $\sigma$  is the standard deviation of  $x_1 \dots x_N$ . Then, the Kurtosis is computed as follows:

$$K(z_1 \dots z_N) = \frac{\sum_{i=1}^N z_i^4}{N} \quad (5.4)$$

Features that are very non-uniform will show a high level of Kurtosis. Therefore, the Kurtosis computation can be viewed as a feature selection measure for anomaly detection. The work in [367] *preselects* a subset of attributes based on the ranking of their univariate Kurtosis values and then constructs the random forest after (globally) throwing away those features. Note that this results in global subspace selection; nevertheless the random split approach is still able to explore different local subspaces, albeit randomly (like feature bagging). A generalization of the Kurtosis measure, referred to as *multidimensional Kurtosis*, is discussed in section 1.3.1 of Chapter 1. This measure evaluates subsets of features jointly using Equation 5.4 on the Mahalanobis distances of points in that subspace rather than ranking features with univariate Kurtosis values. This measure is generally considered to be more effective than univariate Kurtosis but it is computationally expensive because it needs to be coupled with feature subset exploration in a structured way. Although the generalized Kurtosis measure has not been used in the isolation forest framework, it has the potential for application within settings in which computational complexity is not as much of a concern.

### 5.2.6.2 Early Termination

A further enhancement is that the tree is not grown to full height. The growth of a node is stopped as soon as a node contains either duplicate instances or it exceeds a certain threshold height. This threshold height is set to 10 in the experiments of [367]. In order to estimate the path length of points in such nodes, an additional credit needs to be assigned to account for the fact that the points in these nodes have not been materialized to full height. For a node containing  $r$  instances, its additional credit  $c(r)$  is defined as the expected path length in a binary search tree with  $r$  points [442]:

$$c(r) = \ln(r - 1) - \frac{2(r - 1)}{r} + 0.5772 \quad (5.5)$$

Note that this credit is added to the path length of that node from the root to compute the final outlier score. Early termination is an efficiency-centric enhancement, and one can choose to grow the trees to full length if needed.

### 5.2.6.3 Relationship to Clustering Ensembles and Histograms

The isolation forest can be viewed as a type of clustering ensemble as discussed in section 5.2.4. The isolation tree creates hierarchical projected clusters from the data, in which clusters are defined by their bounding boxes. A bounding box of a cluster (node) is defined by the sequence of axis-parallel splits from the root to that node. However, compared to most projected clustering methods, the isolation tree is *extremely* randomized because of its focus on ensemble-centric performance. The isolation forest is a decision-tree-based approach to clustering. Interestingly, the basic isolation forest can be shown to be a variation of an earlier clustering ensemble method, referred to as *extremely randomized clustering forests (ERC-Forests)* [401]. The main difference is that the ERC-Forest uses multiple trials at each node to enable a small amount of supervision with class labels; however, by setting the number of trials to 1 and growing the tree to full length, one can obtain an unsupervised isolation tree as a special case. Because of the *space*-partitioning (rather than *point*-partitioning) methodology used in ERC-Forests and isolation forests, these methods also share some intuitive similarities with histogram- and density-based methods. An isolation tree creates hierarchical and randomized grid regions, whose expected volume reduces by a factor of 2 with each split. The path length in an isolation tree is therefore a rough surrogate for the negative logarithm of the (fractional) volume of a maximal grid region containing a single data point. This is similar to the notion of log-likelihood density used in traditional histograms. Unlike traditional histograms, isolation forests are not parameterized by grid-width and are therefore more flexible in handling data distributions of varying density. Furthermore, the flexible shapes of the histograms in the latter naturally define local subspace regions.

The measures for scoring the outliers with projected clustering methods [406] also share some intuitive similarities with isolation forests. One of the measures in [406] uses the sum of the subspace dimensionality of the cluster and the number of points in the cluster as an outlier score. Note that the subspace dimensionality of a cluster is a rough proxy for the path length in the isolation tree. Similarly, some variations of the isolation tree, referred to as half-space trees [532], use fixed-height trees. In these cases, the number of points in the relevant node for a point is used to define its outlier score. This is similar to clustering-based outlier detection methods in which the number of points in the nearest cluster is often used as an important component of the outlier score.

## 5.2.7 Selecting High-Contrast Subspaces

The feature bagging method [344] discussed in section 5.2.3 randomly samples subspaces. If many dimensions are irrelevant, at least a few of them are likely to be included in each subspace sample. At the same time, information is lost because many dimensions are dropped. These effects are detrimental to the accuracy of the approach. Therefore, it is natural to ask whether it is possible to perform a pre-processing in which a smaller number of *high-contrast* subspaces are selected. This method is also referred to as *HiCS*, as it selects high-contrast subspaces.

In the work proposed in [308], the outliers are found only in these high-contrast subspaces, and the corresponding scores are combined. Thus, this approach decouples the sub-

space search as a generalized pre-processing approach from the outlier ranking of the individual data points. The approach discussed in [308] is quite interesting because of its pre-processing approach to finding relevant subspaces in order to reduce the irrelevant subspace exploration. Although the high-contrast subspaces are obtained using aggregation-based statistics, these statistics are only used as hints in order to identify multiple subspaces for greater robustness. The assumption here is that rare patterns are *statistically more likely* to occur in subspaces where there is significant non-uniformity and contrast. The final outlier score combines the results over different subspaces to ensure that at least a few relevant subspaces will be selected. The insight in the work of [308] is to combine *discriminative* subspace selection with the score aggregation of feature bagging in order to determine the relevant outlier scores. Therefore, the only difference from feature bagging is in how the subspaces are selected; the algorithms are otherwise identical. It has been shown in [308] that this approach performs better than the feature bagging method. Therefore, an overview of the *HiCS* method is as follows:

1. The first step is to select discriminative subspaces using an *Apriori*-like [37] exploration, which is described at the end of this section. These are *high-contrast subspaces*. Furthermore, the subspaces are also pruned to account for redundancy among them.
  - An important part of this exploration is to be able to *evaluate* the quality of the candidate subspaces during exploration. This is achieved by quantifying the contrast of a subspace.
2. Once the subspaces have been identified, an exactly similar approach to the feature bagging method is used. The LOF algorithm is executed after projecting the data into these subspaces and the scores are combined as discussed in section 5.2.3.

Our description below will therefore focus only on the first step of *Apriori*-like exploration and the corresponding quantification of the contrast. We will deviate from the natural order of presentation and first describe the contrast computation because it is germane to a proper understanding of the *Apriori*-like subspace exploration process.

Consider a subspace of dimensionality  $p$ , in which the dimensions are indexed as  $\{1 \dots p\}$  (without loss of generality). The conditional probability  $P(x_1|x_2 \dots x_p)$  for an attribute value  $x_1$  is the same as its unconditional probability  $P(x_1)$  for the case of uncorrelated data. High-contrast subspaces are likely to violate this assumption because of non-uniformity in data distribution. In our earlier example of the young Alzheimer patients, this corresponds to the unexpected rarity of the *combination* of youth and the disease. In other words  $P(\text{Alzheimer} = 1)$  is likely to be very different from  $P(\text{Alzheimer} = 1 | \text{Age} \leq 30)$ . The idea is that subspaces with such unexpected non-uniformity are more *likely* to contain outliers, although it is treated only as a weak hint for pre-selection of one of multiple subspaces. The approach in [308] generates candidate subspaces using an *Apriori*-like approach [37] described later in this section. For each candidate subspace of dimensionality  $p$  (which might vary during the *Apriori*-like exploration), it repeatedly draws pairs of “samples” from the data in order to estimate  $P(x_i)$  and  $P(x_i|x_1 \dots x_{i-1}, x_{i+1} \dots x_p)$  and test whether they are different. A “sample” is defined by (i) the selection of a particular attribute  $i$  from  $\{1 \dots p\}$  for testing, and (ii) the construction of a random rectangular region in  $p$ -dimensional space for testing. Because of the construction of a rectangular region for testing, each  $x_i$  refers to a 1-dimensional range of values (e.g.,  $\text{Age} \in (10, 20)$ ) in the  $i$ th dimension. The values of  $P(x_i)$  and  $P(x_i|x_1 \dots x_{i-1}, x_{i+1} \dots x_p)$  are computed in this random rectangular region. After  $M$  pairs of samples of  $P(x_i)$  and  $P(x_i|x_1 \dots x_{i-1}, x_{i+1} \dots x_p)$  have been drawn, it is

determined whether the independence assumption is violated with hypothesis testing. A variety of tests based on the Student's  $t$ -distribution can be used to measure the deviation of a subspace from the basic hypothesis of independence. This provides a measure of the non-uniformity of the subspace and therefore provides a way to measure the quality of the subspaces in terms of their propensity to contain outliers.

A bottom-up *Apriori*-like [37] approach is used to identify the relevant projections. In this bottom-up approach, the subspaces are continuously extended to higher dimensions for non-uniformity testing. Like *Apriori*, only subspaces that have sufficient contrast are extended for non-uniformity testing as potential candidates. The non-uniformity testing is performed as follows. For each candidate subspace of dimensionality  $p$ , a random rectangular region is generated in  $p$  dimensions. The width of the random range along each dimension is selected so that the 1-dimensional range contains  $N \cdot \alpha^{(1/p)}$  points, where  $\alpha < 1$ . Therefore, the entire  $p$ -dimensional region is expected to contain  $N \cdot \alpha$  points. The  $i$ th dimension is used for hypothesis testing, where the value of  $i$  is chosen at random from  $\{1 \dots p\}$ . One can view the index  $i$  as the test dimension. Let the set of points in the intersection of the ranges along the remaining  $(p - 1)$  dimensions be denoted by  $S_i$ . The fraction of points in  $S_i$  that lies within the upper and lower bounds of the range of dimension  $i$  provides an estimate of  $P(x_i | x_1 \dots x_{i-1}, x_{i+1} \dots x_d)$ . The statistically normalized deviation of this value from the unconditional value of  $P(x_i)$  is computed using hypothesis testing and it provides a deviation estimate for that subspace. The process is repeated multiple times over different random slices and test dimensions; then, the deviation values over different tests are averaged. Subspaces with large deviations are identified as high-contrast subspaces. At the end of the *Apriori*-like phase, an additional pruning step is applied to remove redundant subspaces. A subspace of dimensionality  $p$  is removed, if another subspace of dimensionality  $(p + 1)$  exists (among the reported subspaces) with higher contrast.

The approach decouples subspace identification from outlier detection and therefore all the relevant subspaces are identified up front as a preprocessing step. After the subspaces have been identified, the points are scored using the LOF algorithm in each such subspace. Note that this step is very similar to feature bagging, except that we are restricting ourselves to more carefully chosen subspaces. Then, the scores of each point across various subspaces are computed and averaged to provide a unified score of each data point. In principle, other combination functions like maximization can be used. Therefore, one can adapt any of the combination methods used in feature bagging. More details of the algorithm for selecting relevant subspaces are available in [308].

The *HiCS* technique is notable for the intuitive idea that statistical selection of relevant subspaces is more effective than choosing random subspaces. The main challenge is in discovering the high-contrast subspaces, because it is computationally intensive to use an *Apriori*-like algorithm in combination with sample-based hypothesis testing. Many straightforward alternatives exist for finding high-contrast subspaces, which might be worth exploring. For example, one can use the multidimensional Kurtosis measure discussed in section 1.3.1 of Chapter 1 in order to test the relevance of a subspace for high-dimensional outlier detection. This measure is simple to compute and also takes the interactions between the dimensions into account because of its use of the Mahalanobis distance.

## 5.2.8 Local Selection of Subspace Projections

The work in [402] uses *local* statistical selection of relevant subspace projections in order to identify outliers. In other words, the selection of the subspace projections is optimized to specific data points, and therefore the locality of a given data point matters in the

selection process. For each data point  $\bar{X}$ , a set of subspaces is identified, which are considered *high-contrast* subspaces from the perspective of outlier detection. However, this exploration process uses the high-contrast behavior as statistical *hints* in order to explore *multiple* subspaces for robustness, since a single subspace may be unable to completely capture the outlierness of the data point.

The *OUTRES* method [402] examines the density of lower-dimensional subspaces in order to identify relevant projections. The basic hypothesis is that for a given data point  $\bar{X}$ , it is desirable to determine subspaces in which the data is sufficiently non-uniformly distributed in its locality. In order to characterize the distribution of the locality of a data point, the work in [402] computes the local density of data point  $\bar{X}$  in subspace  $S$  as follows:

$$\text{den}(S, \bar{X}) = |\mathcal{N}(\bar{X}, S)| = |\{\bar{Y} : \text{dist}_S(\bar{X}, \bar{Y}) \leq \epsilon\}| \quad (5.6)$$

Here,  $\text{dist}_S(\bar{X}, \bar{Y})$  represents the Euclidean distance between data point  $\bar{X}$  and  $\bar{Y}$  in subspace  $S$ . This is the simplest possible definition of the density, although other more sophisticated methods such as kernel density estimation [496] are used in *OUTRES* in order to obtain more refined results. Kernel density estimation is also discussed in Chapter 4. A major challenge here is in comparing the subspaces of varying dimensionality. This is because the density of the underlying subspaces reduces with increasing dimensionality. It has been shown in [402], that it is possible to obtain comparable density estimates across subspaces of different dimensionalities by selecting the bandwidth of the density estimation process according to the dimensionality of the subspace.

Furthermore, the work in [402] uses statistical techniques in order to meaningfully compare different subspaces. For example, if the data is uniformly distributed, the number of data points lying within a distance  $\epsilon$  of the data point should be regulated by the fractional volume of the data in that subspace. Specifically, the fractional parameter defines a binomial distribution characterizing the number of points in that volume, if that data were to be uniformly distributed. Of course, one is really interested in subspaces that deviate significantly from this behavior. The (local) relevance of the subspace for a particular data point  $\bar{X}$  is computed using statistical testing. The two hypotheses are as follows:

- Hypothesis  $H_0$ : The local subspace neighborhood  $\mathcal{N}(\bar{X}, S)$  is uniformly distributed.
- Hypothesis  $H_1$ : The local subspace neighborhood  $\mathcal{N}(\bar{X}, S)$  is not uniformly distributed.

The Kolmogorov-Smirnoff goodness-of-fit test [512] is used to determine which of the aforementioned hypotheses is true. It is important to note that this process provides an idea of the *usefulness* of a subspace, and is used in order to enable a *filtering condition* for removing irrelevant subspaces from the process of computing the outlier score of a specific data point. A subspace is defined as relevant, if it passes the hypothesis condition  $H_1$ . In other words, outlier scores are computed using a combination of subspaces which *must* satisfy this relevance criterion. This test is combined with an ordered subspace exploration process in order to determine the relevant subspaces  $S_1 \dots S_k$ . This exploration process will be described later in detail (cf. Figure 5.2).

In order to combine the point-wise scores from multiple *relevant* subspaces, the work in [402] uses the product of the outlier scores obtained from different subspaces. Thus, if  $S_1 \dots S_k$  are the different abnormal subspaces found for data point  $\bar{X}$ , and if  $O(S_i, \bar{X})$  is its outlier score in subspace  $S_i$ , then the overall outlier score  $OS(\bar{X})$  is defined as follows:

$$OS(\bar{X}) = \prod_i O(S_i, \bar{X}) \quad (5.7)$$



**Algorithm** *OUTRES*(Data Point:  $\bar{X}$ , Subspace:  $S$ );  
**begin**  
  **for** each attribute  $i$  not in  $S$  **do**  
    **if**  $S_i = S \cup \{i\}$  passes Kolmogorov-Smirnoff non-uniformity test **then**  
      **begin**  
        Compute  $den(S_i, \bar{X})$  using Equation 5.6 or kernel density estimation;  
        Compute  $dev(S_i, \bar{X})$  using Equation 5.8;  
        Compute  $O(S_i, \bar{X})$  using Equation 5.9;  
         $OS(\bar{X}) = O(S_i, \bar{X}) \cdot OS(\bar{X})$ ;  
         $OUTRES(\bar{X}, S_i)$ ;  
      **end**  
    **end**  
**end**

Figure 5.2: The *OUTRES* algorithm

The details of the computation of  $O(S_i, \bar{X})$  will be provided later, although the basic assumption is that *low scores* represent a greater tendency to be an outlier. The advantage of using the product over the sum in this setting is that the latter is dominated by the high scores, as a result of which a few subspaces containing normal behavior will dominate the sum. On the other hand, in the case of the product, the outlier behavior in a small number of subspaces will be greatly magnified. This is particularly appropriate for the problem of outlier detection. It is noteworthy that the product-wise combination can also be viewed as the sum of the logarithms of the scores.

In order to define the outlier score  $O(S_i, \bar{X})$ , a subspace is considered significant for particular objects only if its density is at least two standard deviations less than the mean value. This is essentially a condition for that subspace to be considered deviant. Thus, the deviation  $dev(S_i, \bar{X})$  of the data point  $\bar{X}$  in subspace  $S_i$  is defined as the ratio of the deviation of the density of the object from the mean density in the neighborhood of  $\bar{X}$ , divided by two standard deviations.

$$dev(S_i, \bar{X}) = \frac{\mu - den(S_i, \bar{X})}{2 \cdot \sigma} \quad (5.8)$$

The values of  $\mu$  and  $\sigma$  are computed over data points in the neighborhood of  $\bar{X}$  and therefore this computation provides a *local* deviation value. Note that deviant subspaces will have  $dev(S_i, \bar{X}) > 1$ . The outlier score of a data point in a subspace is the ratio of the density of the point in the space to its deviation, if it is a deviant subspace. Otherwise the outlier score is considered to be 1, and it does not affect the overall outlier score in the product-wise function in Equation 5.7 for combining the scores of data point  $\bar{X}$  from different subspaces. Thus, the outlier score  $O(S_i, \bar{X})$  is defined as follows:

$$O(S_i, \bar{X}) = \begin{cases} \frac{den(S_i, \bar{X})}{dev(S_i, \bar{X})} & \text{if } dev(S_i, \bar{X}) > 1 \\ 1 & \text{otherwise} \end{cases} \quad (5.9)$$

The entire recursive approach of the *OUTRES* algorithm (cf. Figure 5.2) uses the data point  $\bar{X}$  as input, and therefore the procedure needs to be applied separately *for scoring each candidate data point*. In other words, this approach is inherently an instance-based method (like nearest-neighbor detectors), rather than one of the methods that selects subspaces

up front in a decoupled way (like feature bagging or *HiCS*). An observation in [402] is that subspaces that are either very low dimensional (e.g., 1-dimensional subspaces) or very high dimensional are not very informative for outlier detection. Informative subspaces can be identified by careful attribute-wise addition to candidate subspaces. In the recursive exploration, an additional attribute is included in the subspace for statistical testing. When an attribute is added to the current subspace  $S_i$ , the non-uniformity test is utilized to determine whether or not that subspace should be used. If it is not relevant, then the subspace is discarded. Otherwise, the outlier score  $O(S_i, \bar{X})$  in that subspace is computed for the data point, and the current value of the outlier score  $OS(\bar{X})$  is updated by multiplying  $O(S_i, \bar{X})$  with it. Since the outlier scores of subspaces that do not meet the filter condition are set to 1, they do not affect the density computation in this multiplicative approach. The procedure is then recursively called in order to explore the next subspace. Thus, such a procedure potentially explores an exponential number of subspaces, although the real number is likely to be quite modest because of pruning. In particular, the non-uniformity test prunes large parts of the recursion tree during the exploration. The overall algorithm for subspace exploration for a given data point  $\bar{X}$  is illustrated in Figure 5.2. Note that this pseudocode assumes that the overall outlier score  $OS(\bar{X})$  is like a global variable that can be accessed by all levels of the recursion and it is initialized to 1 before the first call of *OUTRES*. The initial call to *OUTRES* uses the empty subspace as the argument value for  $S$ .

### 5.2.9 Distance-Based Reference Sets

A distance-based method for finding outliers in lower-dimensional projections of the data was proposed in [327]. In this approach, instead of trying to find local subspaces of abnormally low density over the whole data, a local analysis is provided specific to each data point. For each data point  $\bar{X}$ , a *reference set* of points  $S(\bar{X})$  is identified. The reference set  $S(\bar{X})$  is generated as the top- $k$  closest points to the candidate with the use of shared nearest-neighbor distances [287] (see section 4.3.3).

After this reference set  $S(\bar{X})$  has been identified, the relevant subspace for  $S(\bar{X})$  is determined as the set  $Q(\bar{X})$  of dimensions in which the variance is small. The specific threshold on the variance is set to a user-defined fraction of the average dimension-specific variance of the points in  $S(\bar{X})$ . Thus, this approach analyzes the statistics of individual dimensions independently of one another during the crucial step of subspace selection. The Euclidean distance of  $\bar{X}$  is computed to the mean of the reference set  $S(\bar{X})$  *in the subspace defined by  $Q(\bar{X})$* . This is denoted by  $G(\bar{X})$ . The value of  $G(\bar{X})$  is affected by the number of dimensions in  $Q(\bar{X})$ . The *subspace outlier degree*  $SOD(\bar{X})$  of a data point is defined by normalizing this distance  $G(\bar{X})$  by the number of dimensions in  $Q(\bar{X})$ :

$$SOD(\bar{X}) = \frac{G(\bar{X})}{|Q(\bar{X})|}$$

The approach of using the variance of *individual* dimensions for selecting the subspace set  $Q(\bar{X})$  is a rather naive generalization derived from subspace clustering methods, and is a rather questionable design choice. This is because the approach completely ignores the interactions among various dimensions. In many cases, such as the example of the young Alzheimer patient discussed earlier, the unusual behavior is manifested in *the violations of dependencies among* dimensions rather than the variances of the individual dimensions. Variances of individual dimensions tell us little about the dependencies among them. Many

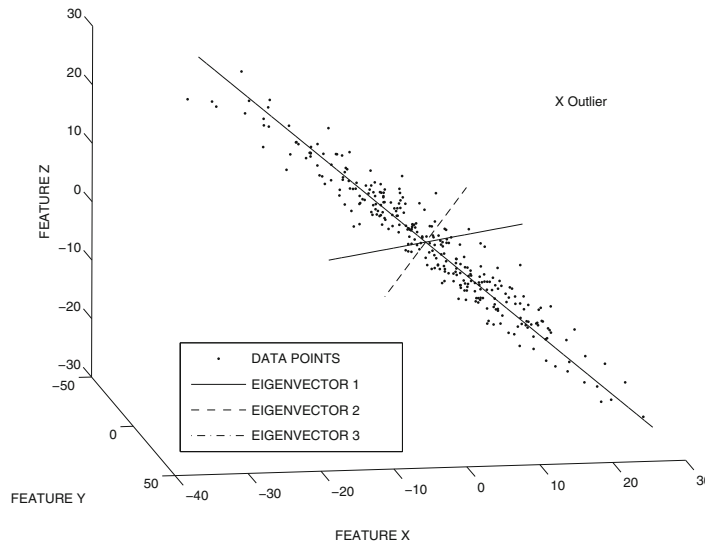


Figure 5.3: The example of Figure 3.4 re-visited: Global PCA can discover outliers in cases, where the entire data is aligned along lower dimensional manifolds.

other insightful techniques like *HiCS* [308, 402], which use biased subspace selection, almost always use the dependencies among dimensions as a key selection criterion.

Another problem is the use of a *single* subspace to score outliers. Since the subspace selection criterion ignores dependencies, it can cause the removal of relevant dimensions. In the interesting cases, where the number of relevant dimensions is limited, the negative effects of removing a single relevant dimension can be even more drastic than keeping many irrelevant dimensions. The particularly problematic factor in using a single subspace is that if a mistake is made in subspace selection, there is virtually no chance of recovering from the mistake. In general, these types of methods are almost always outperformed by the various subspace ensemble methods discussed in this chapter (feature bagging [344], rotated bagging [32], *RS-Hash* [476], and isolation forests [367]).

### 5.3 Generalized Subspaces

Although axis-parallel methods are effective for finding outliers in most real settings, they are not very useful for finding outliers in cases where the points are aligned along arbitrary lower-dimensional manifolds of the data. For example, in the case of Figure 5.4, no 1-dimensional feature from the 2-dimensional data can find the outliers. On the other hand, it is possible to find *localized* 1-dimensional correlated subspaces so that most of the data aligns along these localized 1-dimensional subspaces, and the remaining deviants can be classified as outliers. Although this particular data set seems to be relatively easy for outlier detection because of its low dimensionality, this problem can become more challenging with increasing dimensionality.

These algorithms are generalizations of the following two classes of algorithms:

- The PCA-based linear models discussed in Chapter 3 find the *global* regions of correlation in the data. For example, in the case of Figure 5.3, the outliers can be effectively

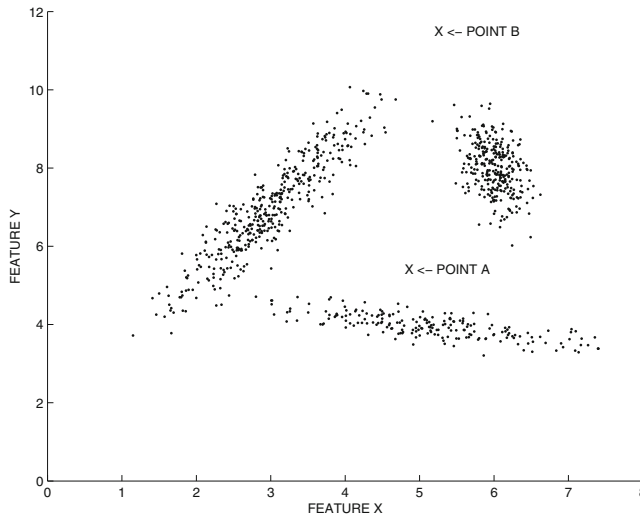


Figure 5.4: The example of Figure 2.9 revisited: Outliers are best discovered by determining deviations from local PCA-based clusters. Neither axis-parallel subspace outliers nor global-PCA can capture such clusters.

identified by determining these global directions of correlation. However, no such *global* directions of correlation exist in Figure 5.4.

- The axis-parallel subspace outliers discussed earlier in this chapter can find deviants, when the data is naturally aligned along low dimensional axis-parallel subspace clusters. However, this is not the case in Figure 5.4, in which the data is aligned along arbitrary directions of correlation.

The goal in generalized subspace analysis is to combine the ideas in these two types of algorithms. In other words, it is desired to determine the arbitrarily oriented subspaces *simultaneously* with outlier discovery. In the following, we will discuss several methods for finding such generalized subspaces. We will also discuss some methods for discovering *nonlinear* subspaces, in which the data is distributed along *local nonlinear manifolds*.

### 5.3.1 Generalized Projected Clustering Approach

This problem can be partially addressed with the use of generalized projected clustering methods, where the clusters are identified in arbitrarily aligned subspaces of the data [7]. The method discussed in [7] has a built-in mechanism in order to determine the outliers *in addition to* the clusters. Such outliers are naturally data points that do not align with the clusters. However, the approach is not particularly optimized for finding the outliers, because the primary purpose of the method is to determine the clusters. The outliers are discovered as a side-product of the clustering algorithm, rather than as the primary goal. Therefore, the approach may sometimes discover the weaker outliers, which correspond to the noise in the data. Clearly, methods are required for properly distinguishing between the strong and weak outliers by using an outlier scoring mechanism to distinguish between various points. The simplest approach is to compute the local Mahalanobis distance of every candidate outlier to each cluster centroid. The computation of the local Mahalanobis

distance of a point to a cluster centroid uses only the mean and covariance matrix of that cluster. The computation of the local Mahalanobis distance is described in section 4.2 of Chapter 4 (cf. Equation 4.2). For any given point, its smallest Mahalanobis distance (i.e., distance to its nearest centroid) is reported as its outlier score.

To improve robustness, one should use randomized methods to cluster the data in multiple ways, and average the point-wise scores from the different models. It is *very important* to combine scores from multiple subspaces, because the individual subspaces discovered using subspace clustering do not tell us much about the relevant subspaces for outlier detection. However, the outliers discovered using clustering often inherit the properties of the underlying clusters when an ensemble method is used. Therefore, using clusters in subspaces of the data yields outliers that are subspace sensitive. The work in [406] provides a specific example in the axis-parallel setting, where it shows that the combination of scores from multiple clusterings with the use of the *Multiple-Proclus* method greatly improves over the performance of a single application of the clustering algorithm.

Many other generalized projected clustering methods can be used and a detailed survey may be found in [23] (Chapter 9). Many of these methods are quite efficient. One advantage of this approach is that once the clusters have been identified up front, the scoring process is very efficient. Furthermore, the model-building process usually requires less than  $O(N^2)$  time, which is required by most distance-based detectors.

### 5.3.2 Leveraging Instance-Specific Reference Sets

In order to determine the outliers that are optimized to the locality of a particular data point, it is critical to determine localized subspaces that are optimized to the candidate data point  $\bar{X}$  being scored. The determination of such subspaces is non-trivial, since it often cannot be inferred from locally aggregate properties of the data, for detecting the behavior of *rare* instances. A method was recently proposed in [328] for finding outliers in generalized subspaces with the use of reference sets. The main difference from earlier generalized subspace clustering methods is that local reference sets are *specific to the various data points*, whereas clusters provide a fixed set of reference sets that are used to score all points. The price of this flexibility is that the running time of finding *each* point-specific reference set is  $O(N)$ . Therefore, the approach requires  $O(N^2)$  time for scoring.

For a given data point  $\bar{X}$ , this method finds the full-dimensional  $k$ -nearest neighbors of  $\bar{X}$ . This provides a reference set  $S$  with mean vector  $\bar{\mu}$ . The PCA approach of Chapter 3 is applied to the covariance matrix  $\Sigma(S)$  of the *local* reference set  $S$  in order to determine the key eigenvectors  $\bar{e}_1 \dots \bar{e}_d$ , in increasing order of variance, with corresponding eigenvalues  $\lambda_1 \leq \lambda_2 \dots \leq \lambda_d$ . The discussion in section 3.3 of Chapter 3 performs these same steps [493] except that they are performed on a *global* basis, rather than on a local reference set  $S$ . Even if all  $d$  dimensions are included, it is possible to create a normalized outlier score of a data point  $\bar{X}$  to the centroid  $\bar{\mu}$  of the data with the use of local eigenvalue scaling, as discussed in Chapter 3:

$$Score(\bar{X}) = \sum_{j=1}^d \frac{|(\bar{X} - \bar{\mu}) \cdot \bar{e}_j|^2}{\lambda_j} \quad (5.10)$$

As discussed in section 2.2.2.2 of Chapter 2, this can be approximately modeled as a  $\chi^2$  distribution with  $d$  degrees of freedom for each data point, and the outlier scores of the different data points can be reasonably compared to one another. Such an approach is used in [493] in the context of global data analysis. The survey paper of Chandola *et al.* [125]

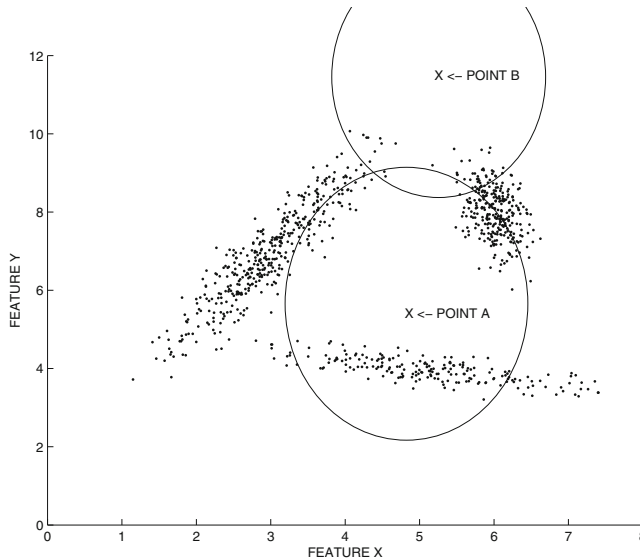


Figure 5.5: Local reference set may sometimes contain points from multiple generating mechanisms

provides a simpler exposition. Note that this approach can be viewed as a local version of soft PCA and it is not necessary to use subspaces.

The work in [328] is different from this basic approach in that it emphasizes the pre-selection of a subset of the eigenvectors for scoring. This is primarily because the work is positioned as a generalized subspace outlier detection method. As we will discuss later, this is not necessarily desirable.

The  $\delta$  eigenvectors<sup>3</sup> with the smallest eigenvalues are selected for score computation. Correspondingly, the pruned score is defined on the basis of the smallest  $\delta \leq d$  eigenvectors:

$$Score(\bar{X}, \delta) = \sum_{j=1}^{\delta} \frac{|(\bar{X} - \bar{\mu}) \cdot \bar{e}_j|^2}{\lambda_j} \quad (5.11)$$

How should the value of  $\delta$  be fixed for a particular data point  $\bar{X}$ ? The score is a  $\chi^2$ -distribution with  $\delta$ -degrees of freedom. It was observed in [328] that the value of  $\delta$  can be parameterized, by treating the  $\chi^2$  distribution as a special case of the  $\Gamma$  distribution.

$$Score(\bar{X}, \delta) \sim \Gamma(\delta/2, 2)$$

The optimal value of  $\delta$  is selected specifically for each data point, by selecting the value of  $\delta$  in order to determine the maximal unlikely deviation based on this model. This is done by using the cumulative density function of the aforementioned distribution. While this value can be directly used as an outlier score, it was also shown in [328], how this score may be converted into a more intuitive probability value.

This approach, however, has several weaknesses. These weaknesses arise from the lack of robustness in using a single subspace as relevant set of dimensions, from the way in which

<sup>3</sup>The work in [328] uses  $\delta$  as the number of *longest* eigenvectors, which is only a notational difference, but is noted here to avoid confusion.

the reference set is constructed, and from how the hard pruning methodology is used in score computation. We discuss each of these issues in detail below:

- A *single subspace* has been used by this approach for finding the outliers with the use of the local reference set  $S$ . If the local reference set  $S$  is not accurately determined, then this will not provide the proper directions of local correlation. The use of a single subspace is risky, especially with the use of weak aggregation-based hints, because it is often possible to unintentionally remove relevant subspaces. This can have drastic effects. The use of multiple subspaces may be much more relevant in such scenarios, such as the methods proposed in [32, 308, 344, 367, 402, 406].
- There is an inherent circularity in identifying the reference set with the use of full-dimensional  $k$ -nearest neighbor distances, especially if the distances are not meaningfully defined in full dimensionality. The choice of points in the reference set and the choice of the subspace clearly impact each other in a circular way. This is a classical “chicken-and-egg” problem in subspace analysis, which was first pointed out in [5]. The analysis in such cases needs to be *simultaneous* rather than *sequential*. As is well known, the most robust techniques for handling circularity in virtually all problem domains (e.g., projected clustering methods) use iterative methods, so that the point-specific and dimension-specific aspects of the problem are able to interact with one another. This is however, not the case in [328], in which a sequential analysis is used.

In particular, it may happen that many locally irrelevant features may be used during the determination of the local reference set, when full-dimensional distances are used. This set could therefore contain data points from multiple generating mechanisms, as illustrated in Figure 5.5. When the number of irrelevant features is unknown, a specific number of points in the reference set will not be able to avoid this problem. The use of a smaller reference set size can reduce the chance of this happening to some extent, but can never guarantee it, especially when many irrelevant features are used. On the other hand, reducing the reference set size can also result in a correlation hyperplane, whose eigenvalue statistics overfit an artificially small set of reference points. In fact, the real challenge in such problems is in properly selecting the reference set; this issue has been trivialized by this approach.

- It is not necessary to select a particular set of eigenvectors in a hard way, since the eigenvalues in the denominator of Equation 5.10 already provide a soft weighting to their relative importance (or relevance). For example, if for a large value of  $\lambda_i$ , a data point shows even larger deviations along that direction, such an outlier would either be missed by dimension pre-selection, or would include other less relevant dimensions. An example is the outlier  $B$  in Figure 5.5, which is aligned along the longer eigenvector, and therefore the longest eigenvector is the *most informative* about its outlier behavior. In particular, the method of selecting the  $\delta$  smallest eigenvectors implicitly assumes that the relevance of the attributes are ordered by eigenvalue magnitude. While this may generally be true for aggregation-based clustering algorithms, it is very often not true in outlier analysis because of the unusual nature of outliers. The possibility of outliers aligning along long eigenvectors is not uncommon at all, since two highly correlated attributes may often show highly deviant behavior of a similarly correlated nature. This example also shows, how *brittle* the rare nature of outlier analysis is to aggregation-based measures. This is because of the varying causes of rarity, which cannot be fully captured in aggregation statistics. This observation exemplifies the fact that straightforward generalizations of subspace selection methods from clustering

(based on aggregates), are often not appropriate or optimized for (the rare nature of) outlier analysis. One advantage of using all the dimensions is that it reduces to a local Mahalanobis distance with the same dimensionality, and allows better comparability in the scores across different outliers. In such cases, intuitive probability values may be derived more simply from the  $\chi^2(d)$  distribution.

The high-dimensional case is an extremely difficult one, and it is understandable that no given method will be able to solve these problems perfectly.

### 5.3.3 Rotated Subspace Sampling

The rotated subspace sampling method was recently proposed in [32] as an ensemble method, which improves over feature bagging [344]. This approach is also referred to as *rotated bagging*. Just as feature bagging is designed for discovering outliers in axis-parallel subspaces, the rotated bagging method is designed for discovering outliers in generalized subspaces. As in the case of feature bagging, this approach is an ensemble method and it can use any off-the-shelf outlier detection algorithm (e.g., LOF) as the *base detector*.

The basic idea in rotated bagging is to sample randomly rotated subspaces in lower-dimensional space, and score each point in this low-dimensional space. The scores from various subspaces can be combined to provide the final result. In particular, the approach uses subspaces of dimensionality  $r = 2 + \lceil \sqrt{d}/2 \rceil$ , which is much lower the typical dimensionality of the subspace used in feature bagging. This is because the axis rotation enables the capturing of information from all dimensions to varying degrees. The ability to use lower-dimensional projections is also useful for inducing diversity and thereby improving the quality of the overall *ensemble* score. The rotated bagging algorithm works as follows:

1. Determine a randomly rotated axis system in the data.
2. Sample  $r = 2 + \lceil \sqrt{d}/2 \rceil$  directions from rotated axis system. Project data along these  $r$  directions.
3. Run the base outlier detector on projected data and store the scores of each point.

The component scores can be combined by either (a) using the average score of a point across different projections, or (b) using the maximum score of a point across different projections. Other combination functions are discussed in Chapter 6. It is important to use standardization on the scores before the combination.

In order to determine  $r = 2 + \lceil \sqrt{d}/2 \rceil$  randomly rotated mutually orthogonal directions, a  $d \times r$  random matrix  $Y$  is generated, such that each value in the matrix is uniformly distributed in  $[-1, 1]$ . Let the  $t$ th column of  $Y$  be denoted by  $\overline{y}_t$ . Then, the  $r$  random orthogonal directions  $\overline{e}_1 \dots \overline{e}_r$  are generated using a straightforward Gram-Schmidt orthogonalization of  $\overline{y}_1 \dots \overline{y}_r$  as follows:

1.  $t = 1$ ;  $\overline{e}_1 = \frac{\overline{y}_1}{|\overline{y}_1|}$
2.  $\overline{e}_{t+1} = \overline{y}_{t+1} - \sum_{j=1}^t (\overline{y}_{t+1} \cdot \overline{e}_j) \overline{e}_j$
3. Normalize  $\overline{e}_{t+1}$  to unit norm.
4.  $t = t + 1$
5. if  $t < r$  go to step 2



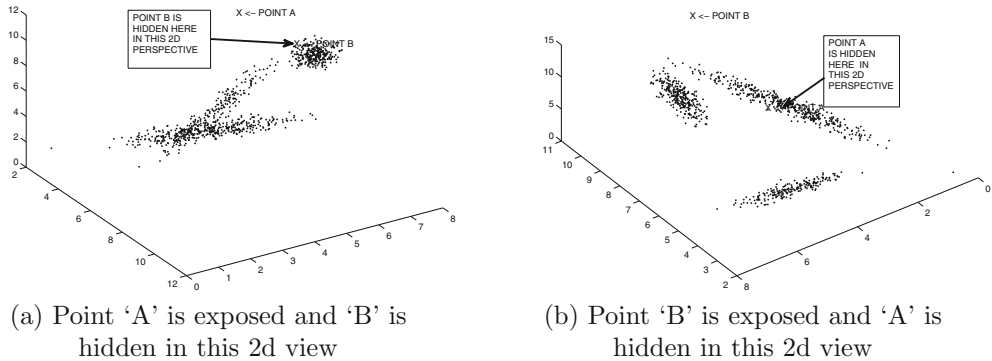


Figure 5.6: In this 3-dimensional data set, points ‘A’ and ‘B’ are exposed in different 2-dimensional views (projections). However, the averaging or maximization score-combination will expose both ‘A’ and ‘B.’

Let the resulting  $d \times r$  matrix with columns  $\bar{e}_1 \dots \bar{e}_r$  be denoted by  $E$ . The  $N \times d$  data set  $D$  is transformed and projected to these orthogonal directions by computing the matrix product  $DE$ , which is an  $N \times r$  matrix of  $r$ -dimensional points. The results in [32] show that the approach improves over feature bagging. Further improvements may be obtained by combining it with other ensemble techniques. Rotated bagging uses a more general model to describe the outliers than feature bagging in terms of *arbitrary subspaces* and therefore using an ensemble is even more important. It is often difficult to discover a particular local subspace that is relevant to a particular data point. The great power of ensembles lies in the fact that an averaged ensemble combination of many global subspace selections is often able to discover the locally relevant subspaces. In other words, the ensemble is inherently more powerful than its individual members. This point can be explained from the perspective of how such types of averaged models use ensembles to reduce representational bias (cf. section 6.4.3.1 of Chapter 6). Furthermore, the maximization combination function often does even better in terms of reducing representational bias.

An example of a 3-dimensional data set is illustrated in Figure 5.6. Here, we have shown different 2-dimensional views of the data. It is clear that in the case of Figure 5.6(a), outlier ‘A’ is exposed, whereas in the case of Figure 5.6(b), outlier ‘B’ is exposed. However, if one were to use the average or maximization combination of the scores obtained by running the detector on these two views, *both* points ‘A’ and ‘B’ might be scored highly in the combination (and therefore discovered as outliers). This is an example of how ensemble methods overcome the *representational bias* in individual detectors in order to provide a more general model. Another example of this power of ensemble methods to overcome representational bias is provided in Figure 6.4 of Chapter 6. An ensemble-centric description of rotated bagging is also provided in section 6.4.4 of Chapter 6.

### 5.3.4 Nonlinear Subspaces

The most difficult case of subspace outlier detection occurs in cases in which the manifolds exist in lower-dimensional subspaces of arbitrary shape. Examples of such patterns in the

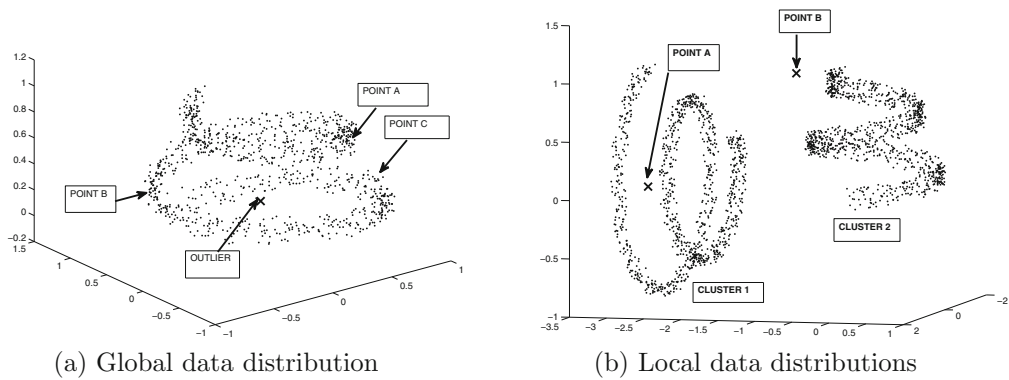


Figure 5.7: Arbitrarily shaped clusters can exist in lower-dimensional manifolds (Revisiting Figure 4.3 of Chapter 4)

data are illustrated in Figure 5.7. This figure is the same as Figure 4.3 of Chapter 4. In fact, the clustering method proposed in section 4.2.1 provides one of the ways in which outliers can be discovered in such nonlinear settings. Since the discussion in this section is roughly based on the framework established in section 4.2.1 of Chapter 4, the reader is advised to revisit that section before proceeding further.

The arbitrary shape of the clusters presents a number of unique challenges. There are several cases in Figures 5.7(a) and (b), in which the outliers are placed in the interior of non-convex patterns. Without an appropriate *local nonlinear transformation*, it becomes more challenging to discover such outliers. This extraction is achieved with spectral methods, which can be viewed as a special class of kernel methods that is friendly to discovering clusters of arbitrary shape.

As in the case of section 4.2.1, a spectral embedding of the data is extracted. This is achieved by constructing a  $k$ -nearest neighbor graph and computing its top eigenvectors. Because of the way in which the spectral embedding is constructed, it already adapts itself to local nonlinear patterns in the data, and Euclidean distances can be used on the transformed data set. One difference from the method of section 4.2.1 is that an additional step is used to reduce the noise in the spectral embedding [475]. The main problem in spectral embedding is caused by the presence of bad edges in the neighborhood graph, which connect disjoint clusters. Therefore, the similarity graph needs to be corrected to improve the quality of the embedding. In particular, an iterative approach is used to construct the spectral embedding in which the similarity matrix (used to construct the embedding) is corrected for erroneous computations between pairs of points. The spectral embedding is used to correct the similarity matrix, and the corrected similarity matrix is used to construct the spectral embedding. The basic idea here is that the spectral embedding from a similarity matrix  $S$  will itself create an embedding from which a new similarity matrix  $S'$  can be constructed. In the new representation, weakly related points move away from each other on a relative basis (like points A and C in Figure 5.7(a)), and strongly related points move towards each other. Therefore, the Hadamard product of  $S$  and  $S'$  is used to adjust  $S$ :

$$S \Leftarrow S \circ S' \quad (5.12)$$

This has the effect of correcting the original similarity matrix  $S$  to de-emphasize noisy

similarities on a *relative* basis. The new similarity matrix is used to again create a new embedding. These steps are repeated for a small number of iterations and the final embedding is used to score the points. This process is described in detail in [475]. Each of the new dimensions of the embedding are normalized to unit variance. A  $k$ -nearest neighbor detector is executed on the embedding in order to return the final outlier score. By using a  $k$ -nearest neighbor detector in the transformed space, one is effectively using a data-dependent distance function that is sensitive to the nonlinear subspace patterns in the original space. Although the work in [475] does not discuss it, the approach can be strengthened with the use of averaged scores from multiple clusterings with different parameters.

### 5.3.5 Regression Modeling Techniques

It is also possible to use regression models for subspace outlier detection by identifying points violating attribute-wise dependencies (e.g., the young Alzheimer patient discussed earlier). The basic idea is to decompose a  $d$ -dimensional unsupervised outlier detection problem into a set of  $d$  regression modeling problems. One can predict each attribute with the remaining  $(d - 1)$  attributes using an off-the-shelf regression model. For each instance, the squared errors of predicting the various attributes are aggregated to create the outlier score. This approach is discussed in detail in section 7.7 of Chapter 7. When certain types of base detectors like random forests are used, the approach can be interpreted as a local subspace outlier detection method. These connections are discussed in section 7.7.

## 5.4 Discussion of Subspace Analysis

---

The major reason for difficulty in high-dimensional outlier analysis is because of the masking effects of the locally noisy and irrelevant nature of some of the dimensions, which is often also manifested in the concentration of distances. It has been claimed in a recent survey [620] that the literature only focuses on the distance-concentration issue and “abstains from” discussing the impact of *locally relevant dimensions* caused by differential generating mechanisms (thereby claiming the latter as a new insight of the survey [620]). This is an incorrect and particularly surprising assertion, because *both* the aspects of local feature selection (relevance) and distance concentration have been studied extensively in the literature together with their connections to one another. The original work in [4] (and virtually every subsequent work [308, 344, 402]) provides a pictorial illustration (similar to Figure 5.1) and a fairly detailed discussion of how (locally) irrelevant attributes mask outliers in different feature-specific views of the data. In the context of such a pictorial illustration, the following is stated in [4] about locally irrelevant attributes:

“Thus, by using full dimensional distance measures it would be difficult to determine outliers effectively because of the averaging behavior of the noisy and irrelevant dimensions. Furthermore, it is impossible to prune off specific features a priori, since different points may show different kinds of abnormal patterns, each of which use different features or views.”

The ineffectiveness of *global* feature selection in high-dimensional data in fact forms the motivating reason for subspace analysis, which can be considered a *local* feature selection method, or a *local* dimensionality reduction method [7, 121]. These connections of local subspace analysis to the ineffectiveness of global feature selection in high-dimensional data were explicitly discussed in detail in the motivational discussion of one of the earliest works

on subspace analysis [5, 263]. At this point, these results are well known and established<sup>4</sup> wisdom. While it is possible to reduce the distance concentration effects by carefully calibrating the fraction of informative dimensions, such cases are (usually) not interesting for subspace analysis.

Although it is most definitely true that noisy and irrelevant attributes mask the outliers, the observation is certainly not new, and the two factors of distance concentration and local feature relevance are closely related. In fact, even the experimental simulations in the survey [620] show that concentration effects tend to co-occur in settings where there are too many irrelevant attributes. This is, of course, not a hard rule; nevertheless, it is a significant issue for distance-based detectors. The interesting cases for subspace analysis (typically) show some levels of both properties. Even limited levels of distance concentration impact the effectiveness of full-dimensional distance-based algorithms, and this impact is therefore important to examine in outlier analysis. It should be noted that noisy and irrelevant attributes are more likely to lead to concentration of distances. For example, for the case of uniformly distributed data, where all attributes are noisy, the concentration effect is extreme, and an outlier deviating along *a relatively small number of dimensions* will be hard to discover by full-dimensional methods. In such cases, from a full-dimensional distance-based or density-based perspective, all data points have almost equally good outlier scores, and this can be equivalently understood in terms of *either* locally irrelevant features or distance concentration effects. Of course, real data sets are not uniformly distributed, but *both* irrelevant features and concentration effects are present to varying degrees in different data sets. The general assumption for subspace analysis is that the addition of more dimensions often does not add *proportionally* more information for a particular outlier. The challenging outliers are often defined by the behavior of a small number of dimensions, and when the point-specific information does not increase substantially with data dimensionality, even modest concentration effects will have a negative impact on full-dimensional algorithms. The more the number of irrelevant attributes there are, the more erroneous the computations for full-dimensional distance-based methods are likely to be. An extreme example at the other end of the spectrum is where an outlier shows informative and deviant behavior in every dimension, and therefore outlier characteristics grow *stronger* with increasing dimensionality. However, in this rather uninteresting case, since the outlier shows *both* many relevant features *and* also typically does not conform to the distance concentration behavior of the remaining data, a trivial full-dimensional distance-based algorithm would discover it easily in most cases. In general, cases where the informative dimensions also increase significantly with data dimensionality, are not as interesting for subspace analysis because the full-dimensional masking behavior becomes less prominent in this easier case. Subspace analysis does not exclude the possibility that the more obvious deviants may also be found by full-dimensional analysis.

There are many high-dimensional data sets, where one can perform effective outlier detection by simpler full-dimensional algorithms. The effectiveness of a particular algorithm on a full-dimensional data set depends on the application scenario and the way in which the features are extracted. If the features are extracted with a particular anomaly detection scenario in mind, then a full-dimensional algorithm is likely to work well. In spite of this fact, subspace analysis will often discover outliers that are not easily found by other full-dimensional algorithms. Indeed, subspace analysis should not be viewed as a stand-alone method, but inherently as an ensemble method to improve the performance of various types

---

<sup>4</sup>Some of the earliest methods even refer to these classes of techniques as local dimensionality reduction [121] in order to emphasize the enhanced and differential local feature selection effect, which arises as a result of different generating mechanisms.

of base detectors. Whether one develops a specific base method for high-dimensional outlier detection (like subspace histograms) or whether one wraps it around existing detectors (like feature and rotated bagging), there are clear benefits to incorporating this principle in outlier detection. Furthermore, subspace analysis provides useful insights about the causality of the anomaly; one can use the relevant local subspaces to extract an understanding about the specific combination of relevant features. A simple example is the case of the young Alzheimer patient discussed earlier in this chapter.

Outliers, by their very rare nature, may often be hidden in small combinations of dimensions in a high-dimensional data set. Subspace analysis is particularly interesting for such scenarios. On the other hand, when more dimensions do add (significantly) more information, then this becomes an easy case for analysis, which no longer remains interesting. In more difficult cases, the vast majority of noisy dimensions make it difficult to discriminate between data points from a density-based or data-sparsity perspective. Subspace analysis works particularly well in these cases when the outliers exist in a modest number of locally relevant dimensions. This observation also points to a particularly difficult case in which the number of locally irrelevant dimensions increases with data dimensionality, and a very large number of dimensions also continue to be *weakly* relevant (but not strongly relevant). This situation often occurs in data sets containing thousands of dimensions. Such data sets remain an unsolved case for all classes of full-dimensional and subspace methods.

To summarize, the following principles need to be kept in mind while designing subspace methods:

- A direct exploration of rare regions is possible, though it is computationally challenging because of combinatorial explosion [4].
- Aggregation-based methods provide only weak hints about the relevant subspaces. The main power of these methods lies only in the ensemble-centric setting by combining the results from different subspaces.
- The individual component of an ensemble should be designed with efficiency considerations in mind. This is because efficient components enable the practical use of a larger number of components for better accuracy.

One interesting observation is that even when weak base detectors are used, combining them often leads to very strong results. The success of methods like feature bagging, subspace histograms, and rotated bagging is based on this fact. Note that in each case, the underlying base detector is not particularly strong; yet the final outlier detection results are very effective. Recent advancements in the field of high-dimensional outlier detection and ensemble analysis have been very significant. In spite of these advancements, many high-dimensional data sets continue to remain a challenge for outlier analysis.

## 5.5 Conclusions and Summary

---

Subspace methods for outlier detection are used in cases, where the outlier tendency of a data point is diluted by the noise effects of a large number of locally non-informative dimensions. In such cases, the outlier analysis process can be sharpened significantly by searching for subspaces in which the data points deviate significantly from the normal behavior. The most successful methods identify multiple relevant subspaces for a candidate outlier, and then combine the results from different subspaces in order to create a more robust ensemble-based ranking.

Outlier analysis is the most difficult problem among all classes of subspace analysis problems. This difficulty arises out of the rare nature of outliers, which makes direct statistical analysis more difficult. Since subspace analysis and local feature selection are related, it is noteworthy that even for global feature selection, there are few known methods for outlier analysis, as compared to clustering and classification algorithms. The reason is simple: enough statistical evidence is often not available for the analysis of rare characteristics. Robust statistics is all about *more* data, and outliers are all about *less* data and statistical non-conformity with most of the data! Regions and subspaces containing statistical conformity tell us very little about the complementary regions of non-conformity in the particular case of high-dimensional subspace analysis, since the *potential* domain of the latter is much larger than the former. In particular, a local subspace region of the greatest aggregate conformity does not necessarily reveal anything about the rare point with the greatest statistical non-conformity.

Although many recent ensemble methods for subspace analysis have shown great success, a particularly difficult case is one in which a large number of dimensions are weakly relevant (but not strongly relevant), and even more dimensions are locally irrelevant. These cases often occur in data sets containing thousands of dimensions, and remain unsolved by existing methods. While it is doubtful that the more difficult variations of the problem will ever be fully solved, or will work completely in all situations, the currently available techniques work in many important scenarios. There are many merits in being able to design such methods, because of the numerous insights they can provide in terms of identifying the causes of abnormality. The main challenge is that outlier analysis is so brittle, that it is often impossible to make confident assertions about inferences drawn from aggregate data analysis. The issue of efficiency seems to be closely related to that of effectiveness in high-dimensional outlier analysis. This is because the search process for outliers is likely to require exploration of multiple local subspaces of the data in order to ensure robustness. With increasing advances in the computational power of modern computers, there is as yet hope that this area will become increasingly tractable for analysis.

## 5.6 Bibliographic Survey

---

In the context of high-dimensional data, two distinct lines of research exist, one of which investigates the *efficiency* of high-dimensional outlier detection [58, 219, 557], and the other investigates the more fundamental issue of the *effectiveness* of high-dimensional outlier detection [4]. Unfortunately, the distinction between these two lines of work is sometimes blurred in the literature, even though these are clearly different lines of work with very different motivations. It should be noted that the methods discussed in [58, 219, 557] are all *full-dimensional methods*, because outliers are defined on the basis of their full-dimensional deviation. Although the method of [557] uses projections for indexing, this is used only as an approximation to improve the efficiency of the outlier detection process.

In the high-dimensional case, the efficiency of (full-dimensional) outlier detection also becomes a concern, because most outlier detection methods require repeated similarity searches in high dimensions in order to determine the nearest neighbors. The efficiency of these methods degrades because of two factors: (i) the computations now use a larger number of dimensions, and (ii) the effectiveness of pruning methods and indexing methods degrades with increasing dimensionality. The solution to these issues still remains unresolved in the vast similarity search literature. Therefore, it is unlikely that *significantly* more efficient similarity computations could be achieved in the context of high-dimensional

outlier detection, although some success has been claimed for improving the efficiency of high-dimensional outlier detection in methods proposed in [58, 219, 557]. On the whole, it is unclear how these methods would compare to the vast array of techniques available in the similarity-search literature for indexing high-dimensional data. This chapter does *not* investigate the efficiency issue at all, because the efficiency of a *full-dimensional* outlier-detection technique is not important, if it does not even provide meaningful outliers. Therefore, the focus of the chapter is on methods that *re-define* the outlier detection problem in the context of lower-dimensional projections.

The problem of subspace outlier detection was first proposed in [4]. In this paper, an evolutionary algorithm was proposed to discover the lower dimensional subspaces in which the outliers may exist. The method for distance-based outlier detection with subspace outlier degree was proposed in [327]. Another distance-based method for subspace outlier detection was proposed in [411]. Some methods have also been proposed for outlier analysis by randomly sampling subspaces and combining the scores from different subspaces [344, 367]. In particular, the work in [344] attempts to combine the results from these different subspaces in order to provide a more robust evaluation of the outliers. These are essentially *ensemble-based* methods that attempt to improve detection robustness by bagging the results from different sets of features. The major challenge of these methods is that random sampling may not work very well in cases where the outliers are hidden in specific subspaces of the data. The work in [308] can be considered a generalization of the broad approach in [344], where only high-contrast subspaces are selected for the problem of outlier detection. The use of information-theoretic measures for biased subspace selection is discussed in [413].

The work on isolation forests is related to earlier work on using random forests and clustering forests for clustering, similarity computation, sparse encoding, and outlier detection [99, 401, 491, 555]. In particular, the work in [401] creates extremely randomized clustering forests (ERC-Forests) for clustering and coding. The isolation forest can be viewed as a special case of the ERC-Forest in which the number of trials at each node is set to 1 and the trees are grown to full height. Many variations of the isolation forest [367], such as half-space trees [532], have been proposed. The main difference between an isolation tree and a half-space tree is that the latter is a fully balanced tree of a fixed height, and splits are performed by picking points that are half-way between the minimum and maximum ranges of each attribute. Furthermore, the minimum and maximum ranges of each attribute are defined in a perturbed way to induce diversity. The number of data points in the leaf node of a test instance is multiplied with the number of leaf nodes to obtain its outlier score in a single ensemble component. These scores are averaged over different half-space trees. The multiplication of each component score with the number of leaf nodes is helpful in cases where different trees have different depth [547]. Recently, the subspace histogram technique, referred to as *RS-Hash*, has been proposed in [476]. This technique averages the log-density in grid regions of varying sizes and shapes in order to provide the final score. The approach uses randomized hashing for efficiency and requires linear time. Such methods can also be used for streaming data.

The reverse problem of finding outlying subspaces *from* specific points was studied in [605, 606, 607]. In these methods, a variety of pruning and evolutionary methods were proposed in order to speed up the search process for outlying subspaces. The work in [59] also defines the exceptional properties of outlying objects, both with respect to the entire population (global properties), and also with respect to particular sub-populations to which it belongs (local properties). Both these methods provide different but meaningful insights about the underlying data. A genetic algorithm for finding the outlying subspaces in high-dimensional data is provided in [606]. In order to speed up the fitness function evaluation,

methods are proposed to speed up the computation of the  $k$ -nearest neighbor distance with the use of bounding strategies. A broader framework for finding outlying subspaces in high-dimensional data is provided in [607]. A method that uses two-way search for finding outlying subspaces was proposed in [582]. In this method, full-dimensional methods are first used to determine the outliers. Subsequently, the key outlying subspaces from these outlier points are detected and reported. A method for using rules in order to explain the context of outlier objects is proposed in [405].

A number of ranking methods for subspace outlier exploration have been proposed in [402, 403, 404]. In these methods, outliers are determined in multiple subspaces of the data. Different subspaces may either provide information about different outliers or about the same outliers. Therefore, the goal is to combine the information from these different subspaces in a robust way in order to report the final set of outliers. The *OUTRES* algorithm proposed in [402] uses recursive subspace exploration in order to determine all the subspaces relevant to a particular data point. The outlier scores from these different subspaces are combined in order to provide a final value. A tool-kit for ranking subspace outliers was presented in [403]. A more recent method for using multiple views of the data for subspace outlier detection is proposed in [406]. Methods for subspace outlier detection in multimedia databases were proposed in [64].

Most of the methods for subspace outlier detection perform the exploration in axis-parallel subspaces of the data. This is based on the complementary assumption that the dense regions or clusters are hidden in axis-parallel subspaces of the data. However, it has been shown in recent work that the dense regions may often be located in arbitrarily oriented subspaces of the data [7]. Such clustering methods can be used in conjunction with the methodology discussed in section 5.2.4 to discover outliers. Another work in [328] provides a solution based on local reference sets rather than clusters. The work in [32] proposes a rotated bagging approach; this can be viewed as the analog of the feature bagging approach for the arbitrarily oriented case. Finally, a method for finding outliers in the context of nonlinear subspaces was proposed in [475].

Recently, the problem of outlier detection has also been studied in the context of dynamic data and data streams. The SPOT method was proposed in [604], which is able to determine projected outliers from high-dimensional data streams. This approach employs a window-based time model and decaying cell summaries to capture statistics from the data stream. The most sparse subspaces are obtained by a variety of supervised and unsupervised learning processes. These are used to identify the projected outliers. A multi-objective genetic algorithm is employed for finding outlying subspaces from training data.

The problem of high-dimensional outlier detection has also been extended to other application-specific scenarios such as astronomical data [261], uncertain data [26], transaction data [255], and supervised data [619]. In the uncertain scenario, high-dimensional data is especially challenging, because the noise in the uncertain scenario greatly increases the sparsity of the underlying data. Furthermore, the level of uncertainty in the different attributes is available. This helps determine the importance of different attributes for outlier detection purposes. Subspace methods for outlier detection in uncertain data are proposed in [26]. Supervised methods for high-dimensional outlier detection are proposed in [619]. In this case, a small number of examples are identified and presented to users. These are then used in order to learn the critical projections that are relevant to the outlierness of an object. The learned information is then leveraged in order to identify the relevant outliers in the underlying data.



## 5.7 Exercises

---

1. Which of the following data points is an outlier in some well chosen two-dimensional projection:  $\{ (1, 8, 7), (2, 8, 8), (5, 1, 2), (4, 1, 1), (3, 1, 8) \}$
2. Download the *Arrhythmia* data set from the UCI Machine Learning Repository [203]. Write a computer program to determine all distance-based outliers in different 2-dimension projections. Are the outliers the same in different projections?
3. In the *Arrhythmia* data set mentioned in the previous exercise, examine the *Age*, *Height* and *Weight* attributes of the *Arrhythmia data* set both independently and in combination. Draw a scatter plot of each of the 1-dimensional distributions and different 2-dimensional combinations. Can you visually see any outliers?
4. Write a computer program to determine the subspace outlier degree of each data point in the *Arrhythmia* data set for all 1-dimensional projections and 2-dimensional projections. Which data points are declared outliers?
5. Write a computer program to perform subspace sampling of the *Arrhythmia* data set, using the approach of [344] by sampling 2-dimensional projections. How many subspaces need to be sampled in order to robustly identify the outliers found in Exercise 2 over different executions of your computer program.
6. Consider a data set with  $d$ -dimensions, in which exactly 3 specific dimensions behave in an abnormal way with respect to an observation. How many minimum number of random subspaces of dimensionality  $(d/2)$  will be required in order to include all 3 dimensions in one of the sampled subspaces with probability at least 0.99? Plot the number of required samples for different values of  $d > 6$ .

---

---

## Chapter 6

# Outlier Ensembles

---

---

“Talent wins games, but teamwork and intelligence wins championships.”– Michael Jordan

### 6.1 Introduction

---

Ensemble analysis is a popular method used to improve the accuracy of various data mining algorithms. Ensemble methods combine the outputs of multiple algorithms or *base detectors* to create a unified output. The basic idea of the approach is that some algorithms will do well on a particular subset of points whereas other algorithms will do better on other subsets of points. However, the ensemble combination is often able to perform more robustly across the board because of its ability to combine the outputs of multiple algorithms. In this chapter, will use the terms *base detector* and *component detector* interchangeably to denote the individual algorithms whose outputs are combined to create the final result.

Ensemble analysis is used frequently in various data mining and machine learning applications such as clustering, classification, outlier analysis, and recommender systems. Furthermore, it has also been applied to various data-types such as multidimensional data, time-series data, ratings data, and networks. The ubiquity of these methods across various problem domains and data types is a result of their relative success in various settings. Ensemble methods regularly win various data mining competitions, the most well-known of which is its victory in the Netflix Prize challenge for recommender systems [34].

Relative to other problem domains, ensemble analysis is a recent field in outlier detection. The reasons for its relative recency (compared to other data mining problems) were discussed in the first position paper on outlier ensembles [31]. One of the key differences between outlier detection and many other problems (like classification) is that the former is an *unsupervised* problem unlike the latter. Supervised ensemble algorithms enjoy the luxury of using the class labels in the analytical process. For example, methods like boosting use the class labels to evaluate the accuracy of data points and weight them accordingly. Such methods cannot be generalized to outlier detection easily because of the unavailability of ground-truth.

In spite of these differences, it has been shown that the field of outlier ensembles shares a very large number of practical [31] and theoretical [32] characteristics with classification ensembles. As discussed in section 6.3, one can formulate a modified bias-variance trade-off for the outlier analysis setting as is available for the classification setting. As a result, many types of algorithms can be generalized directly from classification to outlier detection, and the results are also similar. The main difference from classification is that *the dependent variable (outlier label) is unobserved*. This means that we cannot adapt classification-ensemble algorithms that use these labels to outlier detection. Furthermore, issues such as parameter tuning remain a larger challenge in the context of unsupervised problems. All these challenges require subtle changes to algorithms for classification ensembles in order to generalize them to outlier detection.

The most common use-case for outlier ensembles is the setting of subspace outlier detection. This is primarily because of the uncertainty associated with identifying relevant subspaces for a given data point. Outlier ensembles are most useful in settings in which they are able to reduce the uncertainty associated with difficult algorithmic choices. In fact, the first subspace-outlier detection algorithm [4] can also be viewed as an ensemble method in which a maximization combination function is used on the scores to report outliers. Subsequently, as discussed in the previous chapter, most of the prominent methods for subspace outlier detection tend to use ensemble methods. The reasons for this lie in the natural similarities between subspace outlier detection and ensemble analysis. Just as ensemble analysis works well in cases in which different algorithms work better on different subsets of points, high-dimensional outlier detection works best in cases where different subspaces are relevant to different sets of points. Therefore, by using ensemble components to explore different subsets of dimensions, one is often able to combine the results of these different exploratory base detectors to create stable and accurate results.

There are two key design choices in the construction of an ensemble method:

1. **Choice of base detector:** Selecting the base detector is one of the first steps of an ensemble method. These base detectors might be completely different from one another, have different settings of the parameters, or they might use reconstructed data sets from the base data.
2. **Methodology for score normalization and combination:** Different detectors might create scores on different scales. For example, an average  $k$ -nearest neighbor detector will produce a raw distance score, whereas the LOF algorithm will produce a normalized value. Furthermore, although the general convention is to output larger scores for outliers, some detectors use the opposite convention and output smaller scores for outliers. Therefore, it is desirable to convert the scores from various detectors into normalized values that can be meaningfully combined. After the scores have been normalized, an important issue is the choice of the combination function used for computing the ensemble score of a point as a function of its scores returned by various base components. The most common choices include the averaging and maximization combination functions, which are discussed later in this chapter.

The design of a base detector and its combination method both depend on the specific goals of a particular ensemble method. This aspect critically depends on the theoretical foundations of outlier ensemble analysis, which decompose the error of an outlier detection technique into two components, known as the *bias* and the *variance*. These quantities have exactly similar interpretations to their counterparts used in the classification domain. The underlying theoretical assumption is that the data set is generated from some unknown base

distribution of data points. Note that if we actually had access to the base distribution, we can effectively have access to an infinite resource because we can generate as many training instances as we want from the data. However, we never actually have access to the base distribution, and we only have access to a single instance of the training data set. Trying to create a model with a finite training data set will inevitably lead to errors because of the inability to fully model the complexities of the base distribution. Even if we had infinite data, the specific model used might not be appropriate for the data distribution at hand. These two sources of error contribute to the variance and the bias, respectively. An intuitive understanding of these two sources of error is as follows:

1. **Variance:** It is assumed that the training data sets are generated from a base distribution for scoring outliers. Typically, the analyst only has access to a single finite data set drawn from the base distribution, as a result of which she will obtain a different outlier score for the same point when different finite data sets are used to score it using the same algorithm. For example, the  $k$ -nearest neighbor score of the same out-of-sample test point will be different over two different sets of 100 training points. This difference in results obtained from different training data sets (drawn from the same distribution) is a manifestation of model variance, which is a component of the error.
2. **Bias:** Even though one does not have ground-truth available in outlier detection problems, it is assumed that some *hypothetically ideal set of scores* do exist, which are not available to the algorithm. This is different from classification, where one at least has a (possibly noisy) sample of the dependent variable. Note that a particular outlier detection model might not appropriately reflect these hypothetically ideal scores, and therefore the *expected* scores returned by the algorithm will vary from the true scores. For example, consider a setting in which the generating process (theoretical base distribution) causes outliers uniformly at random throughout the entire range of the data space. On the other hand, if we use a multivariate extreme-value detector like the Mahalanobis method (section 2.3.4) to compute the scores, we are expected to perform poorly because of the inherent mismatch between the theoretical distribution of outliers and the modeling assumptions. This source of error is referred to as the bias. It is generally much harder to reduce bias in outlier detection problems in a controlled way because the ground-truth is not available as a “guiding post” to correct the errors in the assumptions of the model. Most bias-reduction methods in supervised settings use the ground truth in one form or the other.

Most outlier detectors are designed to work with one or more base detectors, and use a repeated application of these base detectors in order to generate scores with improved bias or variance. This chapter will provide several examples of both bias and variance reduction.

This chapter is organized as follows. In the next section, we will provide a broad overview of the various ways in which different base detectors are combined to create an ensemble. The theoretical foundations of outlier ensembles are introduced in section 6.3. In section 6.4, we will discuss several ensemble methods for variance reduction. Ensemble methods for bias reduction are discussed in section 6.5. A detailed discussion of several score combination methods is provided in section 6.6. The conclusions and summary are presented in section 6.7.

## 6.2 Categorization and Design of Ensemble Methods

---

As discussed in Chapter 1, ensemble methods can be categorized in two different ways. These two types of categorization are mutually orthogonal. The first type of categorization defines model-centric and data-centric ensembles as follows:

1. **Model-centric ensembles:** In this case, the different base detectors of the ensemble correspond to different models (algorithms), different parameter settings of the same model, or different randomized instantiations of the same model.
2. **Data-centric ensembles:** In this case, the different base detectors of the ensemble correspond to the application of the same model on different derivatives of the data. These different derivatives may be constructed by drawing samples from the data, drawing random projections from the data, weighting the data points, weighting the dimensions, or adding noise to the data.

After all the base detectors have been executed, the scores from the different models are combined to provide the final result. This combination is often performed by taking the average or the maximum of the scores from various detectors, although other possibilities will also be discussed in this chapter.

It is noteworthy that data-centric ensembles can be considered special cases of model-centric ensembles, when the process of drawing the training data from the base data is considered a part of the model execution. In fact, some data-centric ensembles like feature bagging can be better explained by viewing them from a model-centric perspective. However, explicitly distinguishing between model-centric and data-centric ensembles is helpful in distinguishing between their subtle theoretical differences [35].

There is, however, a different way of categorizing the different ensemble methods in terms of the level of independence of the base detectors. This methodology for categorizing the base detectors is as follows:

1. **Independent ensembles:** In this case, the different components of the base detector are executed independently of one another. The scores from the different detectors are then combined. Most of the existing ensemble methods fall into this category.
2. **Sequential ensembles:** In this case, the base detectors of the ensemble are executed one after the other in order to create successively refined models. An example of such an algorithm is one in which outliers are successively removed from the data to create a better model of normal data. Either the results from the final execution of the ensemble are reported, or the results from various components are combined into a unified prediction.

Note that independent ensembles can be either model-centric or data-centric; similarly, sequential ensembles can be either model-centric or data-centric. A large number of models have been proposed that belong to these various categories. However, sequential ensembles tend to be less explored than other types of models. This is, in part, because sequential ensembles are inherently designed for settings like classification in which ground-truth is available to use as a “guiding post” for future executions of the ensemble method. A more detailed discussion of these different categories of ensemble algorithms is provided in [35].

### 6.2.1 Basic Score Normalization and Combination Methods

Although the topic of score combination will be discussed in greater detail in section 6.6, it is necessary to touch on this topic early in this chapter in order to explain the various ensemble algorithms properly.

Throughout this chapter, we will assume that the data set  $\mathcal{D}$  contains  $N$  points, which is typically both the training and test data. It is common not to distinguish between training and test data in unsupervised problems, although it is much easier to perform theoretical analysis by distinguishing between them. In cases where the training and test data set are distinguished from one another, the number of training points is denoted by  $N$  and the number of test points is denoted by  $n$ . It is assumed that the number of base detectors is  $m$ , and the scores of these base detectors for the  $i$ th point are denoted by  $s_1(i) \dots s_m(i)$ . The goal of the combination process is to create a unified score for each data point from the  $m$  base detectors. As in all the other chapters, the dimensionality of the data set is assumed to be  $d$ .

The first step in any ensemble combination is that of score normalization to account for the fact that the different algorithms may use scores on different scales, or they might even have a different ordering of the scores. For example, a  $k$ -nearest neighbor algorithm would typically output scores on a different scale than the LOF algorithm. Similarly, an EM-algorithm might output fit values in which outliers would tend to have lower scores. On the other hand, most algorithms assume the opposite convention. Therefore, it is important to normalize scores, so that one can meaningfully combine various algorithms without inadvertently over-weighting one of the algorithms.

The simplest approach to the ordering issue is to flip the sign of the scores. By flipping the sign of the scores, an output in which smaller scores indicate greater outlier tendency will be converted into an output in which larger scores indicate greater outlier tendency. Subsequently, the scores can be converted into comparable values by using one of the following two methods:

1. **Range-based scaling:** In this case, the maximum and minimum scores of each detector are computed. Let these values for the  $j$ th detector be  $max_j$  and  $min_j$  respectively. Then, for the  $i$ th data point, the score  $s_j(i)$  of the  $i$ th point by the  $j$ th detector is normalized to the following scaled value  $S_j(i)$ :

$$S_j(i) = \frac{s_j(i) - min_j}{max_j - min_j} \quad (6.1)$$

The resulting score will lie between 0 and 1 in each case. One problem with this approach is that the values of the scores will depend crucially on the values of  $max_i$  and  $min_i$ . In most cases, the point taking on the score  $max_i$  is the strongest outlier in the data, whose score is rather unstable and can be much larger than those of the other points. This can sometimes reduce the discrimination among the remaining scores in a drastic way, even if many of them happen to be outliers.

2. **Standardization:** Standardization is a more reliable approach for converting the scores into normalized values. In effect, the approach converts the scores into the Z-values introduced in Chapter 2. Let  $\mu_j$  and  $\sigma_j$  be the mean and standard deviation of the scores returned by the  $j$ th detector. Then, the standardized score  $S_j(i)$  of the  $i$ th data point by the  $j$ th detector is defined as follows:

$$S_j(i) = \frac{s_j(i) - \mu_j}{\sigma_j} \quad (6.2)$$

Note that standardization uses the assumption that the 1-dimensional scores follow a normal distribution. Although this assumption is almost never true, this type of normalization can often provide reasonably robust results. More complex assumptions on the distributions of the scores have the drawback of being too sensitive to the specific characteristics of the data set at hand.

Another approach, which is discussed in [213], is to convert scores into probabilities with the use of the EM-algorithm. Such an approach is discussed in detail in section 2.4.4 of Chapter 2. However, there is an important detail that needs to be kept in mind when combining probabilistic outputs via addition in an ensemble method. In such cases, additional steps are required to transform the scores to a more discriminative scale. For example, in some settings (such as the fit computation with the EM-algorithm), the fit drops off exponentially with increasing tendency to be an outlier. This is because the fit value exponentiates the squared Mahalanobis distance inside a Gaussian probability density function. In such cases, it may become difficult to differentiate between scores of abnormal points, whereas two normal points may be differentiated well. This is the opposite of what we want. For example, there is virtually no numerical distance between a weak outlier with fit probability of  $10^{-2}$  and a strong outlier with a fit probability of  $10^{-4}$ . Furthermore, there is a modest difference between the fit probability of  $10^{-4}$  (strong outlier) and a fit probability of 0.2 (normal point). However, the difference between two normal points with fit values of 0.2 and 0.5 is higher than both the previous cases. In such cases, the ensemble components in which outliers are erroneously treated as normal points will dominate an additive combination of base-detector scores. Clearly, this can be a problem in any setting where the detector outputs probabilities. These situations should be addressed with the use of a logarithmic function on the scores before combination [31]. The use of the logarithms on probability outputs can also be theoretically justified because log-likelihoods are inherently additive. Summing the logarithms in a score combination function is equivalent to using the product of the arguments inside the logarithm function. One can therefore view the ensemble score as a product of the corresponding probabilities just like a naive Bayes method. The robustness of the averaged log-likelihood fit has also been observed in [184].

After the scores of various detectors have been computed, they are combined into a unified score. Although we will discuss the combination functions in more detail in section 6.6, we provide a brief overview of two commonly used combination functions (with their intuition) in order to facilitate further discussion. The two most commonly used combination functions are averaging and the maximum. These combination functions are defined as follows:

1. **Averaging:** In this case, the outlier score of a data point is its mean score over various detectors. In other words, the score of the  $i$ th data point is computed as follows:

$$\text{Average}(i) = \frac{\sum_{j=1}^m S_j(i)}{m} \quad (6.3)$$

As we will discuss later in section 6.6, the main effect of averaging is to reduce the variance of the scores and thereby improve accuracy. This is a well-known result from the classification domain [617], and generalizes trivially to outlier ensembles because of the corresponding extension [32] of bias-variance theory to outlier detection.

2. **Maximum:** In this case, the outlier score of a data point is its maximum score over various detectors. In other words, the score of the  $i$ th data point is computed as follows:

$$\text{Maximum}(i) = \max_{j=1}^m S_j(i) \quad (6.4)$$

The effect of the maximization function is somewhat more complex because it can often improve bias but increase variance. This makes the overall effect unpredictable. Nevertheless, we can often gain the advantages of the bias-centric improvements by combining it with the averaging function. These theoretical effects and the advanced combination functions will be discussed in section 6.6.

Rank-centric variants of these combination functions can also be developed. These rank-centric variants are sometimes more stable, but the specific performance will depend on the detector at hand.

## 6.3 Theoretical Foundations of Outlier Ensembles

---

The bias-variance trade-off is often used in the context of supervised learning. Although it might seem at first sight that labels are required to quantify the bias-variance trade-off, it turns out that bias-variance theory can also be adapted to outlier detection with some modifications [32]. The trade-off is somewhat more abstract in the outlier-detection setting because of the unavailability of labels. Nevertheless, it has been shown [32] that the similarities in the theoretical analysis in the two cases lead to easier adaptation of many types of classification ensembles to outlier analysis. The discussion of this section is fully based on this recent work.

Most outlier detection algorithms output scores to quantify the “outlierness” of data points. After the scores have been computed, they can be converted to binary labels. Note that the bias-variance trade-off can be developed either for the scoring version of the problem or it can be developed for the binary labeling version of the problem. It is more interesting to develop this trade-off for the scoring version of the problem because most outlier ensemble algorithms combine the scores from different algorithms. An important observation about outlier scores is that they are *relative*. In other words, if all scores are multiplied by the same positive quantity, or translated by the same amount, it does not change various metrics (e.g., area under curve [AUC] of receiver operating characteristic curves [ROC]) of the outlier detector, which depend only on the ranks of the scores. This creates a challenge in quantifying the bias-variance trade-off for outlier analysis because the uniqueness of the score-based output is lost. The traditional notion of the bias-variance trade-off works with the mean-squared error (MSE). A quantification like the MSE is designed only for dependent variables with a clear mathematical interpretation of their absolute value (rather than only a relative interpretation). Measures such as the ROC AUC provide only an incomplete interpretation of the scores (in terms of relative ranks). It is possible to work with crisper definitions of the scores that allow the use of more conventional error measures like MSE. One such approach, which preserves uniqueness of scores, is that the outlier detectors always output standardized scores with zero mean, unit variance, and a crisp probabilistic interpretation. Note that one can always apply [31] a standardization step as a post-processing phase to any outlier detector without affecting the ROC; this also has a natural probabilistic interpretation (discussed below).

Consider a data instance denoted by  $\overline{X}_i$ , for which the outlier score is modeled using the training data  $\mathcal{D}$ . It is assumed that all training data points are generated from some base distribution, which is unavailable to the analyst in practice. We can assume that an ideal outlier score  $y_i$  exists for this data point, even though it is unobserved. The ideal score is output by an unknown function  $f(\overline{X}_i)$ , and it is assumed that the scores that are output by this ideal function also satisfy the zero mean and unit variance assumption over all possible



points generated by the base data distribution:

$$y_i = f(\overline{X}_i) \quad (6.5)$$

The interpretation of the score  $y_i$  is that by applying the (cumulative) standard normal distribution function to  $y_i$ , we obtain the relative outlier rank of  $\overline{X}_i$  with respect to all possible points generated by the base data distribution. In a sense, this crisp definition directly maps the score  $y_i$  to its (percentile) outlier rank in  $(0, 1)$ . In this sense,  $f(\overline{X}_i)$  is like an oracle that cannot be computed in practice; furthermore, in unsupervised problems, we do not have any examples of the output of this oracle.

This score  $y_i$  can be viewed as the analog to a numeric class variable in classification and regression modeling. In problems like classification, we add an additional term to the right-hand side of Equation 6.5 corresponding to the *intrinsic noise* in the dependent variable. However, unlike classification, where the value of  $y_i$  is a part of the *observed* data for training points, the value  $y_i$  in unsupervised problems only represents a theoretically ideal value (obtained from an oracle) which is *unobserved*. Therefore, in unsupervised problems, the labeling noise<sup>1</sup> no longer remains relevant.

Since the true model  $f(\cdot)$  is unknown, the outlier score of a test point  $\overline{X}_i$  can only be estimated with the use of an outlier detection model  $g(\overline{X}_i, \mathcal{D})$  using base data set  $\mathcal{D}$ . The model  $g(\overline{X}_i, \mathcal{D})$  is only a way of approximating the unknown function  $f(\overline{X}_i)$ , and it is typically computed algorithmically. For example, in  $k$ -nearest neighbor outlier detectors, the function  $g(\overline{X}_i, \mathcal{D})$  is defined as follows:

$$g(\overline{X}_i, \mathcal{D}) = \alpha \text{KNN-distance}(\overline{X}_i, \mathcal{D}) + \beta \quad (6.6)$$

Here,  $\alpha$  and  $\beta$  are constants which are needed to standardize the scores to zero mean and unit variance in order to respect the constraint on the absolute interpretation of the outlier scores. It is important to note that the  $k$ -nearest neighbor distance,  $\alpha$ , and  $\beta$  depend on the specific data set  $\mathcal{D}$  at hand. This is the reason that the data set  $\mathcal{D}$  is included as an argument of  $g(\overline{X}_i, \mathcal{D})$ .

If the function  $g(\overline{X}_i, \mathcal{D})$  does not properly model the true oracle  $f(\overline{X}_i)$ , then this will result in errors. This is referred to as *model bias* and it is directly analogous to the model bias used in classification. For example, the use of  $k$ -nearest neighbor algorithm as  $g(\overline{X}_i, \mathcal{D})$ , or a specific choice of the parameter  $k$ , might result in the user model deviating significantly from the true function  $f(\overline{X}_i)$ . Similarly, if we use a linear outlier detection model to score a setting in which the normal data points are arranged in a nonlinear spiral (like Figure 5.7 of Chapter 5), the model will have a high level of bias. A second source of error is the *variance*. The variance is caused by the fact that the outlier score directly depends on the *specific instantiation* of the data set  $\mathcal{D}$  at hand. Any data set is a finite set (assumed to be drawn from some unknown base distribution). Different instantiations of these draws will cause different results for the same outlier detection algorithm. Even if the *expected* value of  $g(\overline{X}_i, \mathcal{D})$  correctly reflects  $f(\overline{X}_i)$ , the estimation of  $g(\overline{X}_i, \mathcal{D})$  with limited data would likely not be exactly correct because of these varied results from different draws. If the data set  $\mathcal{D}$  is relatively small, then the variance in the estimation of  $g(\overline{X}_i, \mathcal{D})$  will be significant and it will have detrimental results on the accuracy. In other words,  $g(\overline{X}_i, \mathcal{D})$  will not be the same as  $E[g(\overline{X}_i, \mathcal{D})]$  over the space of various random choices of training data sets  $\mathcal{D}$ . This phenomenon is also sometimes referred to as *overfitting*. The model variance is high when

<sup>1</sup>If there are errors in the feature values, this will also be reflected in the hypothetically ideal (but unobserved) outlier scores. For example, if a measurement error causes an outlier, rather than an application-specific reason, this will also be reflected in the ideal but unobserved scores.

the same point receives very different scores across different choices of training data sets drawn from the same base distribution.

Although one typically does not distinguish between training and test points in unsupervised problems, one can easily do so by cleanly separating the points used for model building, and the points used for scoring. For example, a  $k$ -nearest neighbor detector would determine the  $k$  closest points in the training data for any point  $\bar{X}_i$  in the test data. We choose to demarcate training and test data because it makes our analysis intuitively similar to that of classification; however, it can be easily adapted<sup>2</sup> to draw approximately the same basic conclusions.

Let  $\mathcal{D}$  be the training data, and  $\bar{X}_1 \dots \bar{X}_n$  be a set of  $n$  test points whose (hypothetically ideal but unobserved) outlier scores are  $y_1 \dots y_n$ . It is assumed that these out-of-sample test points remain fixed over different instantiations of the training data  $\mathcal{D}$ , so that one can measure statistical quantities such as the score variance. We use an unsupervised outlier detection algorithm that uses the function  $g(\cdot, \cdot)$  to *estimate* these scores. Therefore, the resulting scores of  $\bar{X}_1 \dots \bar{X}_n$  using the training data  $\mathcal{D}$  are  $g(\bar{X}_1, \mathcal{D}) \dots g(\bar{X}_n, \mathcal{D})$ , respectively. The mean-squared error, or MSE, of the detectors of the test points over a particular realization  $\mathcal{D}$  of the training data is obtained by the averaging the squared errors over different test points:

$$MSE = \frac{1}{n} \sum_{i=1}^n \{y_i - g(\bar{X}_i, \mathcal{D})\}^2 \quad (6.7)$$

The *expected MSE*, over different realizations of the training data, generated using some random process, is as follows:

$$E[MSE] = \frac{1}{n} \sum_{i=1}^n E[\{y_i - g(\bar{X}_i, \mathcal{D})\}^2] \quad (6.8)$$

The different realizations of the training data  $\mathcal{D}$  can be constructed using any crisply defined random process. In the traditional view of the bias-variance trade-off, one might assume that the data set  $\mathcal{D}$  is generated by a hidden process that draws it from a true distribution. The basic idea is that *only one instance of a finite data set* can be collected by the entity performing the analysis, and there will be some variability in the results because of this limitation. This variability will also lead to some loss in the accuracy over a setting where the entity actually had access to the distribution from which the data was generated. To the entity that is performing the analysis, this variability is hidden because they have only one instance of the finite data set. Other unconventional interpretations of the bias-variance trade-off are also possible. For example, one might construct each instantiation of  $\mathcal{D}$  by starting with a larger base data set  $\mathcal{D}_0$  and use random subsets of points, dimensions, and so on. In this alternative interpretation, the expected value of the mean-squared error is computed over different instantiations of the random process extracting  $\mathcal{D}$  from  $\mathcal{D}_0$ . Finally, one might even view the randomized process of extracting  $\mathcal{D}$  from  $\mathcal{D}_0$  as a part of the base detector. This will yield a randomized *base detector*  $g(\bar{X}_i, \mathcal{D}_0)$ , but a fixed data set  $\mathcal{D}_0$ . Therefore, the random process is now defined with respect to the randomization in base detector, rather than the training data selection process. These different interpretations will provide different bias-variance decompositions (see section 6.3.1). It is important to clearly

---

<sup>2</sup>It is noteworthy that the most popular outlier detectors are based on distance-based methods. These detectors are lazy learners in which the test point is itself never included among the  $k$ -nearest neighbors at prediction time. Therefore, these learners are essentially out-of-sample methods because they do not include the test point within the model (albeit in a lazy way).

define the underlying random process in order to properly analyze the effectiveness of a particular ensemble method. For now, we will work with the conventional interpretation that the random process generates the different training data sets from a base distribution.

Note that even though the training data  $\mathcal{D}$  might have different instantiations because it is generated by a random process, the test points  $\bar{X}_1 \dots \bar{X}_n$  always remain fixed over all instantiations of the random process. This is the reason that we have chosen to demarcate the training and test data; it allows us to evaluate the effects of changing the training data (with a random process) on the same set of test points. If the predictions of the same test points vary significantly over various instantiations of the random process, we say that the model has high *variance*. On the other hand, if the expected prediction of each test point is poor, we say that the model has high *bias*. The basic idea is to decompose the error of the classifier into these two components.

The term in the bracket on the right-hand side of Equation 6.8 can be re-written as follows:

$$E[MSE] = \frac{1}{n} \sum_{i=1}^n E[\{(y_i - f(\bar{X}_i)) + (f(\bar{X}_i) - g(\bar{X}_i, \mathcal{D}))\}^2] \quad (6.9)$$

Note that we can set  $(y_i - f(\bar{X}_i))$  on the RHS of aforementioned equation to 0 because of Equation 6.5. Therefore, the following can be shown:

$$E[MSE] = \frac{1}{n} \sum_{i=1}^n E[\{f(\bar{X}_i) - g(\bar{X}_i, \mathcal{D})\}^2] \quad (6.10)$$

This right-hand side can be further decomposed by adding and subtracting  $E[g(\bar{X}_i, \mathcal{D})]$  within the squared term:

$$\begin{aligned} E[MSE] &= \frac{1}{n} \sum_{i=1}^n E[\{f(\bar{X}_i) - E[g(\bar{X}_i, \mathcal{D})]\}^2] \\ &\quad + \frac{2}{n} \sum_{i=1}^n \{f(\bar{X}_i) - E[g(\bar{X}_i, \mathcal{D})]\} \{E[g(\bar{X}_i, \mathcal{D})] - E[g(\bar{X}_i, \mathcal{D})]\} \\ &\quad + \frac{1}{n} \sum_{i=1}^n E[\{E[g(\bar{X}_i, \mathcal{D})] - g(\bar{X}_i, \mathcal{D})\}^2] \end{aligned}$$

The second term on the right-hand side of the aforementioned expression evaluates to 0. Therefore, we have:

$$E[MSE] = \frac{1}{n} \sum_{i=1}^n E[\{f(\bar{X}_i) - E[g(\bar{X}_i, \mathcal{D})]\}^2] + \frac{1}{n} \sum_{i=1}^n E[\{E[g(\bar{X}_i, \mathcal{D})] - g(\bar{X}_i, \mathcal{D})\}^2] \quad (6.11)$$

$$= \underbrace{\frac{1}{n} \sum_{i=1}^n \{f(\bar{X}_i) - E[g(\bar{X}_i, \mathcal{D})]\}^2}_{\text{Bias}^2} + \underbrace{\frac{1}{n} \sum_{i=1}^n E[\{E[g(\bar{X}_i, \mathcal{D})] - g(\bar{X}_i, \mathcal{D})\}^2]}_{\text{Variance}} \quad (6.12)$$

The first term in the aforementioned expression is the (squared) bias, whereas the second term is the variance. Stated simply, one obtains the following:

$$E[MSE] = \text{Bias}^2 + \text{Variance} \quad (6.13)$$

This derivation is very similar to that in classification, although the intrinsic error term is missing because of the ideal nature of the score output by the oracle. The bias and variance are specific not just to the algorithm  $g(\overline{X}_i, \mathcal{D})$  but also to the random process used to create the training data sets  $\mathcal{D}$ .

Although this analysis does not seem to be general because it makes the assumption of zero mean and unit variance on the scores, it holds as long as the outputs of the base detector and oracle have the same mathematical interpretation. For example, we could very easily have made this entire argument under the assumption that both the base detector  $g(\overline{X}_i, \mathcal{D})$  and the oracle  $f(\overline{X}_i)$  directly output the relative ranks in  $(0, 1)$ . In that case, the ensemble would work with the ranks as the base detector output.

### 6.3.1 What is the Expectation Computed Over?

Note that the bias-variance condition of Equation 6.13 provides a condition in terms of the expected value over a set of random variables. Therefore, a natural question to investigate is the nature of the random process that generates these random variables. The *traditional* view of the bias-variance trade-off is that the random process draws the training data sets from some base distribution which is unknown to the analyst. In fact, the analyst sees only *one finite instantiation* of the data set and the random process is hidden from her. Therefore, there is a hidden variance of the predicted score of a particular test instance (over the prediction that another analyst may obtain on that test instance with a different instantiation of the training data). This hidden variance increases the error of the ensemble method. If the analyst were given access to the base distribution instead of a single instance of the training data, it is theoretically possible for her to reduce the variance to 0 by repeatedly scoring the test point over different generated instantiations of the training data and then averaging the scores. On the other hand, the bias is inherent to the *choice of the model* used by the analyst. In other words, even if the analyst were given access to the base distribution and could generate an unlimited number of data sets, she would still be left with the bias because of the error in *assumptions* made by the model.

There are, however, other non-traditional views of the bias-variance trade-off that are relevant to other types of ensemble algorithms. It is not necessary to compute the bias and variance over different choices of the training data. One can also choose the random process to define the base detector; for example, an isolation forest or a feature bagging approach makes randomized choices during its execution. In such cases, one can also define the random process in terms of the random choices made by the algorithm, and define a variance over such settings. Such a variance is referred to as *model-centric* variance, whereas the first type of variance is referred to as *data-centric* variance. These different views of the bias-variance trade-off are useful in different settings. For example, some ensemble methods such as subsampling are best explained from a data-centric perspective (i.e., random process draws the data set from an unknown base distribution), whereas other ensemble methods are best explained from a model-centric perspective. A detailed discussion of these issues is provided in [35].

### 6.3.2 Relationship of Ensemble Analysis to Bias-Variance Trade-Off

Ensemble analysis is a way of combining different models in order to ensure that the bias-variance trade-off is optimized. In general, one can view the prediction  $g(\overline{X}, \mathcal{D})$  of a base detector as a random variable, depending on a random process over either the selection of the

base data  $\mathcal{D}$ , or the construction of the detector  $g(\cdot, \cdot)$  itself, which might be randomized. The overall mean-squared error of this random variable is reduced with respect to the unknown oracle output  $f(\bar{X})$  by the ensemble process. This is achieved in two ways:

1. *Reducing bias*: Some methods such as boosting reduce bias in classification by using an ensemble combination of highly biased detectors. The design of detectors is based on the performance results of earlier instantiations of the detector in order to encourage specific types of bias performance in various components. The final combination is also carefully designed in a weighted way to gain the maximum advantage in terms of overall bias performance. However, it is generally much harder to reduce bias in outlier ensembles because of the absence of ground truth.
2. *Reducing variance*: Methods such as bagging, bragging, wagging, and subbagging (subsampling) [105, 106, 107], can be used to reduce the model-specific variance in classification. In this context, most classification methods generalize *directly* to outlier ensembles. In most of these methods the final ensemble score is computed as an average of the scores of various detectors. The basic idea is that the average of a set of random variables has lower variance. Therefore, the use of the averaging combination can be credited to the original work in classification [98, 105, 266]; however, it was subsequently adopted by most other outlier ensemble methods [344, 367].

The “unsupervised” nature of outlier detection does not mean that bias and variance cannot be defined. *It only means that the dependent variables are not available with the training data, even though an “abstract,” but unknown ground truth does exist.* However, the bias-variance trade-off does not rely on such an availability *to the base algorithm*. None of the steps in the aforementioned computation of MSE rely on the need for  $g(\bar{X}_i, \mathcal{D})$  to be computed using examples of the output of oracle  $f(\cdot)$  on points in  $\mathcal{D}$ . This is the reason that variance-reduction algorithms for classification generalize so easily to outlier detection.

## 6.4 Variance Reduction Methods

---

The general idea in variance reduction is to average the scores obtained from multiple instantiations of a randomized algorithm. The randomization might be caused by randomized choices of the data (as in bagging and subsampling), or it might be caused by randomized choices of the model (as in isolation forests). In each case, the effect of averaging results in variance reduction and therefore provides improved performance. This is a well-known result in the classification domain [617], and it applies directly to outlier ensembles because of similar theoretical foundations in the two problems [32]. In general, *unstable* base detectors, in which the outlier scores/ranks vary significantly between different runs provide the best improvements. However, better *incremental* improvements do not always mean that the final ensemble performance will be better.

In order to show the “typical” relative performance of some randomized base detectors and ensemble combination, we will use an example. A hypothetical example of various randomized detectors together with the area under curve (AUC) of their receiver operating characteristic (ROC) is shown in Figure 6.1. The figure shows the hypothetical AUC values for 200 randomized runs of the base detector for each of three algorithms. A box plot<sup>3</sup> is used to summarize the 200 different AUC values. The ensemble performance is also shown with a square marker, and it generally performs better than most of the base detectors.

<sup>3</sup>See section 2.2.2.3 of Chapter 2 for a description of box plots.

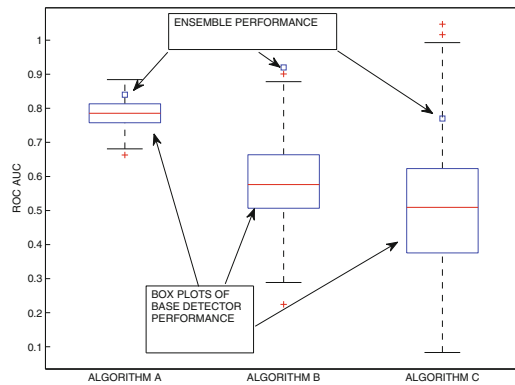


Figure 6.1: Illustrative example of base detector (box plot) and ensemble (square marker) performances for base detectors with varying levels of stability. Relative ensemble performance does not always reflect the relative base-detector performance. Unstable algorithms like algorithms ‘B’ and ‘C’ with thicker box plots lead to better incremental improvements.

This is because the base detectors include a variance component, which is significantly reduced in the ensemble combination. Unstable algorithms often<sup>4</sup> have “thicker” box plots (i.e., large inter-quartile ranges between the upper and lower ends of the box). As shown in Figure 6.1, thicker box plots often lead to larger *incremental* improvements. For example, algorithm ‘B’ has much better incremental improvement than algorithm ‘A,’ when compared with its median base detector performance. However, larger incremental improvements do not always lead to be the best ensemble performance. Even though algorithm ‘C’ has a larger incremental improvement than algorithm ‘A,’ it does not outperform algorithm ‘A’ in terms of the final ensemble performance. This is primarily due to the poor bias-centric characteristics of algorithm ‘C.’ Therefore, the overall choice of the base-detector in such methods often depends on a trade-off between the perceived<sup>5</sup> quality of the detector and its stability.

In this section, we will discuss several variance reduction methods for outlier detection. Although some of these methods, such as feature bagging, rotated bagging, and isolation forests, have already been discussed in the chapter on high-dimensional data, we will include discussions on these methods for completeness. We will also provide a slightly different perspective on these methods from the point of view of ensemble analysis. The discussion will also clarify the connections between ensemble analysis and subspace methods for high-dimensional outlier detection.

### 6.4.1 Parametric Ensembles

The most straightforward application of variance reduction is that of addressing the issue of parameter choice in outlier detection algorithms. For example, in a  $k$ -nearest neighbor algorithm, it is often impossible for an analyst to determine the correct value of  $k$  for a

<sup>4</sup>Strictly speaking, instability is measured in terms of the outlier scores rather than the AUC. Nevertheless, instability in scores is often reflected in the AUC as well.

<sup>5</sup>In unsupervised problems, it is impossible to measure accuracy in a concrete way because labels are unavailable for computing AUC in real application settings.

**Algorithm** *ParameterEnsemble*(Data Set:  $\mathcal{D}$ , Ensemble Components:  $T$ );

```

begin
  { Parameters of algorithm  $\mathcal{A}$  are  $\theta_1 \dots \theta_r$  }
   $t = 1$ ;
  for each  $\theta_i$  identify “reasonable” range  $[min_i, max_i]$ 
  repeat
    for each  $i$  select  $\theta_i$  randomly from  $[min_i, max_i]$ ;
    Compute score vector  $\mathcal{S}(t)$  by executing algorithm  $\mathcal{A}(\mathcal{D}, \theta_1 \dots \theta_r)$ ;
    Standardize score vector  $\mathcal{S}(t)$ ;
     $t = t + 1$ ;
  until ( $t = T$ );
  return averaged score vector  $\frac{\sum_{i=1}^T \mathcal{S}(i)}{T}$ ;
end

```

Figure 6.2: Using ensembles for reducing parameter-centric uncertainty in unsupervised outlier detection

particular algorithm. This is a particularly vexing problem in unsupervised problems like outlier detection, because no ground-truth is available to validate the effectiveness of a particular choice of parameters. In such cases, the use of a particular choice of parameters is no better than random guessing, and this results in an implicit *model-centric* variance in the output.

A typical ensemble approach [31] is to use a range of different values of  $k$ , and then average the scores. Although the overall performance might not be as good as that of using the best value of  $k$ , it is usually better than the median performance over the different values of  $k$  that are tested. This broad approach is applicable to virtually any parameterized algorithm. In cases, where multiple parameters are available, one might sample the parameters in a “reasonable” range of values, and return the averaged values of the scores. Note that the analyst still has the responsibility of identifying a “reasonable” range of values, although this is usually much easier to estimate (based on data set size and dimensionality) than the setting in which the precise value of a parameter needs to be identified on a particular data set. After the algorithm has been applied over the various instantiations of the parameters, the scores of each point are averaged. The overall approach is illustrated in Figure 6.2. The resulting detector has reduced variance than that of a detector that selects one of these reasonable choices of parameters randomly. It is noteworthy that this particular view of variance reduction is over different randomized instantiations of the base detector rather than over different randomized draws of the training data from the base distribution. Therefore, such an approach reduces model-centric variance.

Although the parametric-ensemble in Figure 6.2 samples over various choices of parameters, this is needed only in cases where the parameter space is too large to try all choices. In cases where the parameter space is small, one might simply run the algorithm over all reasonable choices of parameters and average the scores. Good results have been shown in [184] by averaging the outlier scores from multiple mixture-model clusterings, each of which is generated using a different set of parameters. In general, methods like clustering and histograms are so sensitive to the choice of parameters that it makes sense to use them only in this type of ensemble-centric setting.

**Algorithm** *FeatureBagging*(Data Set  $\mathcal{D}$ , Ensemble Components:  $m$ );  
**begin**  
  **repeat**  
    Sample an integer  $r$  from  $\lfloor d/2 \rfloor$  to  $d - 1$ ;  
    Select  $r$  dimensions from the data  $\mathcal{D}$  randomly to  
      create an  $r$ -dimensional projection;  
    Use base detector on projected representation to compute scores;  
  **until**  $m$  iterations;  
  Report score of each point as a combination function  
    of its  $m$  scores from different subspaces;  
  { The most common combinations are maximization and averaging }  
**end**

Figure 6.3: Feature Bagging

### 6.4.2 Randomized Detector Averaging

Many outlier detection models are inherently randomized because they depend on the use of randomly chosen initialization points. For example, if a data point is scored using its distance to the closest centroid of a  $k$ -means algorithm, the scores may vary significantly from one execution to the next because of the effect of the initialization point. This is a manifestation of model-centric variance, which reduces the detector accuracy. Therefore, in all such cases, it is extremely important to run the model multiple tries and average the scores in order to reduce the effect of variance on modeling accuracy. In many cases, parametric ensemble methods are paired with this type of randomization to increase diversity. Many outlier detection models, such as density estimation, mixture-modeling, clustering and histograms, are inherently suitable for this type of paired setting. This is because many of these models have parameters and initialization points to which the performance is very sensitive (i.e., high model-centric variance).

### 6.4.3 Feature Bagging: An Ensemble-Centric Perspective

Feature bagging has been discussed in the context of high-dimensional data in section 5.2.3 of Chapter 5. Here, we provide an ensemble-centric view. The feature bagging method [344] samples different subsets of dimensions. The basic idea is to sample a number  $r$  between  $\lfloor d/2 \rfloor$  and  $d - 1$ , and then select  $r$  dimensions randomly from the data set. The base detector is applied to this lower-dimensional projection. The scores across different base detectors are then combined with the use of either the maximization or the averaging combination function. The former is used in an indirect way by ranking the data points in each component by their scores, and reporting the best rank of each point over all detectors. A pseudocode of the feature bagging method is illustrated in Figure 6.3.

Although feature-bagging might seem like a data-centric ensemble method, it is better explained using a model-centric view of the bias-variance trade-off. Feature bagging (with averaging) is a method that reduces (model-centric) detector variance. Feature bagging with a particular subset of dimensions has a data-centric bias that depends on the selected dimensions. However, if one views the step of randomly selecting the subset of dimensions as a *part* of the component detector, then each such (randomized) detector has exactly the same model-centric bias, and the aforementioned variability in the bias across different



dimension-specific instantiations now becomes a part of this (randomized) detector’s model-centric variance. Note that the random process for quantifying the bias-variance trade-off in this case is over the different randomized base detectors, which are applied to a fixed data set  $\mathcal{D}$ . In such cases, using an average combination is able to achieve variance reduction. The smaller the subset of dimensions that are selected, the greater the variance reduction that is achieved. This is because the underlying detectors tend to be relatively uncorrelated if few overlapping dimensions are selected by different detectors. However, if all dimensions are informative, the bias characteristics of such an approach are likely to work against feature bagging because down-selecting the dimensions will lose information. Therefore, there are subtle trade-offs between the bias and variance with the use of the feature bagging method.

### 6.4.3.1 Connections to Representational Bias

Even though feature bagging uses global subspaces within each base component, the ensemble combination is implicitly able to discover *locally relevant subspaces* as in subspace methods [4]. The reason for this is that *ensemble methods are often able to discover solutions that are more general than their individual components*. This point can be explained with Dietterich’s notion of *representational bias* [170]. For example, we have repeated a pictorial example from Chapter 5 in Figure 6.4. In this case, point ‘A’ is an outlier in view 1, whereas point ‘B’ is an outlier in view 4. One can consider each of these views as a hypothesis about the model that best exposes the outliers. For example, the space  $\mathcal{H}$  of all hypothesis being considered by the global projections of feature bagging is shown in Figure 6.4(e). However, neither of these hypotheses can expose both outliers ‘A’ and ‘B’ because of the inherent representational bias of the space of hypotheses being considered. In fact, the true hypothesis does not even lie in the space of all hypotheses (i.e., hypotheses of global projections) being considered by the feature bagging method. This situation is shown in Figure 6.4(e), where hypothesis  $h_1$  corresponds to view 1, whereas the hypothesis  $h_6$  corresponds to view 4. Because the true hypothesis  $f$  lies outside the space of all hypotheses being considered by global projections, one cannot discover both the outliers ‘A’ and ‘B’ in a single global projection even if one had access to a data set of infinite size. Nevertheless, the averaging or the maximization functions over different projections are often able to expose both outliers, and are therefore much closer to the true hypothesis  $f$  especially if many of these components are able to score these outliers properly.

The averaging function is better able to approximate the true hypothesis by converting the variability in *data-centric* bias across different hypothesis in  $\mathcal{H}$  into a *model-centric* variance. This model-centric variance is computed over the randomized process<sup>6</sup> of generating different feature subsets, which can be reduced by averaging. It is noteworthy that variance reduction is a subtle concept and may be computed differently, depending on how one understands the random process over which the bias and variance are computed. By using different random processes, one can obtain different decompositions of the error into the bias and variance. In this specific example, we have shown that it is sometimes possible to use model-centric variance reduction to reduce data-centric (representational) bias by converting the variability of bias (over the different instantiations of the constituent models) into a portion of the model-centric variance. These notions are discussed in greater detail in [35].

---

<sup>6</sup>Recall that the data-centric bias is computed over the randomized process of generating different training data sets. In traditional bias-variance theory, this is the conventional understanding of bias. However, it is often helpful to interpret the bias-variance theorem in a more generic way over different types of random processes.

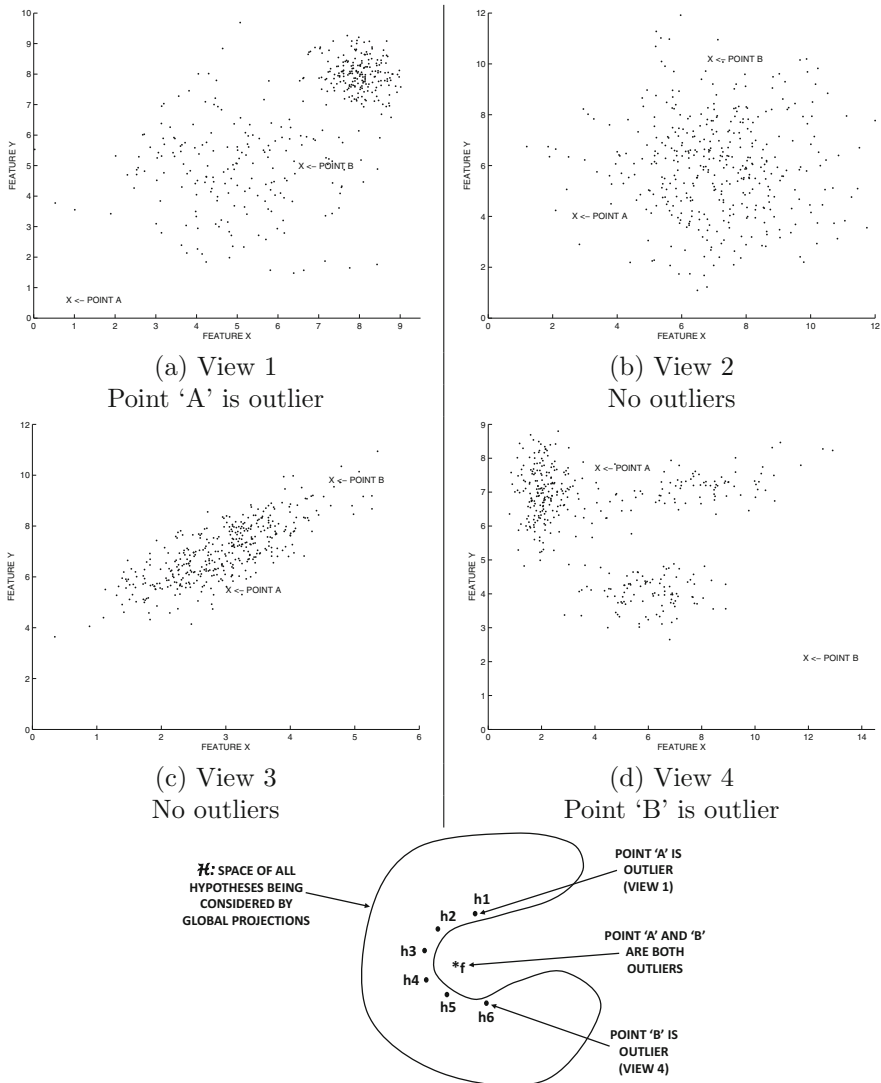


Figure 6.4: The ensemble can reduce representational bias by capturing hypotheses that are more powerful than their individual members. It is possible to use variance reduction on the variability in representational bias.

### 6.4.3.2 Weaknesses of Feature Bagging

The feature bagging method proposes to always randomly select between  $\lfloor d/2 \rfloor$  and  $d - 1$  dimensions; one does not always gain the best variance reduction by selecting so many dimensions because of correlations among different detectors. Correlations among detectors hinder variance reduction. One might even select the same subset of dimensions repeatedly, while providing drastically worse bias characteristics. In particular, consider a 6-dimensional data set. The number of possible 3-dimensional projections is 20, the number of possible 4-dimensional projections is 15, and the number of 5-dimensional projections is 6. The total number of possibilities is 41. Therefore, most of the projections (and especially the 4- and 5-dimensional ones) will be repeated multiple times in a set of 100 trials, and not much variance can be reduced from such repetitions. On the other hand, the 3-dimensional projections, while more diverse in overlap and repetition, will have deteriorated bias characteristics. This will also be reflected in the final ensemble performance. Here, it is important to note that the most diverse dimensions provide the worst bias characteristics and vice versa. The rotated bagging method [32] solves some of these problems.

### 6.4.4 Rotated Bagging

The *rotated bagging* method<sup>7</sup> was proposed in [32], in which the data is rotated to a random axis system before selecting the features. By rotating the axis system and using a lower-dimensional projection, it is often possible to expose the outliers in at least some of the projections. The combination function is often able to discover the different outliers in different subspaces. Since real data has significant correlations, one can afford to use a much lower dimensionality than  $\lfloor d/2 \rfloor$  to represent the data without losing too much information. In real data sets, the implicit dimensionality usually does not grow much faster than  $\sqrt{d}$  with increasing dimensionality  $d$ . Therefore, a total of  $2 + \lceil \sqrt{d}/2 \rceil$  orthogonal directions are selected from the rotated axis-system as the set of relevant feature bags. The approach is not designed to work for three or less dimensions. Therefore, a constant value of 2 is added up front to prevent its use in such cases. The component detectors will be more uncorrelated in high-dimensional cases, which yields a better opportunity for variance reduction. The overall algorithm works as follows:

1. Determine a randomly rotated axis system in the data.
2. Sample  $r = 2 + \lceil \sqrt{d}/2 \rceil$  directions from rotated axis system. Project data along these  $r$  directions.
3. Run the outlier detector on projected data.

After running the detectors, the scores can be averaged with a primary goal of variance reduction. However, other schemes such as maximization can also be used. It is important to use standardization on the scores before the combination.

In order to determine the  $r = 2 + \lceil \sqrt{d}/2 \rceil$  randomly rotated mutually orthogonal directions, a  $d \times r$  random matrix  $Y$  is generated, such that each value in the matrix is uniformly distributed in  $[-1, 1]$ . Let the  $t$ th column of  $Y$  be denoted by  $\bar{y}_t$ . Then, the  $r$  random orthogonal directions  $\bar{e}_1 \dots \bar{e}_r$  are generated using a straightforward Gram-Schmidt orthogonalization of  $\bar{y}_1 \dots \bar{y}_r$  as follows:

---

<sup>7</sup>Although the rotated bagging scheme is described in Chapter 5, we repeat the description here for completeness of this chapter. In the modern publishing era, selective chapter-wise electronic access has become increasingly common.

1.  $t = 1$ ;  $\bar{e}_1 = \frac{\bar{y}_1}{|\bar{y}_1|}$
2.  $\bar{e}_{t+1} = \bar{y}_{t+1} - \sum_{j=1}^t (\bar{y}_{t+1} \cdot \bar{e}_j) \bar{e}_j$
3. Normalize  $\bar{e}_{t+1}$  to unit norm.
4.  $t = t + 1$
5. if  $t < r$  go to step 2

Let the resulting  $d \times r$  matrix with columns  $\bar{e}_1 \dots \bar{e}_r$  be denoted by  $E$ . The  $N \times d$  data set  $D$  is transformed and projected to these orthogonal directions by computing the matrix product  $DE$ , which is an  $N \times r$  matrix of  $r$ -dimensional points. The rotated bagging method can be viewed as a subspace outlier detection method in which the subspaces may be defined by any randomly chosen directions, rather than only features from the original space. Because of the difficulty in choosing discriminative features in arbitrary directions (especially for unsupervised problems like outlier detection), it is crucially important to exploit ensemble methods. It has been shown in [32] that rotated bagging can provide more accurate results than feature bagging. Rotated bagging can be explained in a similar way to feature bagging, except that the outliers are discovered in arbitrarily oriented subspaces of the underlying data. The reason that rotated bagging is effective in the context of high-dimensional data is described in section 5.3.3 of Chapter 5.

### 6.4.5 Isolation Forests: An Ensemble-Centric View

Isolation forests are discussed in detail in section 5.2.6 of Chapter 5 in the context of subspace outlier detection in high-dimensional data. Here, we revisit this discussion in the context of outlier ensembles. An isolation forest is constructed from multiple isolation trees. At a conceptual level, isolation forests also explore random subspaces in the different branches of a tree structure (like feature bagging). The score of a point quantifies the depth required to isolate a point in a single node of the tree with random splits. Outlier points are usually isolated quickly with a few splits. Since different branches of the tree use different splits, isolation forests explore random *local* subspaces unlike feature bagging, which explores a single global subspace in each ensemble component. Furthermore, the scoring is done in a more direct way in isolation trees by quantifying how easy it is to find a local subspace of low dimensionality in which a data point is isolated, and it is not dependent on the vagaries of a particular base detector like LOF (as in feature bagging). As a result, isolation forests are potentially more powerful than feature bagging.

In the most efficient version of isolation trees, a *subsample of the data* is used to recursively partition it with random splits. In each iteration, an attribute is randomly selected and a split point is randomly chosen between the minimum and maximum values of the attribute. The process of splitting is recursively repeated so as to isolate the instances into nodes with fewer and fewer instances until the points are isolated into singleton nodes containing one instance. In such cases, the tree branches containing outliers are noticeably less deep, because these data points are quite different from the normal data. Thus, data points that have noticeably shorter paths in the branches of different trees are more likely to be outliers. Out-of-sample points can also be scored by determining their relevant paths in the tree, based on the univariate split criteria at the various nodes. The process of subsampling adds to the diversity. The final combination step is performed by averaging the path

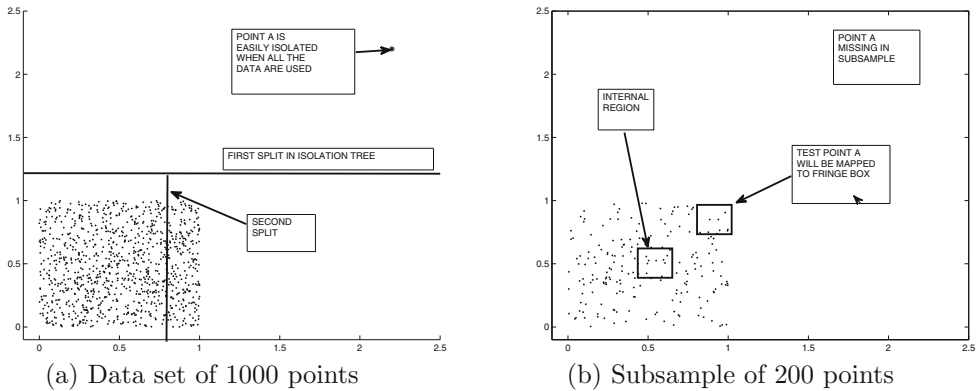


Figure 6.5: An outlier test point is mapped to a fringe region when it is not included in the subsample

lengths of the data points in the different trees of the isolation forest. Refer to section 5.2.6 of Chapter 5 for a more detailed discussion of isolation forests.

Although subsampling provides diversity benefits in the isolation forest, there are often some detrimental bias-centric effects of subsampling on the base detectors. These detrimental effects arise as a result of how out-of-sample points are scored in isolation forests. It is noteworthy that out-of-sample points might often map to empty regions in the subsample. From the perspective of the isolation tree (built on the subsample), this empty region might be merged with a normal region of the space, and therefore the score will correspond to the depth of this normal region. Nevertheless, in most cases, the empty regions often map to the *fringes* of normal regions. The fringes of normal regions also tend to be less dense and receive lower depth scores, which is desirable. There are indeed some occasional cases in which scoring strong outliers like fringe regions has detrimental accuracy effects. An example of a data set of 1000 points is illustrated in Figure 6.5(a), in which point ‘A’ is an obvious outlier. Such an outlier will often be isolated by the very first or second split in an isolation tree when it is included in the training data. However, if the point ‘A’ is missing from the subsample (cf. Figure 6.5(b)), the structure of the isolation tree will be defined by only the normal points. When the score of point ‘A’ is computed as a test point, it will be equal to the depth of a fringe box in the isolation tree, as shown in Figure 6.5(b). In cases where the isolation tree is constructed down to a single point, the score of the point ‘A’ will be equal to that of a fringe point in a cluster. Although fringe points are only weak outliers, the saving grace is that they will often have a lower depth compared to internal points *in expectation*. As a result, the isolation tree can still differentiate between outliers and normal points to a large extent. Nevertheless, it is clear that the outlier score of point ‘A’ in the out-of-sample setting will often be much weaker (i.e., have larger depth) than it ought to be. Although the approach in [367] is able to gain the diversity improvements from subsampling for some data sets, this improvement is not consistent. This because the diversity gains from the randomized process of tree construction are more significant than those of subsampling, and the difference in the treatment of out-of-sample points from in-sample points affects the subsampling component in a negative way.

The differential treatment of fringe points and interior points causes another problem in isolation forests for some pathological data sets; outliers in sparse but central regions of

the data set are sometimes scored<sup>8</sup> like inliers. In other words, the isolation forest tends to differentially favor specific types of outliers, such as the multivariate extreme values discussed in section 2.3.

### 6.4.6 Data-Centric Variance Reduction with Sampling

Bagging [98] and subsampling [105, 106, 107] are common methods used for *data-centric* variance reduction in classification. Although these methods were originally designed for the classification domain, they generalize easily to outlier detection because of significant theoretical similarities [32] between the classification and outlier detection domains. In *data-centric* variance reduction, the assumption is that an unknown base distribution exists from which all the data is generated, but the analyst only has access to only a finite sample of the data. Therefore, the prediction results observed by the analyst would be slightly different (i.e., *variant*) from another analyst who had access to a different sample from the same distribution (even if the same algorithm was used in both cases). This difference in results between the two analysts contributes to additional error (beyond modeling errors) in the form of variance. Therefore, the goal is to use this finite resource (data set) as *efficiently* as possible to minimize the variance.

Now consider an ideal setting in which the analyst actually had access to the base distribution, and could go back to it as many times as she wished in order to draw multiple samples. In such a case, each prediction (from a sample) would have a variance that is the second term of Equation 6.12. By drawing the sample an infinite number of times and averaging the results, the analyst could theoretically reduce the contribution of the second term in Equation 6.12 to 0 and therefore reduce the error by *variance reduction*. Unfortunately, however, analysts do not have access to the base distribution in practice. They only have access to a *single finite instance* of the base data. However, it turns out that applying the base detector once on this finite resource is not the most *efficient* way to minimize the variance term. Rather, one should use this finite resource to roughly simulate the aforementioned process of drawing samples from the base distribution. Two examples of such simulations are bagging and subsampling, which are closely related methods. Although such simulations are clearly imperfect (because a finite data set is being used), they reduce the error *most of the time* over a single application of the base detector [98] because of variance reduction. It has been theoretically shown in [98] that such a simulation reduces the error for unstable base detectors. A similar result has been shown in the case of outlier detection in [35].

#### 6.4.6.1 Bagging

In bagging, samples are drawn from the data with replacement. The size of the sample drawn is typically the same as base data, although some variants of bagging do not impose this restriction. Because of the process of sampling with replacement, some points may occur multiple times in the sample, whereas others might not occur at all. Such a sampling technique is also referred to as *bootstrapping*. The outlier detector is applied to each bootstrapped sample. For example, for a  $k$ -nearest neighbor detector, the original data is treated as the test data, whereas the  $k$ -nearest neighbors are determined from among the bootstrapped sample. For any given test point, care must be taken to exclude its copies in the (bootstrapped) training data while computing its  $k$ -nearest neighbors. Each point

---

<sup>8</sup>For example, a single outlier at the center of normal data points on a unit sphere in 10 dimensions can receive an inlier-like score [35].

is scored multiple times using detectors constructed on different bootstrapped samples and the scores of each point are averaged. The averaged scores provide the final result.

Note that the bootstrapping process is a simulation of the process of drawing data points from a base distribution. However, this simulation is imperfect because of two reasons:

1. One can no longer reduce the variance in the second term of Equation 6.12 to 0 because of correlations (overlaps) among the bootstrapped samples. Therefore, there is a (hidden) residual variance, which is not apparent to the analyst who only has access to a finite resource (data set) rather than the base distribution. The correlations among base detectors are rather large in the case of bagging with bootstrapped sample size equal to data size, and therefore a relatively large part of the variance cannot be reduced. It is often more rewarding to use bagging with smaller re-sample sizes because of less correlations among base detectors (and therefore better variance reduction).
2. Each sample no longer represents the base distribution faithfully. For example, the presence of so many repeated points is a manifestation of this imperfection. This would tend to have a detrimental bias-centric effect, although it is often quite small. However, in some (very occasional) settings, it is possible for these bias-centric effects to overwhelm the variance-reduction advantages.

In general, one can often improve the performance of most detectors with bagging. Some results are presented in [35], which show that bagging can indeed improve detector performance. To maximize the advantages of variance reduction, it is helpful to use unstable detectors, although it is not necessarily helpful to use an unstable base detector with very poor accuracy. After all, the absolute accuracy of the ensemble method is more important than its incremental improvement over the base detectors.

#### 6.4.6.2 Subsampling

Subsampling is frequently used in the classification domain [105, 106, 107] for variance reduction. In subsampling, samples are drawn from the data *without* replacement. Then, each point from the original data set is treated as a test point, and it is scored against the model constructed on the subsample, whether that point is included in the subsample or not. Therefore, much like classification, subsampling distinguishes between the training and test data, although the training data is a subset of the test data. When using an instance-based (i.e., lazy) approach like  $k$ -nearest neighbor, care must be taken to exclude the test point from among the  $k$ -nearest neighbors, if it is included in the training subsample. This is important to avoid overfitting. However, when using explicit generalization (such as isolation forests and one-class SVMs), it is impossible to exclude a specific test point from the model, which is typically constructed *up-front* using the entire subsample. Therefore, there will always be a small amount of over-fitting for the in-sample points in the test data, as compared to the out-of-sample points. This effect is usually negligible, but it sometimes requires some thought during the design of the prediction process. The scores of each point from different subsample-based models are averaged as in classification [105].

An application of subsampling in outlier ensembles was proposed for the case of graph data<sup>9</sup>, and the first subsampling method in multidimensional data was provided in the work on isolation forests [367]. However, the primary focus of the isolation forest was on improving computational efficiency of the ensemble by performing the

<sup>9</sup>The work in [17] subsamples edges of a graph (i.e., *entries* of the adjacency matrix) to create multiple clustering models and scoring the outlier tendency of edges (see section 12.3.2.3 of Chapter 12).

base model construction on a small subsample of size  $s$ . It was shown that one could *sometimes* obtain accuracy improvements with subsampling, although such improvements were dependent on the quirks of the underlying data distribution. In many cases, subsampling worsened the accuracy of the isolation forest. An explanation of this behavior is provided in section 6.4.5.

Although the accuracy improvement provided by subsampling is unpredictable in the isolation forest setting, better accuracy improvements have been observed for distance-based detectors [32, 621]. Even in these cases, the results tend to be sensitive to the size of the subsample in a somewhat unpredictable way mainly because the relative performance of the *base detector* varies with subsample size [32]. For example, if an outlier occurs together with a small cluster of other outliers in a particular data distribution, the base  $k$ -nearest neighbor detector might avoid the interfering effect of this cluster while scoring outlier points with a smaller subsample. In complex data distributions, larger subsample sizes may be beneficial to the bias performance of the base detector. On the other hand, variance reduction is usually better for smaller subsamples because of less overlaps and correlations among base detectors. The combination of the two effects can often result in an optimal sample size to use for a particular base detector and base distribution, although this optimal sample size cannot be estimated in an unsupervised problem like outlier detection. In practice, the optimal sample size is small for simpler base detectors and data distributions.

The computational benefit is significant for distance-based detectors, because their computational complexity increases superlinearly (quadratically) with the number of points. In the case of a quadratic detector, the complexity of subsampled detectors often depends on the testing phase, which requires  $O(N \cdot s)$  time instead of  $O(N^2)$  time. In general, subsampling has significant potential to improve the accuracy and efficiency of outlier detectors when implemented properly; a specific example is *variable subsampling* [32].

### 6.4.6.3 Variable Subsampling

One of the challenging aspects of subsampling is that the performance of a particular base detector is often crucially dependent on the size of the subsample in an unpredictable way. For example, if an original data set contains 1000 data points, and a value of  $k = 50$  is used (at 5% of the data size) for a  $k$ -nearest neighbor detector, then this same parameter value will be at 50% of the data set size for a subsample of size 100. Clearly, the effects of data size on accuracy will be significant and unpredictable and in some cases fixing parameters can lead<sup>10</sup> to poorer performance with more data. In general, the *optimal* size of the required subsample at a particular parameter setting is unknown; furthermore, it is also impossible to estimate this optimal size in unsupervised problems. This is not as much of a problem when using subsampling in supervised problems like classification because it is possible to use cross-validation to determine the optimal subsample size for a particular parameter setting (or the optimal parameter setting for a particular subsample size).

Even in cases where the algorithm is parameter-free, the optimal size of the required subsample for a particular data set is typically unknown because it depends on the specific interaction between the detector and the data set at hand. Therefore, a natural solution to this dilemma is to use variable subsampling. Note that variable subsampling is more

---

<sup>10</sup>This observation has repeatedly been misinterpreted by various researchers to suggest (more generally) that one can improve the accuracy of *base* outlier detectors by simply using less data for model-building. As shown in [32], this is an incorrect claim. In general, the main problem is that it is often difficult to make asymptotically optimal parameter or algorithm design choices in unsupervised problems. For example, one does not observe better performance with less data in an exact  $k$ -nearest neighbor detector if one adjusts the value of  $k$  proportionally with data size.



powerful than simply varying the parameter value for various subsamples, because it also works for parameter-free detectors in which the design choices of the detector might regulate the optimal data size to be used. Parameter-free detectors are often particularly sensitive to subsample size in an unpredictable way.

Let  $N$  be the number of points in the base data set  $\mathcal{D}$ . The algorithm proceeds as follows:

1. Select  $f$  uniformly at random between  $\min\{1, \frac{50}{N}\}$  and  $\min\{1, \frac{1000}{N}\}$ , where  $N$  is the number of points in the original data set  $\mathcal{D}$ .
2. Select  $f \cdot N$  randomly sampled points from the original data  $\mathcal{D}$ , and apply the base outlier detector on this sample to create an outlier detection model. Score each point in  $\mathcal{D}$  using this model.

At the end of the process, the scores of each data point in different components are averaged to create a unified score. However, before averaging, the  $N$  outlier scores from each detector should be standardized to zero mean and unit variance. This standardization is necessary because subsamples of different sizes will create outlier scores of different raw values for unnormalized  $k$ -nearest neighbor algorithms. It is noteworthy that the subsampling approach always selects between 50 and 1000 data points irrespective of base data size. For data sets with less than 1000 points, the maximum raw size would be equal to the size of the data set. For data sets with less than 50 points, subsampling is not recommended.

We now analyze the effect of such an approach on parameter choice, by using the  $k$ -nearest neighbor algorithm as an example. The merit of this approach is that it effectively samples for different values of model parameters. For example, varying the subsample size at fixed  $k$  effectively varies the *percentile value of  $k$*  in the subsample. In general, holding data size-sensitive parameters fixed, while varying subsample size, has an automatic effect of parameter space exploration. If we view each component detector *after* selecting the subsample size, then it has a bias, which is component dependent. However, if we view the randomized process of selecting the subsample size as a part of the component detector, then every component has the same bias, and the variability in the aforementioned component-dependent bias now becomes a part of this detector variance. One can reduce this variance with ensembling, with the additional advantage that the underlying component detectors of variable subsampling tend to be far less correlated with one another as compared to fixed subsampling. As a result, one can now aim for better accuracy improvements in the ensemble. Note that variable subsampling *combines data-centric variance-reduction with model-centric variance reduction*. Therefore, this approach provides variance reduction not only over different choices of the training data, but also over different randomized choices of  $k$  (in an implicit way). In other words, the approach becomes insensitive to specific parameterizations. However, the approach has implications beyond parametric ensembling because it also provides improvements for parameter-free base detectors, in which the optimal subsample size is not known for a particular algorithm. For example, it has been shown in [367] that certain data-specific training sample sizes enhance outlier discovery for parameter-free methods like isolation trees. Unfortunately, this optimal sample size cannot be estimated for unsupervised problems; therefore, variable subsampling reduces the underlying uncertainty. This makes the *VS* approach more general and desirable than simply varying the values of the parameters across detectors; it is independent of the nature of the parameters/design choices in the base detector and it *concurrently* achieves other forms of variance reduction in an implicit way. For data size-sensitive parameters, it is advisable to select them while keeping in mind that subsample sizes vary between 50 and 1000 points. Knowledge of subsample sizes eases the parameter selection process to some extent. For example, for distance-based

detectors, it is recommended [32] that a value of  $k = 5$  will result in a percentile value of  $k$  varying between 0.5% to 10% of data size, which seems reasonable. For very simple and stable detectors like (raw) distance-based detectors, it makes sense to use small values of  $k$  in combination with small subsample sizes to reduce overlaps between subsamples (and correlations among base detectors). Therefore, one can also use  $k = 1$  in combination with proportionally reduced samples sizes between 10 and 200 points [35]. However, this would not be an advisable approach for a more complex detector like the kernel Mahalanobis method, which tries to model more detailed characteristics of the data distribution.

It is noteworthy that variable subsampling works with raw subsample sizes between 50 and 1000, irrespective of base data size. By fixing the subsample size in a constant range, it would seem at first sight that the approach cannot take advantage of the larger base data sizes. This is, however, not the case; larger data sets would result in less overlap across different subsamples, and therefore less correlation across detectors. This would lead to better variance reduction. The idea is to leverage the larger base data size for better de-correlation across detectors rather than build more robust base detectors with larger subsamples; the former is a more efficient form of variance reduction. After all, the number of points required to accurately model a distribution depends on the absolute subsample size, rather than on the size of the original data set obtained by the data collector.

#### 6.4.6.4 Variable Subsampling with Rotated Bagging (VR)

It is possible to combine the base detectors in variable subsampling and rotated bagging to create an even more diverse base detector. This will help in variance reduction. Furthermore, because of the reduction in terms of *both* points and dimensions, significant computational savings are achieved. The combined base detector is created as follows:

1. Project the data into a random  $2 + \lceil \sqrt{d}/2 \rceil$ -dimensional space using the rotation method of section 6.4.4.
2. Select a variable size subsample using the approach described in section 6.4.6.3.
3. Score each point using the reduced data set.

The scores of these individual detectors can then be averaged into the final ensemble score. It is important to use Z-score normalization of the scores from the base detectors before combination. This approach is referred to as variable subsampling with rotated bagging (VR).

Rotated bagging has clear computational benefits because one is using only  $\sqrt{d}$  dimensions. With increasing dimensionality, the benefit increases. When combined with variable subsampling, the benefits can be very significant. For example, for a data set containing ten million points and 100 dimensions (i.e., a billion entries), each ensemble component would use a data matrix of size at most  $1000 \times 7$  (i.e., less than ten-thousand entries). In space-constrained settings, this can make a difference in terms of being able to use the approach at all. For 100 trials, the ensemble (containing quadratic base detectors) would be hundreds of times faster than a single application of the base method on the full data.

#### 6.4.7 Other Variance Reduction Methods

The work in [35] discusses a number of other methods for outlier ensembles. Most of these methods have been adapted from their counterparts in the classification domain. Some examples are as follows:

1. **Randomized feature weighting:** The various features of the data are scaled randomly to induce diversity. First, all features are scaled to zero mean and unit variance. Subsequently, the scaling weight  $w_i$  of the  $i$ th dimension is defined using the following density function  $f_X(x)$ :

$$f_X(x) = \begin{cases} \frac{\alpha}{x^{\alpha+1}} & \text{if } x \geq 1 \\ 0 & \text{otherwise} \end{cases} \quad (6.14)$$

This density function is the Pareto distribution. The value of  $\alpha$  is chosen uniformly at random from  $(1, 2)$ . Each feature value is multiplied with its scaling weight  $w_i$  before applying the base detector. The scores from various ensemble components are combined with the use of an averaging or maximization combination function. The approach can be viewed as a soft version of feature bagging.

2. **Wagging:** In wagging, the various points are weighted by values drawn from a probability distribution. Either the uniform or the Gaussian distribution may be used. However, this approach requires the modification of base detectors to work with weighted points. For example, an average  $k$ -nearest neighbor algorithm weights the points while calculating the average distance to the  $k$  nearest neighbors. The same is true for the LOF algorithm in which average *reachability* distances need to be computed. Furthermore, LOF divides the reachability distance of a point with the harmonically averaged reachability distance in its locality to compute the final score (cf. Equation 4.4). The harmonic averaging<sup>11</sup> is also done in a weighted way. Therefore, the parametrizations of the algorithms do not change, although the computed values are affected by the weights. The test points are scored with the weighted base detectors. The scores across these different instantiations are then averaged.
3. **Geometric subsampling:** Geometric subsampling [35] is a variant of variable subsampling in which the number of points sampled is selected in a way that provides better diversity for certain types of detectors like distance-based detectors. Instead of always sampling fraction of the data between  $\min\{1, \frac{50}{N}\}$  and  $\min\{1, \frac{1000}{N}\}$ , it samples a value  $g$ , which is drawn uniformly at random from  $\log_2(\min\{1, \frac{50}{N}\})$  and  $\log_2(\min\{1, \frac{1000}{N}\})$ . Subsequently, the approach samples a fraction  $f = 2^g$  of the data.

In the following, we briefly describe the main advantage of geometric subsampling over variable subsampling for distance-based detectors. It is noteworthy that more than half the ensemble components in variable subsampling use between 500 and 1000 data points, in which the value of  $k/N$  varies only within a factor of 2. This property reduces the diversity of parameter space exploration and the results are often dominated by the samples larger than 500 points. Geometric subsampling increases the diversity of parameter-space exploration by ensuring that there is greater diversity in the value of  $k/N$  over different subsamples. As shown in [35], the geometric subsampling approach often provides higher-quality results than variable subsampling.

The first two of these ensemble combination methods have been adapted directly from classification. An experimental comparison of many of these methods is provided in [35]. In general, almost all variance-reduction methods from classification can be adapted to outlier detection because of the similarity of the bias-variance trade-off in the two cases.

<sup>11</sup>Consider a set of relative weights  $w_1 \dots w_k$  within a  $k$ -nearest neighbor locality which are scaled to sum to 1. Then, the weighted harmonic mean of  $x_1 \dots x_k$  is given by  $1/(\sum_{i=1}^k [w_i/x_i])$ .

## 6.5 Flying Blind with Bias Reduction

---

Bias reduction is a very difficult problem in the context of outlier detection. Note that bias can be viewed as the *inherent* error of a model because of the poor assumptions made in the model. The first term in Equation 6.12 corresponds to the (squared) bias. This term uses the oracle  $f(\overline{X}_i)$ , which is unknown in unsupervised problems like outlier detection. In supervised problems like classification, (possibly noisy) examples of the output of this oracle are available in the form of labels. Because of the presence of  $f(\overline{X}_i)$  in the bias term, all bias-reduction methods in classification use labels in one form or the other.

A classical example of a bias-reduction method in classification is *boosting*. In this method, the accuracy of a detector on a training instance is used to sequentially readjust its weight in later iterations. Specifically, the weights of incorrectly classified examples are increased. Note that the determination of incorrectly classified examples requires knowledge of the underlying labels. This process is continued until the entire training data is classified with complete accuracy. The final classification of a test instance is performed as a weighted combination of the results from the individual learners. A detailed discussion of boosting is provided in Chapter 7.

Similar methods are, however, hard to construct in outlier detection. This is because the use of the ground-truth is almost always crucial in these cases. Nevertheless, a limited amount of bias reduction can still be *heuristically* achieved in these cases. We emphasize the fact that these methods have much greater uncertainty attached to their performance as compared to variance-reduction methods. This is, in part, because these techniques substitute the outputs of algorithms for the ground-truth, and there is an inherent circularity in making design choices about the algorithm on this basis. The nature of this circularity will become evident from the examples in the subsequent sections.

### 6.5.1 Bias Reduction by Data-Centric Pruning

A technique that is frequently used to improve outlier detection algorithms but is often not recognized as a meta-algorithm is the technique of iterative outlier removal. The basic idea in this case is that all outlier detection methods assume that the model is constructed on normal points in order to estimate scores. Therefore, the removal of outliers from the training data can sometimes be helpful in improving the correctness of this *assumption* (i.e., improving *bias*). Recall that errors in modeling assumptions cause bias. However, since the knowledge of the ground-truth is not available, how do we know which point to remove? It is here that the natural circularity of bias-reduction algorithms becomes evident. The basic idea here is that we can use the output from a base detector in a conservative way to remove outliers in subsequent iterations. By “conservative” we refer to the fact that we set a high bar for removal of a point. For example, one can convert the outlier score to a Z-value and set a large threshold requirement on the Z-value for removal of the outlier. Therefore, one can use the outlier detection algorithm  $\mathcal{A}$  iteratively as shown in Figure 6.6.

The algorithm proceeds by successively refining the outliers from the underlying data  $\mathcal{D}$  to keep a current version of a sanitized data set, referred to as  $\mathcal{D}_{current}$ . In each iteration, an outlier model  $\mathcal{M}$  is constructed on  $\mathcal{D}_{current}$ . For example, if a one-class SVM model is to be constructed, then the data  $\mathcal{D}_{current}$  is used. The data set  $\mathcal{D}$  is used as the test data and each point is scored against the model. If a  $k$ -nearest neighbor detector is used, then each point in  $\mathcal{D}$  is scored using the  $k$ -nearest neighbors in  $\mathcal{D}_{current}$ . Subsequently, the outliers are removed from  $\mathcal{D}$  to create the new data set  $\mathcal{D}_{current}$ . This process is repeated for several iterations as the outlier set is successively refined. The outliers discovered in the

**Algorithm** *IterativeOutlierRemoval*(Data Set:  $\mathcal{D}$ )  
**begin**  
 $\mathcal{D}_{current} = \mathcal{D}$ ;  
**repeat**  
  Apply algorithm  $\mathcal{A}$  to data set  $\mathcal{D}_{current}$  to build model  $\mathcal{M}$ ;  
  Score each point in  $\mathcal{D}$  using model  $\mathcal{M}$  to determine outlier set  $\mathcal{O}$ ;  
   $\mathcal{D}_{current} = \mathcal{D} - \mathcal{O}$ ;  
**until** convergence or maximum number of iterations;  
**return**( $\mathcal{O}$ );  
**end**

Figure 6.6: Iterative Outlier Removal

final iteration of the algorithm are reported.

## 6.5.2 Bias Reduction by Model-Centric Pruning

In the previous section, it was discussed how bias can be reduced with the use of data-centric pruning. It turns out that bias can also be reduced with the use of model-centric pruning methods. An example of such a method is the *SELECT* technique proposed in [461]. The basic idea is to remove the inaccurate models in the ensemble to improve the overall accuracy. From the perspective of Equation 6.12, one is now trying to remove models for which the first term  $\sum_{i=1}^n (f(\bar{X}_i) - E[g(\bar{X}_i, \mathcal{D})])^2$  is large. Unfortunately, we do not have access to  $f(\bar{X}_i)$  in real settings. Therefore, the key idea in [461] is to use a robust ensemble output in lieu of the true ground-truth value  $f(\bar{X}_i)$  in order to determine the detectors that should be removed. This substituted value is also referred to as the *pseudo-ground truth*. It is noteworthy that the effectiveness of the scheme is crucially dependent on the correctness of the pseudo-ground truth; therefore, at least a reasonable number of base detectors in the original set of models should be accurate.

A very primitive<sup>12</sup> and bare-bones version of the *SELECT* scheme is as follows:

1. Normalize the scores from various detectors to the same scale. Numerous methods might be used for standardization, although the *SELECT* approach uses a technique discussed in [213]. Let the normalized score of the  $i$ th point for the  $j$ th detector be  $O(i, j)$ .
2. Compute the average score  $a_i = \sum_{j=1}^m O(i, j)/m$  of the  $i$ th point over all ensemble components.
3. Retain only detectors that correlate well with the pseudo-ground-truth. This step is referred to as the *selection* step.
4. Report the combined score of the remaining detectors as the final score. Examples of such combination functions include the use of averaging or maximum. In principle, any of the combination methods discussed in section 6.6 may be used.

The third step of selecting the detectors requires further explanation. How to select detectors that correlate well with the pseudo-ground-truth? The first step is to compute a *global*

<sup>12</sup>This description is not precisely the same as discussed in [461], although it captures the basic idea behind the approach.

pseudo-ground truth  $G$  based on all the detectors. This artificial ground truth  $G$  is a vector of scores over all points, such that the score of each point is the average normalized score over all detectors.

First, the detectors are sorted based on their Pearson correlation coefficient of their score vectors with that of the pseudo-ground truth  $G$ , and the detector most correlated with the ground-truth is added to an empty set  $\mathcal{L}$  to create a singleton ensemble. Subsequently, detectors are added to  $\mathcal{L}$  iteratively based on correlation with the *current* ensemble represented by  $\mathcal{L}$ . The average (normalized) score of each point from the detectors in  $\mathcal{L}$  is computed. The remaining detectors (i.e., detectors not in  $\mathcal{L}$ ) are sorted based on their Pearson correlation with the current (normalized) average of the scores in  $\mathcal{L}$ . The process is repeated by adding the first detector in this ranked list to  $\mathcal{L}$ . In each iteration, it is also tested whether adding a detector improves the correlation of the ensemble constructed from  $\mathcal{L}$  with the *global* pseudo-ground truth  $G$  over all detectors. If the correlation increases, then the current detector is added. Otherwise, the process terminates. This approach is referred to as *vertical selection*.

The vertical selection method focuses on *all* the points when computing correlations. The horizontal selection method gives greater importance to the anomalous points while computing the relationships between the various detectors. Therefore, in this case, the binary labeling of points is used rather than the scores. In general, any statistical thresholding on the scores can yield binary labels. A mixture-modeling approach (cf. section 2.4.4 of Chapter 2) is used in order to convert the score list from every detector into a list of binary labels. For each outlier point, its ranks are computed across the different detectors. Detectors in which many (pseudo) ground-truth outlier points are ranked very highly are selected. A method based on order-statistics is used to provide a crisp criterion for this selection. Details may be found in [461]. The basic idea in horizontal selection is that the top-ranked points are more important than the entire list because the anomalies are selected only from the top-ranked points.

### 6.5.3 Combining Bias and Variance Reduction

A variety of methods can be used to combine bias and variance reduction. Many of these techniques use biased point or dimension sampling methods. Some of these methods are discussed in the context of high-dimensional outlier detection.

1. One can use statistical selection of relevant subspaces rather than using completely randomized methods. Two such methods are the *HiCS* method [308] and the *OUTRES* method [402]. These methods are discussed in sections 5.2.7 and 5.2.8, respectively, of Chapter 5. Both these techniques perform different types of statistical selection of subspaces. For example, the *HiCS* method preselects subspaces based on a deviation test of their non-uniformity. The base algorithm is applied only to the selected subspaces. By using this approach, one is biasing the detector towards sharpened outlier scores. At the same time, the scores from various subspaces are averaged. Therefore, variance is reduced as well. In principle, one can use any other type of feature selection measure such as Kurtosis (cf. section 1.3.1 of Chapter 1) to bias the dimension selection.
2. One can bias the subsampling approach with a computed probability that a data point is an outlier. In other words, an iterative subsampling approach is used in which data points that are scored as outliers have a lower probability of being selected in the next subsample. The scores are then averaged over various iterations to provide the

final result. This approach can be considered a randomized variant of the method in section 6.5.1 and it is discussed in detail in [35].

Bias reduction can also be achieved in the final step of model combination. A variety of such methods are discussed in the next section.

## 6.6 Model Combination for Outlier Ensembles

---

Given the outlier scores from various detectors, a final step of ensemble-based approach is to combine the scores from various detectors. It was already discussed in section 6.2, how the scores of different detectors may be combined. As in that discussion, consider a setting in which the (normalized) score for the  $i$ th point by the  $j$ th detector is denoted by  $S_j(i)$ . It is assumed that there are  $N$  points and  $m$  detectors. The two combination functions already discussed earlier in this chapter are as follows:

1. **Averaging:** For the  $i$ th data point, the average of  $S_1(i) \dots S_m(i)$  is reported as its final outlier score.
2. **Median:** For the  $i$ th data point, the median of  $S_1(i) \dots S_m(i)$  is reported as its final outlier score [17, 35].
3. **Maximum:** For the  $i$ th data point, the maximum of  $S_1(i) \dots S_m(i)$  is reported as its final outlier score.

These different types of combination functions have different effects in terms of the bias and variance. The effect of both averaging and the median is very similar by promoting stability, although the averaging function is used more commonly. Several studies [32, 35, 344] have shown that the different combination functions may perform differently in various settings. For some data sets, the averaging function seems to perform better, whereas for other data sets, the maximization function seems to perform better. However, the averaging function seems to be relatively stable because its performance does not seem to vary too much over different instantiations from the same data distribution. The main problem with the maximization function is its lack of stability, and it sometimes makes more sense to use it in combination with rank-centric scores [344] in order to reduce the effects of run-away outlier scores.

In order to understand the nature of these combination functions, one needs to revisit the bias-variance trade-off. It is well-known from the theoretical results in the classification domain [98] that averaging reduces variance. Since the bias-variance trade-off in outlier detection is almost identical that in classification [32], it follows that averaging and the median will also reduce variance in outlier detection.

The effect of the maximization function is far more subtle as compared to the averaging function and is based on a heuristic observation of how outlier detectors often behave on real data sets. In real settings, one is often able to de-emphasize irrelevant or weak ensemble (poorly biased) components with the maximization function. Therefore, one is often able to reduce bias. Even though it might seem at first sight that using a maximization function might overestimate scores, it is important to keep in mind that outlier scores are relative, and one always applies the bias-variance trade-off on a normalized representation of the scores in order to meaningfully use measures such as the MSE with respect to an absolute interpretation of the scores. Therefore, one can no longer talk about overestimation or underestimation of the scores, because *relatively* overestimated scores always need

to be counterbalanced by relatively underestimated scores. The main problem with the maximization function is that it *might* increase variance, especially for small training data sets. The specific effect will depend on the data set at hand; indeed it has been shown in multiple papers [32, 344] that the different combination functions work better for different data sets. As a result of this instability (because of possible variance increase), the accuracy of the maximization function may sometimes fall below the base detector quality. However, in other cases, it can also provide large improvements. At the same time, the instability of the maximization scheme is often a matter of concern and discourages practitioners from using it. Not using the maximization function simply because of its instability leaves large potential improvements on the table. A reasonable solution is to combine the maximization function with variance amelioration in order to provide more accurate results. We will revisit such methods in section 6.6.2.

Next, we explain the bias reduction effects of the maximization combination. In many “difficult” data sets, the outliers may be well hidden, as a result of which many ensemble components may give them inlier-like scores. On the other hand, the variance of inlier points is often far more modest because there is little to distinguish them from their nearby points. In such cases, the scores of outlier points are often *relatively* underestimated in most ensemble components as compared to inlier data points. In order to explain this point, let us consider the feature bagging approach of [344], in which the outliers are hidden in small subsets of dimensions. In such cases, depending on the nature of the underlying data set, a large majority of subspace samples may not contain many of the relevant dimensions. Therefore, most of the subspace samples will provide significant underestimates of the outlier scores for the (small number of) true outlier points and mild overestimates of the outlier scores for the (many) normal points. This is a problem of *bias*, which is caused by the well-hidden nature of outliers. As discussed in [32], such types of bias are inherent to the problem of outlier detection. The scores of outlier points are often far more *brittle* to small algorithm modifications, as compared to the scores of inlier points. Using a maximization ensemble is simply a way of trying to identify components in which the outlier-like behavior is best magnified. At the same time, it needs to be kept in mind that bias-reduction in an unsupervised problem like outlier detection is inherently heuristic, and it might not work for a specific data set. For example, if a training data set (or subsample) is very small, then the maximization function will not work very well because of its propensity of pick out the high variance in the scores.

As discussed in [32], the maximization function often has the ability to pick out non-obvious outliers, which appear in only a few of the ensemble components. In the easy cases in which most outliers are “obvious” and can be discovered by the majority of the ensemble components, the averaging approach will almost always do better by reducing variance effects. However, if it can be argued that the discovery of “obvious” outliers is not quite as interesting from an analytical perspective, the maximization function will have a clear advantage.

### 6.6.1 Combining Scoring Methods with Ranks

A related question is whether using ranks as base detector output might be a better choice than using absolute outlier scores. After all, the metrics for outlier detection are based on the rank-wise AUCs rather than the score-wise MSEs. Ranks are especially robust to the instability of *raw* scores of the underlying detectors. A specific example of this is the fact that the LOF algorithm often returns  $\infty$  scores because of harmonic normalization (cf. section 4.4.1.1 of Chapter 4). In such cases, the use of ranks has beneficial effects because



they reduce the effects of run-away outlier scores. On the other hand, ranks do lose a lot of relevant information discriminating between various points. In such cases, using ranks could increase bias-centric errors, which might also be manifested in the ranks of the final combination score.

## 6.6.2 Combining Bias and Variance Reduction

Clearly, the bias-variance trade-off suggests that different combination functions might do better in different settings. The averaging function does better in terms of variance whereas the maximization function often does better in terms of bias. Therefore, it is natural to balance the effort in reducing bias and variance by combining the merits of the two methods. Two such schemes were proposed in [32]. In each cases, it is important to normalize to Z-scores before applying the combination function:

1. **AOM Method:** For  $m$  ensemble components, the components are divided into approximately  $m/q$  buckets of  $q$  components each. First, a maximization is used over each of the buckets of  $q$  components, and then the scores are averaged over the  $m/q$  buckets. Note that one does not need to assign equal resources to maximization and averaging; in fact, the value of  $q$  should be selected to be less than  $m/q$ . For example, the implementation in [32] uses 100 trials with  $q = 5$ . This method is referred to as *Average-of-Maximum*, which is also abbreviated to *AOM*.
2. **Thresh Method:** A method suggested in [31], for combining the scores of multiple detectors, is to use an absolute threshold  $t$  on the (standardized) outlier score, and then adding the (thresholded and standardized) outlier scores for these components. The threshold is chosen in a mild way, such as a value of  $t = 0$  on the standardized score. Note that values less than 0 almost always correspond to strong inliers. The overall effect of this approach is to reward points for showing up as outliers in a given component, but not to penalize them too much for showing up as strong inliers. The implementation in [32] used a threshold value of  $t = 0$  on the Z-score. An important point is that such an approach can sometimes lead to tied scores among the *lowest ranked* (i.e., least outlier-like) points having a score of exactly  $m*t$ . Such ties are broken among the lowest ranked points by using their average standardized score across the  $m$  ensemble components. As a practical matter, one can add a small amount  $\epsilon * avg_i$  proportional to the average standardized score  $avg_i$  of such points in order to achieve the desired tie-breaking. This approach is referred to as *Thresh*.

The *AOM* combination scheme is particularly useful when the maximum number of trials is not a concern from the computationally efficiency perspective. With simple schemes like averaging the benefits are saturated very quickly. However, to saturate the benefits of combining maximization and averaging (e.g., *AOM*) one would need a larger number of trials. However, it shown in [32] that even with the same number of trials, schemes like *AOM* often do better than averaging. With faster base detectors, one can run a far larger number of trials to gain the maximum accuracy improvements from *both* bias and variance reduction. This implies that there are significant computational advantages in designing efficient base detectors. The *Thresh* method can be viewed as a faster way of combining bias and variance reduction, when computational efficiency is important. Other ideas for combining bias and variance reduction include the use of *Maximum-of-Average (MOA)*.

## 6.7 Conclusions and Summary

---

Outlier ensembles have seen an increasing interest from the research community in recent years, which is motivated, in part, by increasingly challenging data sets that are resistant to accurate outlier discovery. An example of such a challenging instantiation is the high-dimensional case in which no single subspace can capture all the outliers. In spite of the superficial differences between the two problems, the theoretical foundations of outlier ensembles are similar to that in the supervised case. Variance-reduction methods are among the most popular techniques used in the context of outlier ensembles. Common variance-reduction techniques include feature bagging, subsampling, and isolation forests. Recently, a number of techniques have also been proposed for bias reduction. Bias-reduction methods are, however, more difficult to implement for outlier detection because of the unavailability of ground truth. The proper combination of the output of outlier detectors is very important for obtaining more accurate results. The most common combination functions include the averaging and the maximization function, which primarily have effects on variance and bias, respectively. These functions can also be combined to benefit from simultaneous bias and variance reduction.

## 6.8 Bibliographic Survey

---

Ensemble analysis has a rich history in the field of classification [98, 266, 617] and clustering [516]. The field is, however, far more nascent in the field of outlier analysis. Much of the earliest work in the field of outlier ensembles was motivated by the work on subspace outlier detection [4]. In particular, it has been shown [258] that the first scheme [4] on subspace outlier detection was an ensemble technique in which the maximization combination function was used. However, the approach was not formally claimed as an ensemble method. Soon after, a scheme by [344] proposed the problem of subspace outlier detection more formally in the context of feature bagging. This approach proposed the use of both the averaging and the maximization function, and is therefore credited with the first use of the averaging function in outlier ensembles, although averaging was used earlier in the classification domain for variance reduction. The work in [621] claims to provide a theoretical proof of the averaging function and subsampling effectiveness, which is incorrect; a correct proof based on the connections of the approach to classification is found in [32]. A number of other methods in subspace outlier detection were also proposed for statistical selection of relevant subspaces and their combinations [32, 308, 367, 368, 402, 473]. The isolation forest approach has been modified to discover clustered anomalies [369] and to the case of streaming data [532]. The work in [412] applied ensembles of heterogeneous detectors on high-dimensional data, whereas methods for combining specific types of one-class methods are discussed in [540]. Among the earliest works, there is also an interesting work [213] that shows how to convert scores to probabilities. This is useful for normalizing the outputs of detectors. The rotated bagging work [32] is also notable because of its ability to reduce the dimensionality of the data, and obtain strong outlier detection results by combining the results from many weak detectors. An independently proposed implementation of rotated bagging [436], referred to as *LODA*, uses 1-dimensional rotated projections in combination with histogram-based methods in order to obtain effective results in high-dimensional data. The *RS-Hash* method [476] samples axis-parallel subspace grid regions of varying size and dimensionality to score data points as outliers (cf. section 5.2.5 of Chapter 5).

A position paper describing the key challenges in outlier ensembles was first published

in [31]. This work stated the many challenges associated with outlier ensembles, and the way in which this area is different from classification ensembles. A taxonomy of the field of outlier ensembles is provided in [31] and various model combination methods are also discussed. In this paper, it was also proposed to use bagging techniques [98] for outlier detection, although experimental results were presented much later in a recent book [35].

Bagging is closely related to subsampling, which is used commonly in classification [105, 106, 107]. Entry-wise subsampling of a graph adjacency matrix is proposed in [17]. For multidimensional data, subsampling was proposed in the context of isolation forests [367]; however, there are some detrimental bias-centric effects of this subsampling. The use of subsampling with  $k$ -nearest neighbor detectors was proposed in [621]. However, this work provides an incorrect theoretical explanation. The correct theoretical explanation for outlier ensembles is provided in [32]. Many new variance-reduction methods for outlier ensembles are discussed in [35]. One merit of the isolation forest (and other random forest) methods is that they are generally very efficient. As a result, variants of this class of methods have been recently adapted to streaming anomaly detection [532, 571]. Many new variance-reduction methods, such as wagging, randomized feature weighting, and geometric subsampling, have been proposed [35]. These results show that the geometric subsampling approach is an extremely efficient method that works well in practice.

Recently, a number of bias-reduction methods have also been proposed for anomaly detection. As discussed in [32], the use of the maximization combination function is able to provide bias reduction. The *SELECT* scheme was proposed in [461], which proposes bias reduction with a model-centric setting. Another approach for reducing bias in the temporal setting by selecting more relevant models is proposed in [471]. As suggested in [32], one can also perform data-centric bias reduction by removing obvious outliers from the data or by using biased subsampling.

## 6.9 Exercises

---

1. Consider the data-centric bias reduction approach in which outliers are iteratively removed. Provide an example of an algorithm and a data set in which the use of such an approach might be counter-productive.
2. Implement the rotated bagging approach to variance reduction.
3. For a given data set, would you expect a  $k$ -nearest neighbor detector with a larger value of  $k$  to have greater variance, or would you expect a smaller value of  $k$  to have greater variance? Why? Which one would typically obtain larger improvement with subsampling at a fixed ratio? Which one would typically obtain a better accuracy?

---

---

## Chapter 7

# Supervised Outlier Detection

---

---

“True, a little learning is a dangerous thing, but it still beats total ignorance.” – Abigail van Buren

### 7.1 Introduction

---

The discussions in the previous chapters focus on the problem of unsupervised outlier detection in which no prior information is available about the abnormalities in the data. In such scenarios, many of the anomalies found correspond to noise or other uninteresting phenomena. It has been observed [338, 374, 531] in diverse applications such as system anomaly detection, financial fraud, and Web robot detection that *interesting anomalies are often highly specific to particular types of abnormal activity in the underlying application*. In such cases, an unsupervised outlier detection method might discover noise, which is not specific to that activity, and therefore may not be of interest to an analyst. In many cases, different types of abnormal instances could be present, and it may be desirable to distinguish among them. For example, in an intrusion-detection scenario, different types of intrusion anomalies are possible, and the specific type of an intrusion is important information.

The goal of supervised outlier detection is to empower learning methods with application-specific knowledge so as to obtain application-relevant anomalies. This knowledge often includes examples of such relevant anomalies, although other types of supervision are also possible. Because of the rare nature of anomalies, such examples are often limited. This causes challenges in creating robust models. Nevertheless, even when a small amount of data is available for supervision, its incorporation usually improves outlier-detection accuracy in a significant way. *The general recommendation for outlier analysis is to always use supervision where possible.*

The supervision is provided by examples of normal or abnormal data. This is referred to as *training data*, and it can be used to create a *classification model* that distinguishes between normal and anomalous instances. The problem of classification has been widely studied in its own right, and numerous algorithms are available in the literature [33, 176] for creating supervised models from training data.

So how is the supervised outlier detection problem different from classification? The supervised outlier detection problem may be considered a very difficult special case (or variation) of the classification problem. This is because the problem is associated with several challenging characteristics, which may be present either in isolation or in combination:

- **Class imbalance:** Since outliers are defined as rare instances in the data, it is natural that the distribution between the normal and rare class will be very skewed. From a practical perspective, this implies that the optimization of classification accuracy may not be meaningful, especially since the misclassification of positive (outlier) instances is less desirable than the misclassification of negative (normal) instances. In other words, false positives are more acceptable than false negatives. This leads to cost-sensitive variations of the classification problem, in which the natural optimization function for classification (which is accuracy) is changed to *cost-sensitive* accuracy.
- **Contaminated normal class examples (Positive-unlabeled class problem):** In many real scenarios, only the positive class is labeled, and the remaining “normal” data contains some abnormalities. This is natural in large-scale settings like the Web and social networks, in which the sheer volume of the underlying data makes contamination of the normal class more likely. For example, consider a social networking application, in which it is desirable to determine spam in the social network feed. A small percentage of the documents may be spam. In such cases, it may be possible to recognize and label some of the documents as spam, but many spam documents may remain in the examples of the normal class. Therefore, the “normal” class may also be considered an unlabeled class. In practice, however, the unlabeled class is predominantly the normal class, and the anomalies in it may be treated as contaminants. Technically, this case can be treated as a difficult special case of full supervision, in which the normal class is noisy and contaminated. Off-the-shelf classifiers can be used on the positive-unlabeled version of the classification problem, as long as the relative frequency of contaminants is not extreme.
- **Partial training information (semi-supervision or novel class detection):** In many applications, examples of one or more of the anomalous classes may not be available. For example, in an intrusion detection application, one may have examples of the normal class and *some* of the intrusion classes, as new types of intrusions emerge with time. In some cases, examples of one or more normal classes are available. A particularly commonly studied case is the *one-class variation* in which examples of only the normal class are available. This particular special case, in which the training data contains only normal classes, is much closer to the unsupervised version of the outlier detection problem. No changes are required to existing algorithms, other than properly distinguishing between the training and test data.

It is possible for these scenarios to be present in combination, and the boundaries between them are often blurred. The goal of this chapter is to elucidate the required *modifications* to classification methods to address these different settings. Therefore, a working knowledge of classification methods is assumed [176].

All classification problems, including rare-class problems, are heavily dependent on the feature representation used for the learning process. For example, kernel methods are often used in order to make nonlinearly separable classes linearly separable via an implicit transformation. However, in *feature engineering*, this transformation is performed explicitly with an understanding of the domain characteristics of the problem at hand. Rare-class learning

is one such domain in which the output scores of outlier detection algorithms can be used as engineered features for more effective learning. Therefore, the use of unsupervised outlier detection algorithms will be explored for feature engineering in supervised problems.

Paucity of training data is a common problem, when the class distribution is imbalanced. Even in a modestly large training data set, only a small number of rare instances may be available, which can lead to poor results. In order to address this problem, *active learning* is used to label training examples in a guided way. This is achieved by providing an expert with pre-filtered candidate points for labeling. Such labeling can sometimes be accomplished with infrastructures like *Amazon Mechanical Turk*, and it comes at a per-instance cost of labeling. Therefore, it is important to present judiciously chosen examples to the expert so that the decision boundary between the rare and normal class is learned with as few examples as possible. In such cases, label acquisition is combined with model construction in order to progressively incorporate more expert knowledge into the outlier analysis process.

Although most of this chapter will focus on settings in which ground-truth (supervision) is already available, this chapter will also investigate the connections between *unsupervised* outlier detection and supervised regression modeling. An interesting approach, which was proposed recently [417, 429], shows how one can use repeated applications of *off-the-shelf* regression models for unsupervised outlier detection. The basic idea is to use regression modeling to predict each of the attributes from the remaining attributes and then combining the errors of these models to create an outlier score (see section 7.7). This approach has the merit that it opens the door to the use of hundreds of off-the-shelf regression models for effective *unsupervised* outlier detection.

This chapter is organized as follows. The next section will discuss the problem of rare-class detection in the fully supervised scenario. The semi-supervised case of classification with positive and unlabeled data will be studied in section 7.3. Section 7.4 discusses the case in which only a subset of the classes are observed whether they are rare or normal. Unsupervised feature engineering methods for rare-class detection are discussed in section 7.5. Active learning methods are discussed in section 7.6. Section 7.7 shows how one can use repeated applications of off-the-shelf regression models for unsupervised outlier detection. The conclusions and summary are presented in section 7.8.

## 7.2 Full Supervision: Rare Class Detection

---

The problem of rare-class detection or *class imbalance* is a common one in the context of supervised outlier detection. The straightforward use of evaluation metrics and classifiers that are not cognizant of this class imbalance might lead to very surprising results. For example, consider a medical application in which it is desirable to identify tumors from medical scans. In such cases, 99% of the instances may be normal, and only the remaining 1% are abnormal.

Consider the trivial classification algorithm, in which every instance is labeled as normal without even examining the feature space. Such a classifier would have a very high absolute accuracy of 99%, but would not be very useful in the context of a real application. However, this (rather useless) classifier is often hard to outperform from an *absolute* accuracy point of view. In many cases, apparently “reasonable” algorithms also degrade to this trivial classifier in terms of performance. For example, consider a  $k$ -nearest neighbor classifier, in which the majority class label in the neighborhood is reported as the relevant class label. Because of the inherent bias in the class distribution, the majority class may very often be normal even for abnormal test instances. Such an approach fails because it does not account for

the *relative* behavior of the test instances with respect to the original class distribution. For example, if 49% of the training instances among the  $k$ -nearest neighbors of a test instance are anomalous, then that instance is much more likely to be anomalous *relative* to its original class distribution. By modifying the prediction criterion, such as reporting a non-majority anomalous class as the relevant label of a test instance, it is possible to improve the classification accuracy of anomalous classes at the expense of the accuracy on the normal class. Because of the predominance of the normal class in accuracy computation, such a classifier will almost always have lower accuracy than the aforementioned trivial classifier; yet, it is more *useful* from an application-specific perspective. Therefore, the question arises whether the use of measures such as overall classification accuracy is meaningful in the first place. It is important to understand that the issue of *evaluation* and *model construction* are closely related in all learning problems. In order to design useful rare-class detection models, one must first design meaningful evaluation mechanisms from an application-specific perspective. In the rare-class setting, the use of absolute accuracy is not a meaningful evaluation mechanism.

The proper evaluation mechanisms in these settings weigh the errors of anomalous instances differently from those of normal instances. The basic assumption is that it is more costly to misclassify anomalous instances as compared to normal instances. For example, misclassifying a fraudulent transaction (which could lead to millions of dollars in losses) is more costly than misclassifying a normal transaction (which causes annoyance to the end-user being incorrectly warned). Therefore, the model should optimize a cost-weighted variant of the accuracy. The underlying algorithms are also changed to incorporate these modified modeling assumptions. There are two primary types of algorithms that are popular in class-imbalanced settings:

- **Cost-sensitive learning:** The objective function of the classification algorithm is modified in order to weight the classification errors in a differential way for the normal classes and the rare classes. The typical assumption is that the misclassification of a rare class incurs higher costs. In many cases, this change in the objective function requires only minor changes to existing classification models.
- **Adaptive re-sampling:** The data are re-sampled so as to magnify the *relative proportion* of the rare classes. Such an approach can be considered an *indirect form* of cost-sensitive learning, since data re-sampling is equivalent to implicitly assuming higher costs for misclassification of rare classes. After all, the presence of a larger *relative* number of instances of a particular class in the sample (with respect to original data) tends to bias the prediction algorithms in favor of that class.

Both these methodologies will be discussed. A working knowledge of classification methods is required to understand the material in this chapter. The reader is also referred to [176] for a description of the different types of classifiers.

For the discussion in this section, it is assumed that the training data set is denoted by  $\mathcal{D}$ , and the label indices are denoted by  $L = \{1, \dots, k\}$ . Without loss of generality, it can be assumed that the normal class takes on the label-index of 1 in the set  $L$  and the remaining classes, which are presumably rare, are indexed from 2 to  $k$ . The  $i$ th record is denoted by  $\bar{X}_i$ , and its label  $l_i$  is drawn from  $L$ . The number of records belonging to the  $i$ th class is denoted by  $N_i$ . The total number of instances in the data is denoted by  $N$ , and therefore we have  $\sum_{i=1}^k N_i = N$ . The class imbalance assumption implies that  $N_1 \gg N - N_1$ . It is also possible for the various anomalous classes to be imbalanced with respect to one another. For example, in a network intrusion-detection application, the intrusions of various types may have widely varying frequencies.

### 7.2.1 Cost-Sensitive Learning

In cost-sensitive learning, the goal is to learn a classifier that maximizes the weighted accuracy over the different classes. The *misclassification cost* of the  $i$ th class is denoted by  $c_i$ . Some models [175] use a  $k \times k$  cost matrix to represent all the pair-wise values of misclassification costs (i.e., cost of misclassifying class  $i$  to class  $j$ ). In such models, the cost is dependent not only on the class identity of the misclassified instance, but is also dependent on the specific class label *to which* it is misclassified. In this chapter, we consider a simpler model in which the misclassification cost depends only on the origin class to which the instance belongs. The destination class is not relevant to the cost. Such a simplified model is more relevant to the rare-class detection problem. Therefore, we can denote the misclassification cost of the  $i$ th class by  $c_i$ . The goal of the classifier is to learn a training model that minimizes the *weighted misclassification rate*. Therefore, if  $n_i \leq N_i$  is the number of examples misclassified for the  $i$ th class, the goal is to minimize the following objective function:

$$J = \sum_{i=1}^k c_i \cdot n_i \quad (7.1)$$

The only difference from the traditional classification accuracy measure is the use of the weights  $c_1 \dots c_k$  in the objective function.

The choice of  $c_i$  is regulated by application-specific requirements, and is therefore a part of the input. However, in the absence of this input, some natural assumptions are sometimes used. For example, by choosing the value of  $c_i$  to be proportional to  $1/N_i$ , the *aggregate* impact of the instances of each class on the weighted misclassification rate is the same, in spite of the imbalance between the classes. Such methods are at best rule-of-thumb techniques for addressing imbalance, although more principled methods also exist in the literature. For example, the work in [601] proposes methods to learn the costs directly in a data-driven manner.

#### 7.2.1.1 MetaCost: A Relabeling Approach

A general framework known as *MetaCost* [175] uses a *relabeling* approach to classification. This is a *meta-algorithm*, which can be wrapped around any existing classification algorithm. In this method, the idea is to relabel some of the training instances in the data, by using the costs, so that normal training instances that have a reasonable probability of classifying to the rare class are relabeled to that rare class. Of course, rare classes may also be relabeled to a normal class, but the cost-based approach is *intended* to make this less likely. Subsequently, a classifier can be used on this more balanced training data set. The idea is to use the costs in order to move the decision boundaries in a cost-sensitive way, so that normal instances have a greater chance of misclassification than rare instances, and the *expected misclassification cost* is minimized.

In order to perform the relabeling, the classifier is applied to each instance of the training data and its classification prediction is combined with costs for re-labeling. Consider a setting in which a classifier predicts class label  $i$  with probability  $p_i(\bar{X})$  for the data instance  $\bar{X}$ . Then, the expected misclassification cost of the prediction of  $\bar{X}$ , *under the hypothesis that it truly belonged to class index  $r$* , is given by  $\sum_{i \neq r} c_i \cdot p_i(\bar{X})$ . Clearly, one would like to minimize the expected misclassification cost of the prediction. Therefore, the *MetaCost* approach tries different hypothetical classes for the training instance, and relabels it to the class that minimizes the expected misclassification cost. A key question arises as to how the probability  $p_i(\bar{X})$  may be estimated from a classifier. This probability clearly depends on



the specific classifier at hand. While some classifiers explicitly provide a probability score, this is not true in general. The work in [175] proposes a bagging approach [98] in which the training data is sampled with replacement (bootstrapping), and a model is repeatedly constructed on this basis. The size of the bootstrapped sample might be smaller than the training data to improve efficiency. These models are used to classify the training instances repeatedly. The fraction of predictions (or votes) for each class across different training data samples is used as its classification probability.

The challenge of such an approach is that relabeling training data is always somewhat risky, especially if the bagged classification probabilities do not reflect intrinsic classification probabilities. In fact, each bagged prediction is *highly correlated* with the others (since they share common training instances), and therefore the aggregate estimate is not a true probability. In practice, the estimated probabilities are likely to be very skewed towards one of the classes, which is typically the normal class. For example, consider a scenario in which a rare-class instance (with global class distribution of 1%) is present in a local region with 15% concentration of rare-class instances. Clearly, this rare instance shows informative behavior in terms of *relative* concentration of the rare class in the locality of the instance. A vanilla 20-nearest neighbor classifier will virtually *always*<sup>1</sup> classify this instance to a normal class in a large bootstrapped sample. This situation is not specific to the nearest-neighbor classifier. For example, an unmodified Bayes classifier will usually assign a lower probability to the rare class because of the much lower a priori probability of the rare class. Consider a situation in which a Bayes classifier assigns a posterior probability of 30% to a rare-class instance and the prior probability of the rare class is only 1%. In spite of increasing the prior probability by a factor of 30, a classifier will typically assign far fewer than 30% of the votes to the rare class in a bagged prediction, especially<sup>2</sup> when large bootstrap samples are used. In such cases, the normal class will win every time in the bagging because of the prior skew.

This suggests that the effect of cost weighting can sometimes be overwhelmed by the erroneous skews in the probability estimation attained by bagging. In this particular example, even with a cost ratio of 100 : 1, the rare-class instance will be wrongly relabeled to a normal class. This moves the classification boundaries in the opposite direction of what is desired. In fact, in cases where the unmodified classifier degrades to a trivial classifier of always classifying to the normal class, the expected misclassification cost criterion of [175] will result in relabeling all rare-class instances to the normal class, rather than the intended goal of selective relabeling in the other direction. In other words, relabeling may result in a further *magnification* of the errors arising from class skew. This leads to degradation of classification accuracy, *even from a cost-weighted perspective*. This problem is caused by the fact that bagging is an extremely imperfect simulation of sampling from a base distribution; it works in some restricted settings like ensemble learning but one cannot generalize this principle to arbitrary applications which are heavily susceptible to the correlations in predictions across different bags.

---

<sup>1</sup>The probability can be (approximately) computed from a binomial distribution to be at least equal to  $\sum_{i=0}^9 \binom{20}{i} \cdot 0.15^i \cdot 0.85^{20-i}$  and is greater than 0.999. This example also suggests that it is extremely important to use very unstable classifiers with this approach. An example would be to use a 1-nearest neighbor classifier.

<sup>2</sup>The original idea of bagging was not designed to yield class probabilities [98]. Rather, it was designed to perform robust prediction for instances, where either class is an almost equally good fit. In cases, where one of the classes has a “reasonably” higher (absolute) probability of prediction, the bagging approach will simply boost that probability to almost 1, when counted in terms of the number of votes. In the rare-class scenario, it is expected for unmodified classifiers to misclassify rare classes to normal classes with “reasonably” higher probability.

In general, it is possible for some classification algorithms to work effectively with meta-cost under the following conditions:

1. It is extremely important to use an unstable algorithm as the base classifier. Stable algorithms will lead to probability estimates that are close to 0 or 1.
2. Even though the approach in [175] proposes the use of smaller bootstrapped samples solely for efficiency considerations, an unobserved side-effect is that it will reduce overlap among different samples. Reducing overlap among samples is helpful for reducing correlation among classifiers, which is likely to improve the probability estimation. Smaller samples will also lead to unstable classifiers.

Note, however, that excessive instability can also have detrimental effects on accuracy.

It is also possible to use soft estimates of classification probability. In the previous example of the nearest neighbor classifier, if the *fraction* of the 20-nearest neighbors belonging to a class are used as its probability estimate for relabeling, then much more robust results can be obtained with *MetaCost*. Therefore, the effectiveness of *MetaCost* depends on the quality of the probability estimate used for re-labeling. Of course, if good probability estimates are directly available from the training model in the first place, then a test instance may be directly predicted using the expected misclassification cost, rather than using the indirect approach of trying to “correct” the training data by re-labeling. This is the idea behind weighting methods, which will be discussed in the next section.

### 7.2.1.2 Weighting Methods

Most classification algorithms can be modified in natural ways to account for costs. The primary driving force behind these modifications is to implicitly treat each training instance with a weight, where the weight of the instance corresponds to its misclassification cost. This modification is directly motivated by the presence of costs in the accuracy objective of Equation 7.1 that needs to be optimized. This leads to a number of simple modifications to the underlying classification algorithms. In the following, a discussion is provided about the natural modifications to the more common classification algorithms.

#### Bayes Classifier

The modification of the Bayes classifier provides the simplest case for cost-sensitive learning. In this case, changing the weight of the example only changes the a priori probability of the class, and all other terms within the Bayes estimation remain the same. In other words, the prior probability in the unweighted case needs to be multiplied with the cost. Note that this criterion is also used in *MetaCost*, although the latter uses this criterion for *relabeling* training instances rather than predicting test instances. When good probability estimates are available from the Bayes classifier, it makes more sense to use them to directly predict test instances rather than to relabel training instances.

#### Proximity-Based Classifiers

In nearest-neighbor classifiers with binary classes, the classification label of a test instance is defined as the majority class from its  $k$  nearest neighbors. In the context of *cost-sensitive* classification, the *weighted* majority label is reported as the relevant one, where the weight of an instance from class  $i$  is denoted by  $c_i$ . The majority class is therefore selected by reporting the class with the maximum *weight* as the relevant one. Because of the weighting,

a test instance may be classified as an anomaly, even when fewer examples of the rare class are present in its neighborhood. A discussion of methods for  $k$ -nearest neighbor classification in the context of data classification may be found in [609].

### Rule-Based Classifiers

Many rule-based classifiers leverage association rule mining with support and confidence parameters. A rule relates a condition in the data (e.g., ranges on numeric attributes) to a class label. The condition occurs on the left-hand side (antecedent) of the rule, whereas the class label occurs on the right-hand side (consequent) of the rule. The support of a rule is defined as the number of training instances that are relevant to that rule. The confidence of a rule is the fractional probability that the training instance belongs to the class on the right-hand side if it satisfies the conditions on the left-hand side. The data is first discretized, and all relevant rules are mined from it at pre-specified levels of support and confidence. These rules are then prioritized based on the underlying confidence (and sometimes also the support). For a given test instances, all the relevant rules are identified as the rules whose antecedents match the test instance. These rules might sometimes predict conflicting class labels. Therefore, the results from different rules can be combined using a variety of heuristics (e.g., majority class from relevant rules or top-matching rule) in order to yield the final class label.

Such an approach is not difficult to adapt to the cost-sensitive case. The main adaptation is that the weights on the different training examples need to be used during the computation of measures such as the support or the confidence. Clearly, when rare examples are weighted more heavily, the support and confidence of a rule will be much higher for rules with the rare class in the consequent. This will result in the selective emphasis of rules corresponding to prediction of rare instances. Some methods for using rule-based methods in imbalanced data classification are proposed in [298, 300].

### Decision Trees

In decision trees, the training data is recursively partitioned, so that the instances of different classes are successively separated out at lower levels of the tree. The partitioning is performed by using conditions on one or more features in the data. Typically, the split criterion uses the various entropy measures such as the Gini index for deciding the choice of attribute and the position of the split. For a node containing a fraction of instances of different classes denoted by  $p_1 \dots p_k$ , its Gini index is denoted by  $1 - \sum_{i=1}^k p_i^2$ . Better separations of different classes lead to lower Gini index. Typically, the data is first discretized and then converted into a binary representation. The split attribute is the one that minimizes the Gini index of the children nodes. By using costs as weights for the instances, the values of the fractions  $p_1 \dots p_k$  are computed. This will also impact the computation of the Gini index, so as to selectively create nodes in which a higher proportion of data points belong to the rare class. This type of approach generally tends to lead to better separation between the normal class and the rare class. In cases where leaf nodes do not exclusively belong to a particular class, instances are weighted by their misclassification costs for labeling that leaf node. Some examples of cost-sensitive decision trees are discussed in [546, 566].

### Support-Vector Machines

Support-vector machine (SVM) classifiers work by learning hyperplanes that optimally separate the two classes in order to minimize a carefully defined penalty function. This penalty

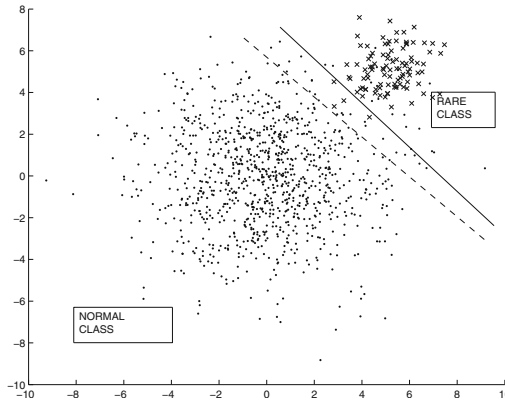


Figure 7.1: Optimal hyperplanes will change because of weighting of examples

function penalizes training data points for being on the wrong side of the decision boundary and for even being close to the decision boundary (on the correct side). The notion of “closeness” to the decision boundary is defined by the *margin*, which is simultaneously maximized. Thus, an SVM classifier can be modeled as an optimization problem, in which the goal is to learn the coefficients of the underlying hyperplane in order to optimize a weighted combination of these two objectives. For example, a two-class example has been illustrated in Figure 7.1. The optimal separator hyperplane for the two classes is illustrated in the same figure with the solid line. However, it is possible to change the optimization model by incorporating weights (or costs) into the optimization problem. Therefore, violation of the linear margin constraints incurs a cost of 1 for the normal class, whereas it incurs a cost of  $c_2/c_1$  for the rare class. Since it is assumed that  $c_2 \gg c_1$ , this differential weighting shifts the decision boundary, so as to allow erroneous classification of a larger number of normal instances, while correctly classifying more rare instances. The result would be a reduction in the overall classification accuracy, but an increase in the cost-sensitive accuracy. For example, in the case of Figure 7.1, the optimal separator hyperplane would move from the solid line to the dotted line in the figure. The issue of class-boundary re-alignment for SVMs in the context of imbalanced data sets has been explored in detail in [535, 570]. These methods use class-biased penalties during the SVM model creation. At a basic level, the linear decision boundary of the SVM may be written as follows:

$$\overline{W} \cdot \overline{X} + b = 0 \quad (7.2)$$

Here, the coefficient vector  $\overline{W}$  and the bias  $b$  defines the hyperplane. Normal data points satisfy  $\overline{W} \cdot \overline{X} + b \geq 0$  and anomalous data points satisfy  $\overline{W} \cdot \overline{X} + b \leq 0$ . Therefore, we punish the violation of these constraints. In the cost-sensitive setting, the normal points are punished by 1 unit, whereas the violations on rare-class instances are punished by  $\frac{c_2}{c_1} > 1$ . Note that the differential costs in the two cases will push the decision boundary towards making fewer errors on the rare class. Furthermore, we not only punish the violation of the decision boundary, but also the presence of points that are too close to the decision boundary (on the correct side). In other words, we penalize the violation of even stricter constraints than the one implied by the linear decision boundary. These stricter constraints correspond to two parallel hyperplanes on either side of the decision hyperplane, which are  $\overline{W} \cdot \overline{X} + b \geq 1$  (for normal points) and  $\overline{W} \cdot \overline{X} + b < -1$  (for anomalous points). The goal is to ensure that very few points lie between these two hyperplanes. This is achieved by using

penalties to discourage points from lying in these regions, no matter which class they might belong to.

Consider a data set  $\mathcal{D}$  with anomalous points  $\mathcal{O}$ . Then, the penalty for the normal points is given by  $\sum_{\bar{X} \in \mathcal{D} - \mathcal{O}} \max\{0, 1 - \bar{W} \cdot \bar{X} - b\}$  and the penalty for the anomalous points is given by  $\frac{c_2}{c_1} \sum_{\bar{X} \in \mathcal{O}} \max\{0, 1 + \bar{W} \cdot \bar{X} + b\}$ . In addition, we add a regularization term  $\lambda \|\bar{W}\|^2$ . Here,  $\lambda > 0$  is the regularization parameter. Therefore, the overall objective function may be written as follows:

$$J = \underbrace{\sum_{\bar{X} \in \mathcal{D} - \mathcal{O}} \max\{0, 1 - \bar{W} \cdot \bar{X} - b\}}_{\text{Penalty (normal points)}} + \underbrace{\frac{c_2}{c_1} \sum_{\bar{X} \in \mathcal{O}} \max\{0, 1 + \bar{W} \cdot \bar{X} + b\}}_{\text{Penalty (anomalous points)}} + \underbrace{\lambda \|\bar{W}\|^2}_{\text{Regularizer}} \quad (7.3)$$

The differential weighting of the costs is particularly notable in this setting. Note that one can easily solve this optimization problem with gradient-descent. However, if kernel methods are to be used to determine non-linear decision boundaries, then a dual formulation is used frequently. Discussions of the dual formulation for SVMs may be found in [33], and its modification to the aforementioned cost-sensitive case is straightforward.

### 7.2.2 Adaptive Re-sampling

It is noteworthy that most of the weighting methods simply increase the weight of the rare class in order to bias the classification process towards classifying rare-class instances correctly. In adaptive re-sampling, a similar goal of increasing the impact of the rare class is achieved by differentially sampling the training data in favor of the rare class. After all, classification models are directly influenced by the frequency of presence of each class. For example, in the case of the naive Bayes classifier, the prior probabilities are proportional to the relative presence of each class.

Sampling can be performed either with or without replacement. Either the rare class can be oversampled, or the normal class can be under-sampled, or both types of sampling can be simultaneously executed. Sampling with replacement is necessary when it is needed to oversample a class. The classification model is learned on the re-sampled data. The sampling probabilities are typically chosen in proportion to their misclassification costs. This enhances the proportion of the rare-class costs in the sample used for learning. It has generally been observed [173], that under-sampling has a number of advantages over over-sampling. When under-sampling is used, the sampled training data is much smaller than the original data set. In some variations, all instances of the rare class are used in combination with a small sample of the normal class [124, 332]. This is also referred to as *one-sided selection*. Under-sampling has several advantages:

- The model construction phase for a smaller training data set requires much less time.
- The normal class is less important for modeling purposes, and most of the rare class is included for modeling. Therefore, the discarded instances do not take away too much from the modeling effectiveness.
- Because of the improved efficiency, one can draw multiple subsamples and average the predictions in order to create a more robust ensemble model.

Sampling methods are often used in the context of subsampling ensembles in classification [105, 106, 107], in which the samples are repeatedly drawn in order to create multiple

models. The prediction of a test point is obtained by averaging the predictions from different models. This type of approach significantly improves the accuracy of the base method because of its variance reduction effects of the ensemble method. This can be viewed as the supervised avatar of the subsampling approach discussed in Chapter 6.

### 7.2.2.1 Relationship between Weighting and Sampling

Since cost-sensitive learning can be logically understood as methods that weight examples of various classes in a differential way, a question arises as to how these methods relate to one another. Adaptive re-sampling methods can be understood as methods that sample the data in proportion to their weights, and then treat all examples equally. From a practical perspective, this may often lead to similar models in the two cases, although sampling methods may throw away some of the relevant data. It should also be evident that a direct weight-based technique retains more information about the data, and is therefore likely to be more accurate if *only a single instantiation is used in both cases*. This seems to be the case from many practical experiences with real data [132]. On the other hand, adaptive re-sampling has distinct *efficiency* advantages because it works with a much smaller data set. For example, for a data set containing 1% of labeled anomalies, it is possible for a re-sampling technique to work effectively with 2% of the original data, when the data is re-sampled into an equal mixture of the normal and anomalous classes. This translates to a performance improvement of a factor of 50. This means that we can use 50 different instantiations of the sampling method at the same cost of one instantiation of the weighting method. By averaging the results from these different instantiations, one will typically obtain more accurate results with the use of sampling than in the case of weighting. In most cases, a small number of ensemble components (between 10 and 25) are required in order to gain most of the accuracy advantages.

In addition, a special type of ensemble known as the *sequential ensemble* has also been proposed in [371]. In the sequential ensemble, the choice of the majority class instances selected in a given iteration is regulated by the predictions of the classifier during previous iterations. Specifically, only majority instances that are correctly classified by the classifier in a given iteration are not included in future iterations. The idea is to reduce the redundancy in the learning process, and improve the overall robustness of the ensemble. Note that this is a *supervised* sequential ensemble, and is exactly analogous to the sequential ensemble method introduced in Chapter 1 for general-purpose outlier analysis.

### 7.2.2.2 Synthetic Over-sampling: SMOTE

Over-sampling methods are also used in the literature, though less frequently so than under-sampling. One of the problems of over-sampling the minority class is that a larger number of samples with replacement leads to repeated samples of the same data point. This could lead to over-fitting and deteriorated bias-centric effects on accuracy. In order to address this issue, it was suggested [133] that synthetic over-sampling could be used to create the over-sampled examples in a way which provides better accuracy.

The *SMOTE* approach works as follows. For each minority instance, its  $k$  nearest neighbors are found. Then, depending upon the level of over-sampling required, a fraction of them are chosen randomly. A synthetic data example is generated on the line segment connecting that minority example to its nearest neighbor. The exact position of the example is chosen uniformly at random along the line segment. In addition to over-sampling the rare class, the *SMOTE* algorithm under-samples the minority class. The *SMOTE* algorithm has been

shown to provide more robust over-sampling than a vanilla over-sampling approach. This approach forces the decision region of the re-sampled data to become more general than one in which only members from the rare classes in the *original* training data are over-sampled. Another advantage of the *SMOTE* algorithm is that it can be easily combined with boosting algorithms to create more robust methods like *SMOTEBoost* (see next section). Although it has not been discussed in the original paper, a simpler approach for creating a very robust ensemble is to average the point-wise predictions from various (randomized) *SMOTE* instantiations; we refer to this variation as *SMOTEAverage*.

### 7.2.3 Boosting Methods

Boosting methods are commonly used in classification in order to improve the classification performance on difficult instances of the data. The well-known *Adaboost* algorithm [478] works by associating each training example with a *weight* that is updated in each iteration depending on the accuracy of its prediction in the previous iteration. Misclassified instances are given larger weights. The idea is to give greater importance to “difficult” instances which may lie near the decision boundaries of the classification process. The overall classification results are computed as a combination of the results from different rounds. The basic idea is that the classifier has a bias caused by the incorrect shape of this decision boundary. By combining the predictions from multiple training models on biased data sets, one is able to roughly approximate the shape of the decision boundary correctly. In other words, a combination of the results on a set of biased *data sets* are used to correct for the inherent bias in a particular *model*.

Next, we describe the boosting method in more detail. In the  $t$ th round, the weight of the  $i$ th instance is  $D_t(i)$ . The algorithm is initialized with an equal weight of  $1/N$  for each of the  $N$  instances, and updates them in each iteration depending on the correctness of the classification model (on the training instances). In practice, it is always assumed that the weights are normalized in order to sum to 1, although the approach will be described below in terms of (unscaled) relative weights for notational simplicity. In the event that the  $i$ th example is misclassified in the  $t$ th iteration, its (relative) weight is increased to  $D_{t+1}(i) = D_t(i) \cdot e^{\alpha_t}$ . In the case of a correct classification, the weight is decreased to  $D_{t+1}(i) = D_t(i) \cdot e^{-\alpha_t}$ . Here,  $\alpha_t$  is chosen as the function  $(1/2) \cdot \ln((1 - \epsilon_t)/\epsilon_t)$ , where  $\epsilon_t$  is the fraction of incorrectly predicted instances on a weighted basis. The final result for the classification of a test instance is a weighted prediction over the different rounds, where  $\alpha_t$  is used as the weight for the  $t$ th iteration. The weighting is typically implemented using sampling.

The rare-class setting also has an impact on the algorithmic design of boosting methods. A specific example is the *AdaCost* method [191], which uses the costs to update the weights. In this method, instead of updating the misclassified weights for instance  $i$  by the factor  $e^{\alpha_t}$ , they are instead updated by  $e^{\beta_-(c_i) \cdot \alpha_t}$ , where  $c_i$  is the cost of the  $i$ th instance. Note that  $\beta_-(c_i)$  is a function of the cost of the  $i$ th instance and serves as the “adjustment” factor, which accounts for the weights. For the case of correctly classified instances, the weights are updated by the factor  $e^{-\beta_+(c_i) \cdot \alpha_t}$ . Note that the adjustment factor is different depending on whether the instance is correctly classified. This is because for the case of costly instances, it is desirable to increase weights more than less costly instances in case of misclassification. On the other hand, in cases of correct classification, it is desirable to reduce weights less for more costly instances. In either case, the adjustment is such that costly instances get relatively higher weight in later iterations. Therefore,  $\beta_-(c_i)$  is a non-decreasing function with cost, whereas  $\beta_+(c_i)$  is a non-increasing function with cost. A different way to perform

the adjustment would be to use the same exponential factor for weight updates as the original *Adaboost* algorithm, but this weight is further multiplied with the cost  $c_i$  [191], or other non-decreasing function of the cost. Such an approach would also provide higher weights to instances with larger costs. The use of boosting in weight updates has been shown to significantly improve the effectiveness of the imbalanced classification algorithms.

Boosting methods can also be combined with synthetic oversampling techniques. An example of this is the *SMOTEBoost* algorithm, which combines synthetic oversampling with a boosting approach. A number of interesting comparisons of boosting algorithms are presented in [299, 301]. A general observation about boosting methods is that their primary focus is on bias correction. Therefore, it is important to use classifiers with high bias and low variance (e.g., linear SVM instead of kernel SVM). This is because we want to ensure that the incorrect classification of training examples is caused by bias rather than variance. In cases where the classifier has high variance, the boosting method could preferentially overweight examples based on random variations in algorithm output or other noisy examples. Similarly, boosting methods tend to work poorly in data sets with a large amount of noise.

### 7.3 Semi-Supervision: Positive and Unlabeled Data

---

In many data domains, the positive class may be easily identifiable, though examples of the negative class may be much harder to model simply because of their diversity and inexact modeling definition. Consider, for example, a scenario in which it is desirable to classify or collect all documents that belong to a rare class. In many scenarios, such as the case of Web documents, the types of the documents available are too diverse. As a result, it is hard to define a representative negative sample of documents from the Web.

This leads to numerous challenges at the *data acquisition stage*, in which the collection of negative examples is difficult. The problem is that the universe of instances in the negative class is rather large and diverse, and the collection of a representative sample may be difficult. For very large-scale collections such as the Web and social networks [595], this scenario is quite common. Although a number of methods have been proposed for negative data collection in such unstructured domains, none of which are fully satisfactory in terms of being *truly representative* of what one might encounter in a real application. For example, for Web document classification, one simple option would be to simply crawl a random subset of documents off the Web. Nevertheless, such a sample would contain contaminants which do belong to the positive class, and it may be hard to create a purely negative sample, unless a significant amount of effort is invested in cleaning the sample of the negative class. The amount of human effort involved in labeling is especially high because the vast majority of examples are negative, and a manual process of filtering out the positive examples would be too slow and tedious. Therefore, a simple solution is to use the sampled background collection as the unlabeled class for training, and simply tolerate the presence of positive contaminants. This could lead to two different types of challenges:

- The contaminants in the negative class can reduce the effectiveness of a classifier, although it is still better to use the contaminated training examples rather than to completely discard them. This setting is referred to as *positive-unlabeled (PUC)* data classification.
- The collected training instances for the unlabeled class may not reflect the true distribution of documents. In such cases, the classification accuracy may actually be *harmed*



by using the negative class [356]. Therefore, there are a few algorithms that discard the unlabeled data altogether and work with only the positive class.

A number of methods have been proposed in the literature for this variant of the classification problem, which can address the aforementioned issues.

Although some methods in the literature treat this as a new problem that is distinct from the fully supervised classification problem [362], other methods [183] recognize this problem as a noisy variant of the classification problem, to which traditional classifiers can be applied with some modifications. An interesting and fundamental result proposed in [183] is that the accuracy of a classifier trained on this scenario differs by only a constant factor from the true conditional probabilities of being positive. These results provides strong support for the view that learning from positive and unlabeled examples is essentially equivalent to learning from positive and negative examples.

There are two types of methods that can be used in order to address the problem of contaminated normal class examples. In the first class of methods, heuristics are used in order to identify (unlabeled) training examples that are negative (i.e., normal). Subsequently, a classifier is trained on the positive examples, together with the unlabeled examples that have been *reliably* predicted to be negative. A less common approach is to assign weights to the unlabeled training examples [348, 362]. The second case is a soft version of the first, because the first setting can be viewed as an approach in which binary weights are used. It has been shown in the literature [363], that the second approach is superior. An SVM approach is used in order to learn the weights. Another work in [603] uses a probabilistic approach in which the weight vector is learned in order to provide robust estimates of the prediction probabilities.

Although the use of unlabeled data as the negative class is adequate in most settings, this is not always true. In some scenarios, the unlabeled class in the training data reflects the behavior of the negative class in the test data very poorly. In such cases, it has been shown that the use of the negative class actually *degrades* the effectiveness of classifiers [356]. Therefore, it may be better to simply discard the negative examples and build the model only from the anomalous examples (cf. section 7.4.1).

## 7.4 Semi-Supervision: Partially Observed Classes

---

The data might contain a normal class and several anomalous classes. In such cases, several interesting settings might arise in which some of the classes are observed, whereas others are not. For example, one might have examples of only the anomalous class, only the rare class, or a combination of some subset of the rare classes together with the normal class. In this context, it is noteworthy that missing anomalous classes are more common than missing normal classes. This is because anomalous classes are naturally so rare that it is difficult to collect examples of their occurrence. Some examples of various scenarios of missing classes are as follows:

- In some cases, it is difficult to obtain a clean sample of normal data, when the background data is too diverse or contaminated. In such cases, the anomalous class may be observed but the normal class may be missing.
- In a bio-terrorist attack application, it may be easy to collect normal examples of environmental variables, but no explicit examples of anomalies may be available, if an attack has never occurred.

- In an intrusion or viral attack scenario, many examples of normal data and previous intrusions or attacks may be available, but new forms of intrusion may arise over time.

These versions of the problem are truly semi-supervised because training data are available about some portions of the data but not others. The case of positive and unlabeled data is much simpler because the unlabeled class can often be approximated as a negative class. Therefore, such scenarios are best addressed with a combination of supervised and unsupervised techniques. The cases in which examples of anomalous class are available are very different from those in which examples of only the normal class are available. In the former, outliers are points that are as *similar* as possible to the training data, whereas in the latter, outliers are points that are as *different* as possible from the training data. In cases where examples of only the normal class are available, the distinction from unsupervised outlier detection is minimal. A distinction between unsupervised outlier detection and one-class *novelty* detection is that the term “novelty” is often used in a temporal context in which the normal examples have arrived in the past.

### 7.4.1 One-Class Learning with Anomalous Examples

In most natural settings, the anomalous class is missing, and copious examples of the normal class may be available. There are a few cases in which only the anomalous class is available, although such situations are rare. As a result, there are very few methods addressing this setting. Such a problem may sometimes occur naturally in scenarios where the background class is too diverse or noisy to be sampled in a meaningful way. Therefore, one cannot sample the background class to create unlabeled instances and use positive-unlabeled learning methods.

In such cases, unsupervised models can be constructed on the subset of the data corresponding to the positive class. The major difference is that *higher fit* of the data to the positive class corresponds to greater outlier scores. This is the reverse of what is normally performed in outlier detection. The assumption is that the representative data contains only anomalies, and therefore outliers are more likely to be similar to this data. Proximity-based outlier detectors are very natural to construct in the one-class scenario, because the propensity of a test instance to belong to a class can be naturally modeled in terms of distances. Therefore, a data point may be scored on the basis of its average  $k$ -nearest neighbor distance, except that *lower* scores indicate a greater degree of outlierness. It is much harder to adapt other outlier detection methods to this variation of the one-class setting because of paucity of data and overfitting.

Another possibility is the adaptation of one-class SVMs to this setting. However, the SVM method in section 3.4 of Chapter 3 assumes the availability of examples from the normal class rather than the anomalous class. The main difference is that we are now looking for test points that are classified on the same side of the separator as the observed points (which are known to be anomalous). It is, however, much harder to learn this type of classifier because examples of the anomalous class are usually scarce. The use of a kernel function would typically cause overfitting. Certain variations of one-class SVMs have been shown [113] to work effectively in this setting (as well as the original setting of normal points only). This approach determines a linear separator that is attracted towards the center of the points in kernel feature space, rather than repelled away from the origin. The work in [113] shows that this approach can detect abnormalities in ball-bearing cages, when only examples of anomalies (faulty cages) are available.

### 7.4.2 One-Class Learning with Normal Examples

A more common setting is one in which only normal examples are available and none of the rare classes are observed. This problem is, of course, no different from the unsupervised outlier detection setting. *In fact, any of the unsupervised models for outlier detection can be used in this case.* The major difference is that the training data is guaranteed to contain only the normal class, and therefore the outlier analysis methods are likely to be more robust. Strictly speaking, when only examples of the normal class are available, the problem is hard to distinguish from the unsupervised version of the problem, at least from a methodological point of view. From a *formulation* point of view, the training and test records are not distinguished from one another in the unsupervised case (any record can be normal or an anomaly), whereas the training (only normal) and test records (either normal or anomaly) are distinguished from one another in the semi-supervised case.

Virtually all unsupervised outlier detection methods attempt to model the normal behavior of the data, and can be used for novel class detection, especially when the only class in the training data is the normal class. *Therefore, the distinction between normal-class only variation of outlier detection and the unsupervised version of the problem is limited and artificial, especially when other labeled anomalous classes do not form a part of the training data.* In spite of this, the semi-supervised version of the (normal-class only) problem seems to have a distinct literature of its own, which is somewhat unnecessary, since any of the unsupervised algorithms can be applied to this case. The main difference is that the training and test data are distinguished from one another, and the outlier score is computed for a test instance with respect to the training data. In general, when working with inductive learning methods for unsupervised outlier detection (like one-class SVMs), it is always a good practice to distinguish between training and test data (by cross-validation) even when they are not distinguished in the original data. This helps in avoiding overfitting.

### 7.4.3 Learning with a Subset of Labeled Classes

A more challenging scenario arises when labeled rare classes are present in the training data, but new (i.e., *novel*) classes appear from time to time. Such scenarios can arise quite often in many applications such as intrusion detection in which partial knowledge is available about *some* of the anomalies, but others may need to be modeled in an unsupervised way. For a given test point, two key decisions need to be made in the following order:

1. Is the test point a natural fit for a model of the training data? This model also includes the currently occurring rare classes. A variety of unsupervised models such as clustering can be used for this purpose. If not, it is immediately flagged as an outlier or a novelty.
2. If the test point is a fit for the training data, then a classifier model is used to determine whether it belongs to one of the rare classes. Any cost-sensitive model (or an ensemble of them) can be used for this purpose.

Thus, this model requires a combination of unsupervised and supervised methods to identify outliers. The most common scenario for novel class detection occurs in the context of *online* scenarios in concept drifting data streams. In fact, novel class detection usually has an implicit assumption of *temporal* data, since classes can be defined as novel only in terms of what has already been seen in the *past*. In many of the aforementioned batch-centric algorithms, this temporal aspect is not fully explored because a single snapshot of training data is assumed. Many applications such as intrusion detection are naturally

focused on a streaming scenario. In such cases, novel classes may appear at any point in the data stream, and it may be desirable to distinguish different types of novel classes from one another [46, 391, 392]. Furthermore, when new classes are discovered, these kinds of anomalies may recur over time, albeit quite rarely. In such cases, the effectiveness of the model can be improved by keeping a memory of the *rarely recurring classes*. This case is particularly challenging because aside from the temporal aspects of modeling, it is desirable to perform the training and testing in an online manner, in which only one pass is allowed over the incoming data stream. This scenario is a true amalgamation of supervised and unsupervised methods for anomaly detection, and is discussed in detail in section 9.4.3 of Chapter 9.

## 7.5 Unsupervised Feature Engineering in Supervised Methods

---

Feature engineering is the problem of selecting the best representation of the data set for a particular model. For example, consider a classification problem in which the data is distributed according to a single Gaussian distribution. Furthermore, rare-class instances are located far away from the data mean in all directions, whereas normal-class instances are clustered near the data mean. This situation is quite conceivable in real settings. If we used a linear support vector machine to construct a training model, it would work very poorly because the rare and normal class are not linearly separable. However, if we extracted the distance of each point from the data mean as one of the features, then the linear support vector machine will provide perfect separation. This is an example of feature engineering, in which extracting a more convenient representation can often overcome the inherent representational limitations of a model.

Although it is possible to use more complex models (e.g., kernel methods) to overcome these types of limitations, there are several challenges in using these models such as computational complexity and overfitting. Overfitting is a common problem when there is a limited number of instances of any class; this is common in rare-class learning because of the paucity of anomalous examples. Kernel methods also cannot control the opaque representation created by the kernel function and are therefore unable to perform feature selection in a controlled way. Feature engineering provides additional flexibility in selecting the correct representation to use out of a set of “reasonable” features.

What is a reasonable representation in the rare-class setting? What types of features are most likely to emphasize the difference between the rare class and the normal class? It is here that unsupervised outlier detection methods can play an important role because the different unsupervised methods are designed to emphasize the “typically” rare characteristics of data sets [395]. In our previous example of outliers at the extremities of a multivariate Gaussian distribution, the outlier score of the linear Mahalanobis method would be an excellent feature to use. Since different outlier detection algorithms emphasize different types of rare characteristics (which may be suitable for different data sets), one can use multiple outlier detection algorithms to extract different features. These features can either replace or be added to the original data representation. A rare-class learning algorithm can be trained on the augmented representation in order to obtain results of better quality.

In order to extract different types of features, one can use the outlier scores from different models, different parameter settings of the same outlier detection model, or the execution of the same model on different feature bags. Since different outlier detection algorithms work well on different data sets, it is evident that the useful features will also be data-specific.

For example, the features extracted from LOF would be useful for a data set containing many local outliers, whereas the features extracted from the  $k$ -nearest neighbor method would be useful for a data set containing many global outliers. Serendipity also plays a role in extracting useful features. For example, the scores from a sampled feature bag may work very well for a particular data set, although there is some element of luck in being able to sample the correct bag.

Note that this approach leads to a significant redundancy in the feature representation of the rare-class learning problem. This is particularly true because many outlier detection models are similar and might lead to highly correlated features. However, one can use any number of supervised feature selection and regularization methods to minimize the effect of redundant features. For example, a linear support-vector machine or logistic regressor with  $L_1$ -regularization will automatically remove the effect of redundant or irrelevant features from the model. In this particular setting, the underlying data representation may be very high-dimensional, and therefore  $L_1$ -regularization may be more suitable than  $L_2$ -regularization.  $L_1$ -regularizers automatically set the coefficients of many features to 0 in a regression model, and therefore the effect of using the regularization is to perform guided feature selection. It has been shown in [395] that this type of feature engineering approach significantly enhances the representation of the data and therefore improves the classification results.

Why should such an approach work? Feature engineering methods work best when the features are encoded with an understanding of either the problem or application domain. It is noteworthy that the same data sets and evaluation measures are used for benchmarking both outlier detection and (supervised) rare-class detection problems. In many rare-class detection problems, unsupervised outlier detection algorithms achieve very high accuracy (AUC) values [35]. This is primarily because outlier detection algorithms have been developed over the years with a problem domain-specific understanding of the typical characteristics of anomalies. As a result, such features also enhance the effectiveness of supervised learning methods.

It is noteworthy that feature engineering methods can be either supervised or unsupervised. For example, this approach shares a number of similarities with a supervised feature engineering and ensemble method, referred to as *stacking* [33]. In stacking, the features are learned using the output of supervised classification algorithms rather than unsupervised outlier detection algorithms. Unsupervised feature engineering methods are often more effective when the amount of training data is small. If a lot of labeled data is available, then it is possible to partition the data between the feature learning part and the second-phase of prediction for better results (with stacking). In the rare-class setting, the amount of training data for at least one of the classes is very little. As a result, an unsupervised approach, such as the use of outlier scores, is appropriate in this setting.

## 7.6 Active Learning

---

It is expensive to manually label examples in order to identify the rare class. Therefore, the basic idea is to identify and label more *informative* examples that reduce the need to acquire too many of them. Such examples often occur near the decision boundary between the rare and normal class. In addition, suspected outliers are more valuable than suspected inliers for labeling, because the former are harder to acquire. By identifying such examples, it is possible to train classifiers with far fewer examples. This is particularly useful in reducing the cost of label acquisition. Therefore, the most important aspect of active learning is in

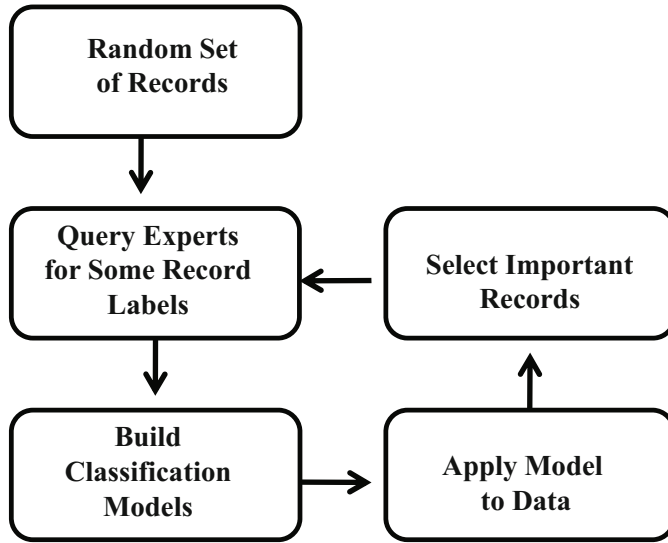


Figure 7.2: The overall procedure for active learning

identifying these critical examples near the decision boundary. Active learning is a common approach used in classification domains [11, 33, 486]. However, in anomaly and rare-class detection, a bigger issue is the *cold-start problem* in which few training examples are available at the very beginning. In fact, in a purely unsupervised setting, there might be no labeling to begin with. While the active learning for rare-class detection is not too different from the classification domain, there are some additional challenges from the cold-start point of view.

An interesting procedure for active learning from unlabeled data is proposed in [431]. An iterative procedure is used in order to label some of the examples in each iteration. In each iteration, a number of interesting instances are identified, for which the addition of labels would be helpful for further classification. These are considered the “important” instances. The human expert provides labels for these examples. These are then used in order to classify the data set with the augmented labels. The proper determination of the important instances is key to the effectiveness of this approach. An iterative approach is therefore used in which the examples are identified with the use of labels that are available, and these are then presented to the human expert in order to augment the labeled data.

In the setting discussed in [431], no labeled examples are available at the beginning of the process. The first iteration is special, in which a purely unsupervised approach is used for learning the important examples and presenting them to the user. The user labels these examples to provide the first set of training data. Subsequent iterations are similar to the traditional active learning setting for classification. The key lies in the process of properly selecting the examples to be labeled, which is discussed below. This procedure is performed iteratively until the addition of further examples is no longer deemed significantly helpful for further classification or a predefined labeling budget is exhausted. The overall procedure is illustrated in Figure 7.2. It should be noted that this approach can also be used in scenarios in which few positive examples are available to begin with.

A key question arises as to *which* examples should be presented to the user for the purposes of labeling. It is clear that examples that are predicted to be clearly positive or negative (based on current models) are not particularly useful to present to the user. Rather, it is the examples with the greatest *uncertainty* or *ambiguity* in prediction that should be presented to the user for labeling. The intuition is that such examples lie on the decision boundary and by labeling them, one can gain the greatest knowledge about the decision boundaries between the different classes. Such an approach maximizes the learning of the contours separating different classes, while requiring the least amount of (potentially expensive) expert supervision. In the context of rare-class detection, however, the greater value of labeling rare-class examples is also taken into account. Outliers are far more valuable than inliers as examples, and, therefore, if a point is suspected of being an outlier (but not included), it should be labeled. In other words, some subtle changes need to be made to the usual rules for selecting examples in the rare-class setting. In other words, the two key selection criteria are as follows [431]:

- *Low likelihood*: These are data points that have low fit to the model describing the data. For example, if an EM algorithm is used for modeling, then these are points that have low fit to the underlying model. Since these are suspected outliers, it would be desirable to label them.
- *High uncertainty*: These are points that have the greatest uncertainty in terms of their membership to either the rare class or the normal class. Therefore, by acquiring the label of this example, one learns something new about the decision space, thereby helping the underlying algorithms.

All data points are ranked on the basis of the two aforementioned criteria. The lists are merged by alternating between them, and adding the next point in the list, which has not already been added to the merged list. This ranking is used in order to select the next data point to be labeled by the expert.

Note that the low-likelihood criterion is specific to the rare class and outlier detection setting, whereas the high-uncertainty criterion is common to all classification settings. Several surveys [11, 486] on active learning in classification describe methods to quantify the uncertainty in prediction of a data point. Some examples of such quantifications are as follows:

- Consider a case in which a Bayes classifier predicts the probability of a point belonging to the positive class to be very similar to the (estimated) fraction of outliers in the data. In such a case, this example might be informative because it lies on the decision boundary between the rare and anomalous class. The estimated fraction of outliers in the base data can be computed based on domain knowledge. However, in the rare-class setting, the acquisition of rare examples is more critical. Therefore, as long as the Bayes classifier predicts the probability of a point belong to the rare class to be *greater* than the estimated fraction of outliers, the point is considered informative. In a sense, this approach merges the low-likelihood criterion with the high-uncertainty criterion in a seamless way.
- Another approach is the principle of *query by committee* [485]. An ensemble of classifiers is trained, and points with the greatest *predictive disagreement* (by different classifiers) are selected. The basic idea is that different models make varying predictions on data points near the decision boundary between the normal and anomalous class and are therefore more informative for learning. A variety of such criteria based on ensemble learning are discussed in [394].

Details of other relevant methods for active learning are discussed in [11, 486].

## 7.7 Supervised Models for Unsupervised Outlier Detection

---

In section 1.2.1 of Chapter 1, it is pointed out that outlier detection models can be viewed as one-class avatars of classification models. In this section, we point out yet another connection to regression modeling. This particular connection has significant *practical* applicability because it enables the use of off-the-shelf classification and regression models for outlier detection.

Section 3.2.1 of Chapter 3 shows how one can use linear regression models to predict errors in a particular attribute in the data from the other attributes. Such models are very useful in contextual settings like time-series data where there is a clear distinction between contextual and behavioral attributes. However, it turns out that these models can also be used for unsupervised anomaly detection in multidimensional data. This broad idea has been explored recently [417, 429] in several anomaly detectors, although we focus on [429] because of its recency, extensive experimentation and clear explanations.

The basic idea [429] is to repeatedly apply a regression model  $\mathcal{M}_k$  by selecting the  $k$ th attribute as the dependent variable and the remaining attributes as the independent variables. Cross-validation is used [33] to ensure that each data point receives an out-of-sample score. The squared-error in prediction of the dependent variable is used to compute the score for the instance for that iteration. This approach is repeated with each attribute as the dependent variable and a weighted average is used to construct the final score. The weight is defined by the ability of the regression model to predict that attribute; the weight ensures that irrelevant attributes like identifiers are not used. It is noteworthy that one is not restricted to using the linear models of section 3.2.1; in fact, *virtually any of the hundreds of off-the-shelf regression models may be used for prediction*. This makes the approach almost trivial to implement, and numerous options are available for the regression subroutine. The approach is referred to as *attribute-wise learning for scoring outliers (ALSO)*.

Let  $\epsilon_k^2(\bar{X})$  be the squared error of data point  $\bar{X}$  using the model  $\mathcal{M}_k$  in which the  $k$ th attribute is used as the dependent variable. Note that cross-validation is used to compute each  $\epsilon_k^2(\bar{X})$  so that each point receives an out-of-sample score. It is important to standardize the data to unit variance as a preprocessing step in order to avoid problems associated with relative scaling of various attributes. Standardization ensures that the differences in  $\epsilon_k^2(\bar{X})$  across different attributes are not regulated by the scale of the attributes.

It now remains to describe how one might choose the weight of each component of the regression model. The basic idea is to ensure that the components of the model corresponding to irrelevant dependent attributes, which cannot be predicted easily by other attributes, are not weighted significantly. How does one determine such irrelevant attributes? Consider a trivial regression model  $\mathcal{T}_k$  that always predicts the mean of the  $k$ th dependent attribute for every data point irrespective of the values of the independent attributes. The root-mean-squared error of the  $k$ th such trivial model is always 1 because each attribute has been scaled to unit variance. Let  $RMSE(\mathcal{M}_k)$  represent the root-mean-squared error of the  $k$ th attribute:

$$RMSE(\mathcal{M}_k) = \sqrt{\frac{\sum_{i=1}^N \epsilon_k^2(\bar{X}_i)}{N}} \quad (7.4)$$

Then, the weight  $w_k$  of the model with the  $k$ th attribute as the dependent variable is given



by the following:

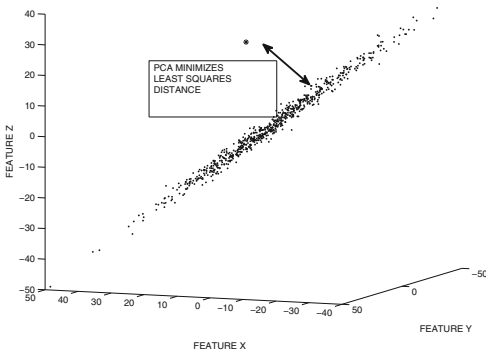
$$w_k = 1 - \min\{1, RMSE(\mathcal{M}_k)\} \quad (7.5)$$

The basic idea is that if the prediction performs worse than predicting the mean, the attribute is irrelevant and one should give the model a weight of 0. The maximum weight of the model can be 1 when it gives zero error. Then, the outlier score of the  $d$ -dimensional data point  $\bar{X}$  is as follows:

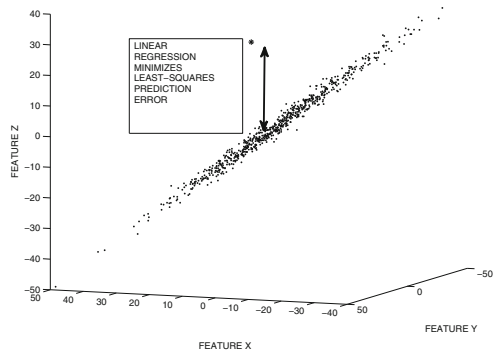
$$\text{Score}(\bar{X}) = \sum_{k=1}^d w_k \epsilon_k^2(\bar{X}) \quad (7.6)$$

Larger values of the score are more indicative of outlierness. Note that the score presented here is a simplified version of the one in [429], although it is equivalent because of the relative nature of outlier scores. It is noteworthy that the approach in [429] is able to achieve high-quality results with only a limited number of relatively simple base models, and it is possible that better results may be obtained with improved regression models. In particular, even though it has been shown that the use of M5 regression trees [453] as the base model provides excellent results, the use of its most obvious enhancement in the form of random forests for regression has not been tested. Since random forests are known to be very robust compared to most other classification and regression models in a wide variety of settings [195], it might be worthwhile to test the effectiveness of the approach with the use of random forests as the base regressor. One can also adjust the classifier or regressor depending on the domain that is tested (e.g., text versus images).

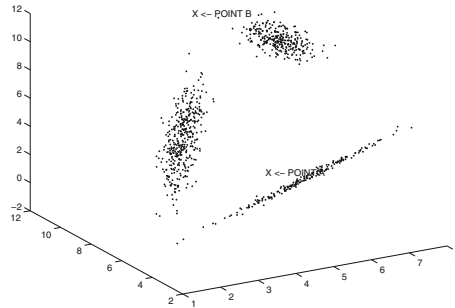
This approach also has natural connections with subspace outlier detection [4] because it provides an explanation of the outlierness of each data point in terms of *locally relevant* subspaces when certain types of base predictors are used. In this context, random forests provide the best interpretability in terms of subspace outlier detection. For example, if a random forest is used as the base learner and the attribute  $k$  has a large value of  $w_k \epsilon_k^2(\bar{X})$ , then the most erroneous paths in the underlying decision trees of  $\mathcal{M}_k$  provide insights about locally relevant subspaces. This is somewhat easier to understand when the target attribute is symbolic, or a numeric target attribute is discretized into a symbolic value. For example, consider a setting in which the target attribute is a binary disease attribute *Alzheimer* for which the value is drawn from  $\{0, 1\}$ . This is a disease that predominantly affects very old individuals except for a few rare (i.e., outlier) cases. Although the disease is primarily genetic, some factors like head-trauma further increase the risk. Therefore, a sequence of splits such as  $Age \leq 30, Trauma = 0$  will typically predict *Alzheimer* = 0. However, some test instances satisfying this sequence of splits might have *Alzheimer* = 1, which suggests that these are outliers. In such cases, one has an immediate understanding of the causality of this outlier. Furthermore, the sequence (corresponding to tree path) can be concatenated with the target attribute value (or its numeric range) to infer a locally relevant subspace for the specific test point. In this particular example, the locally relevant subspace is  $Age \leq 30, Trauma = 0, Alzheimer = 1$ . Furthermore, as is desirable [31] for subspace outlier detection, more than one locally relevant subspace is implicitly used by the approach to score points (and interpret them). This is because we can repeat this process with all the random forests  $\mathcal{M}_k$  and each random forest contains multiple decision trees. This is yet another reason why random forests should be used as base learners in this setting; they are likely to be robust because of their implicit use of multiple locally relevant subspaces.



(a) PCA-based error (Chapter 3)



(b) Linear regression-based error



(c) Cannot be modeled by PCA or linear regression

Figure 7.3: Showing connections between regression methods and PCA. The use of linear regression as a base method for the approach in this section results in intuitively similar models to the PCA method of Chapter 3. However, supervised modeling also allows the use of complex base methods like random forests. Such methods can handle complex data distributions like (c).

### 7.7.1 Connections with PCA-Based Methods

Chapter 3 discusses dependent variable regression in section 3.2.1 and independent variable regression (PCA) in section 3.2.2. A soft version of the latter is the Mahalanobis method (section 3.3.1). The approach in the aforementioned section is an instantiation of dependent-variable regression, especially when linear regression is used as the subroutine for each of the decomposed subproblems. Furthermore, the error (outlier score) with the use of PCA will be highly correlated with the aggregate regression errors using this approach when linear regression is used in the latter case. For example, we have shown PCA-wise regression errors in Figure 7.3(a) and an attribute-wise (linear) regression error in Figure 7.3(b). These errors are shown with double-sided arrows for the same outlier in Figures 7.3(a) and (b), respectively, and they are clearly correlated. However, there are cases that cannot be handled so easily by either method.

For example, the distribution in Figure 7.3(c) cannot be modeled by either PCA or dependent-variable modeling because outlier ‘A’ is exposed in a different rotated subspace than outlier ‘B.’ However, if a random forest is used as the base regression method, then such outliers will be exposed as well because of the ability of random forests to model such complex nonlinear distributions. The decomposition of a single unsupervised problem into many supervised problems also has the advantage of allowing the most relevant type of feature selection in each subproblem. In principle, such outliers can also be exposed with nonlinear PCA (section 3.3.8), when a spectral kernel is used. Therefore, there are deep connections between the Mahalanobis method and the approach discussed in this section. The approach discussed in this section is potentially more powerful because the use of certain base regressors like random forests provides a complex model with embedded feature selection. On the other hand, soft PCA allows better score aggregation along independent component directions. In general, their relative performance might depend on the complexity of the underlying data set at hand.

One can also show that the Mahalanobis method is a variation of this approach in another way by modifying the weighting function of Equation 7.5 as follows:

$$w_k = \frac{1}{[RMSE(\mathcal{M}_k)]^2} \quad (7.7)$$

Consider a setting in which we transformed the data into a new  $d$ -dimensional PCA-based axis system in which the various dimensions are uncorrelated. If we apply (supervised) least-squares regression to this transformed data by using the aforementioned partitioning scheme into independent and dependent variables, the regression model would not be (theoretically) expected to learn anything more than predicting the mean of the dependent variable. This is because we have already removed all the inter-attribute correlations by transformation, and the transformed variables do not have predictive power with respect to each other with the use of a linear regression model. In such a case, the best that one can theoretically expect<sup>3</sup> from the regression model is to report the mean along each transformed dimension as the prediction for each point. Therefore, the weight  $w_k$  in Equation 7.7 will simply be the inverse of the variance of the attribute. In other words, if  $\epsilon_i(\bar{X})$  is the error along the  $i$ th transformed dimension of data point  $\bar{X}$ , then the score according to Equation 7.6 (using

---

<sup>3</sup>For a PCA-transformed data set  $D$ , which is also mean-centered, the coefficients of linear regression can be computed as  $(D^T D)^{-1} D^T \bar{y}$ , where  $\bar{y}$  is the dependent variable (section 3.2.1). We do not need to account for the bias term by adding an additional column of 1s to  $D$  because of mean-centering. However,  $D^T \bar{y}$  is a vector of 0s because of the orthogonality property of SVD embeddings. Therefore, all coefficients will be 0, and one will always predict the dependent variable to be 0 (i.e., the mean).

the modified weight of Equation 7.7) is as follows:

$$\text{Score}(\bar{X}) = \sum_{k=1}^d \frac{\epsilon_k^2(\bar{X})}{\text{Variance of } k\text{th attribute}} \quad (7.8)$$

This is exactly the scoring function used by the Mahalanobis method (cf. Equation 3.17 in Chapter 3). One advantage of using this weight is that the outlier score can be modeled from a  $\chi^2$  distribution (see Chapter 3).

Although the Mahalanobis method has an advantage in some settings, it needs to be kept in mind that the supervised approach can discover outliers with complex base learners like random forests. In such cases, the underlying classifier can leverage the use of higher-order correlations among the dimensions, even after the first-order correlations have been removed. On the other hand, poor predictive power with respect to too many weakly relevant dimensions can continue to cause challenges in *ALSO*, no matter how one might adjust the scoring mechanism to account for irrelevant dimensions. This can be a particularly major problem in very high-dimensional domains like text in which it is almost impossible to predict the frequency of any particular word from other words even though there might be significant latent correlations in the aggregate. The second problem with the approach is that if two dimensions are perfectly correlated (e.g., Celsius and Fahrenheit readings) and a data point shows extreme values respecting that correlation, the approach will not detect the point as an outlier.

## 7.7.2 Group-wise Predictions for High-Dimensional Data

Many of the aforementioned challenges can be addressed using predictions on groups of sampled targets [35]. Both the Mahalanobis method and supervised learning can be viewed as special cases of this approach.

The approach is inherently ensemble-centric in nature. In each base detector, the first step is to randomly partition the attributes into  $r$  target variables and  $(d - r)$  predictor variables. However, the  $r$  target variables, denoted by  $S_r$ , are not used in their original form but are transformed to latent concepts to sharpen their predictability. The  $r$ -dimensional target variables are transformed to uncorrelated targets using PCA and then standardized to zero mean and unit variance. Transformation might occasionally cause target attributes with zero variance (eigenvalues) when the value of  $r$  is of the same magnitude as the number of data points. Transformed dimensions with zero variance are dropped, as a result of which one might be left with only  $r' \leq r$  targets. The  $(d - r)$  predictor variables are used to predict each of the  $r'$  transformed attributes using  $r'$  individually trained models for the various target attributes. One can use<sup>4</sup> cross-validation in order to ensure out-of-sample scores. Subsequently, the scores  $\epsilon_1(\bar{X}) \dots \epsilon_{r'}(\bar{X})$  over these  $r'$  targets are averaged. The resulting score, which is specific to the target set  $S_r$ , is as follows:

$$\text{Score}(S_r, \bar{X}) = \frac{\sum_{k=1}^{r'} w_k \epsilon_k^2(\bar{X})}{r'} \quad (7.9)$$

Here, the weight  $w_k$  is set according to Equation 7.7. This concludes the description of a single base detector for the sampled target attribute set  $S_r$ . Note the presence of  $S_r$  as

<sup>4</sup>For larger data sets, one can also subsample a small part of the data as the training portion but score all points with the model. This is an efficient approximation which is natural to use in an ensemble-centric setting [35].

an argument of the score in Equation 7.9. The approach is repeated over several samples of  $(d - r)$  predictors and  $r$  targets. The resulting scores for each data point are averaged in order to compute its final score. For very high-dimensional domains like text, it makes more sense to predict values along each of the  $r'$ -dimensional latent vectors rather than the individual word frequencies. The approach is referred to as *GASP* (Group-wise Attribute Selection and Prediction) [35].

Both *ALSO* and the Mahalanobis method are special cases of the *GASP* technique. Setting  $r = 1$  yields the *ALSO* technique (with minor scoring modifications), whereas setting  $r = d$  yields the Mahalanobis method, assuming that each attribute is predicted as its mean in the absence of predictor attributes. Note that the latter can also find outliers caused by correlated extreme values. As the dimensionality of the data increases, it often makes sense to increase the raw value of  $r$ , albeit sublinearly with increasing data dimensionality (e.g.,  $r = \lfloor \sqrt{d} \rfloor$ ). The work in [35] presents experimental results for the case in which  $r$  is set to  $d/2$  along with an ensemble-centric variation.

### 7.7.3 Applicability to Mixed-Attribute Data Sets

One of the nice characteristics of this approach is that it is directly generalizable to mixed-attribute data sets with minor modifications. Most regression models can be applied to mixed-attribute settings very easily. Furthermore, in cases in which the predictive attribute is categorical, one can use a classifier rather than a regression model. In such cases, the value of  $\epsilon_k(\bar{X})$  is drawn from  $\{0, 1\}$ . All other steps of the approach, including the final computation of the score, remain the same.

### 7.7.4 Incorporating Column-wise Knowledge

The supervised approach is particularly useful in settings where we are looking for specific types of anomalies and it is known that certain attributes are more important than others for anomaly detection. For example, consider an unsupervised setting in which we are predicting fraudulent transactions. Even though some attributes such as age and gender have marginal predictive value, other attributes such as the size of the transaction are far more important. In such cases, it is possible to restrict the predictive process only over the relevant subset of attributes as the target, although all attributes might be used for predicting the target attribute. Therefore, the score of Equation 7.6 is aggregated only over the relevant subset of target attributes. In a sense, one is able to properly distinguish between primary and secondary attributes with this process. This approach is also used for outlier detection in contextual data types such as time-series data in which the prediction of behavioral attributes (e.g., temperature) is more important than that of the contextual attributes (e.g., time). Such relationships between prediction and outlier detection will be discussed in greater detail in Chapter 9 on time-series data.

### 7.7.5 Other Classification Methods with Synthetic Outliers

A related method for using supervised methods is to generate synthetic data for the outlier class, and treat the provided data set as the normal class. This creates a supervised setting because one can now use any off-the-shelf classifier for the two-class problem. As discussed in [491], the outliers are generated uniformly at random between the minimum and maximum range of each attribute. Subsequently, any binary classification model may be applied to this data set. This method is, however, not quite as desirable because it might introduce

bias in the form of synthetic data. The “typical” pattern of outliers in any given data set is far from random. This is because outliers are often caused by specific types of anomalies in the underlying system, and they are often clustered into small groups. Therefore, the use of randomly distributed data as outliers can result in poor performance for certain types of data sets.

## 7.8 Conclusions and Summary

---

This chapter discusses the problem of supervised outlier analysis. In many real scenarios, training data are available, which can be used in order to greatly enhance the effectiveness of the outlier detection process. Many off-the-shelf classification algorithms can be adapted to this problem with minor modifications. The most common modifications are to address class imbalance with sampling and re-weighting.

The partially supervised variations of the problem are diverse. Some of these methods do not provide any labels on the normal class. This corresponds to the fact that the normal class may be contaminated with an unknown number of outlier examples. In other cases, no examples of the normal class may be available. Another form of partial supervision is the identification of novel classes in the training data. Novel classes correspond to scenarios in which the labels for some of the classes are completely missing from the training data. In such cases, a combination of unsupervised and supervised methods need to be used for the detection process. In cases where examples of a single normal class are available, the scenario becomes almost equivalent to the unsupervised version of the problem.

In active learning methods, human experts may intervene in order to add more knowledge to the outlier detection process. Such combinations of automated filtering with human interaction can provide insightful results. The use of human intervention sometimes provides more insightful results, because the human is involved in the entire process of label acquisition and final outlier detection.

Supervised methods can also be used for unsupervised anomaly detection by decomposing the anomaly detection problem into a set of regression modeling problems. This is achieved by setting each dimension in the data as a predicted variable and the other dimensions as the predictor variables. A weighted average of prediction errors over these different models is used to quantify the outlier score.

## 7.9 Bibliographic Survey

---

Supervision can be incorporated in a variety of ways, starting from partial supervision to complete supervision. In the case of complete supervision, the main challenges arise in the context of class imbalance and cost-sensitive learning [132, 135, 182]. The issue of evaluation is critical in cost-sensitive learning because of the inability to model the effectiveness with measures such as the absolute accuracy. Methods for interpreting receiver operating characteristic (ROC) curves and classification accuracy in the presence of costs and class imbalance are discussed in [174, 192, 302, 450, 451]. The impact of class imbalance is relevant even for feature selection [399, 616], because it is more desirable to select features that are more indicative of the rare class.

A variety of general methods have been proposed for cost-sensitive learning such as *MetaCost* [175], weighting [600], and sampling [124, 132, 173, 332, 600]. Weighting methods are generally quite effective, but may sometimes be unnecessarily inefficient, when most of the training data corresponds to the background distribution. In this context, sampling

methods can significantly improve the efficiency. Furthermore, they can often improve effectiveness because of their ability to use subsampled ensembles [32]. Numerous cost-sensitive variations of different classifiers have been proposed along the lines of weighting, including the Bayes classifier [600], nearest-neighbor classifier [609], decision trees [546, 566], rule-based classifiers [298, 300] and SVM classifiers [535, 570].

Ensemble methods for improving the robustness of sampling are proposed in [124, 371]. Since the under-sampling process reduces the number of negative examples, it is natural to use an ensemble of classifiers that combine the results from training on different samples. This provides more robust results and ameliorates the instability which arises from under-sampling. The major problem in over-sampling of the minority class is the over-fitting obtained by re-sampling duplicate instances. Therefore, a method known as *SMOTE* creates synthetic data instances in the neighborhood of the rare instances [133].

The earliest work on boosting rare classes was proposed in [305]. This technique is designed for imbalanced data sets, and the intuition is to boost the positive training instances (rare classes) faster than the negatives. Thus, it increases the weight of false negatives more the false positives. However, it is not cost-sensitive, and it also decreases the weight of true positives more than true negatives, which is not desirable. The *AdaCost* algorithm proposed in this chapter was proposed in [191]. Boosting techniques can also be combined with sampling methods, as in the case of the *SMOTEBoost* algorithm [134]. An evaluation of boosting algorithms for rare-class detection is provided in [299]. Two new algorithms for boosting are also proposed in the same paper. The effect of the base learner on the final results of the boosting algorithm are investigated in [301]. It has been shown that the final result from the boosted algorithm is highly dependent on the quality of the base learner.

A commonly encountered case is one in which the instances of the positive (anomalous) class are specified, whereas the other class is unlabeled [183, 356, 348, 362, 363, 595, 603]. Since the unlabeled class is pre-dominantly a negative class with contaminants, it is essentially equivalent to a fully supervised problem with some quantifiable loss in accuracy [183]. In some cases, when the collection mechanisms for the negative class are not reflective of what would be encountered in test instances, the use of such instances may harm the performance of the classifier. In such cases, it may be desirable to discard the negative class entirely and treat the problem as a one-class problem [356]. However, as long as the training and test distributions are not too different, it is generally desirable to also use the instances from the negative class.

The one-class version of the problem is an extreme variation in which only positive (anomalous) instances of the class are used for training purposes. SVM methods are particularly popular for one-class classification [303, 384, 460, 479, 538]. However, most of these methods are designed for the case in which the observed single class corresponds to normal points. It is also possible to adapt these methods to cases in which the observed class corresponds to anomalous points. Methods for one-class SVM methods for scene classification are proposed in [580]. It has been shown that the SVM method is particularly sensitive to the data set used [460].

An important class of semi-supervised algorithms is known as *novelty detection*, in which no training data is available about *some* of the anomalous classes. The special case of this setting is the traditional outlier detection problem in which no training data is available about *any* of the anomalous classes. This is common in many scenarios such as intrusion detection, in which the patterns in the data may change over time and may therefore lead to novel anomalies (or intrusions). These problems exist in the combination of the supervised and unsupervised case, and numerous methods have been designed for the streaming scenario [391, 392, 46]. The special case, where only the normal class is available is not very

different from the unsupervised scenario. Numerous methods have been designed for this case such as single-class SVMs [480, 113], minimax probability machines [336], kernel-based PCA methods [462], direct density ratio estimation [262], and extreme value analysis [270]. Single class novelty detection has also been studied extensively in the context of the first story detection in text streams [622] and will be discussed in detail in Chapter 8. The methods for the text-streaming scenario use standard clustering or nearest neighbor models. In fact, a variety of stream-clustering methods [28, 29] discover newly forming clusters (or emerging novelties) as part of their output of the overall clustering process. A detailed survey of novelty detection methods may be found in [388, 389]. Unsupervised feature engineering methods for improving the accuracy of rare-class detection are discussed in [395].

Human supervision is a natural goal in anomaly detection, since most of the anomalous instances are not interesting, and it is only by incorporating user feedback that the interesting examples can be separated from noisy anomalies. Methods for augmenting user-specified examples with automated methods are discussed in [618, 619]. These methods also add artificially generated examples to the training data to increase the number of positive examples for the learning process. Other methods are designed for *selectively* presenting examples to a user, so that only the relevant ones are labeled [431]. A nearest-neighbor method for active learning is proposed in [252]. The effectiveness of active learning methods for selecting good examples to present to the user is critical in ensuring minimal human effort. Such points should lie on the decision boundaries separating two classes [149]. Methods that use query-by-committee to select such points with ensembles are discussed in [394, 485]. A recent approach that minimizes the amount of required data for supervised active learning is discussed in [224]. Surveys on supervised active learning may be found in [18, 486].

A selective sampling method that uses active learning in the context of outlier detection is proposed in [1]. A method has also been proposed in [366] as to how unsupervised outlier detection algorithms can be leveraged in conjunction with limited human effort in order to create a labeled training data set. The decomposition of unsupervised outlier detection into a set of supervised learning problems is discussed in [35, 417, 429]. In particular, the work in [35] discusses a number of ensemble-centric variants of the decomposition process.

## 7.10 Exercises

---

1. Download the *Arrhythmia* data set from the *UCI Machine Learning Repository*.
  - Implement a 20-nearest neighbor classifier which classifies the majority class as the primary label. Use a 3 : 1 ratio of costs between the normal class, and any other minority cost. Determine the overall accuracy and the cost-weighted accuracy.
  - Implement the same algorithm as above, except that each data point is given a weight, which is proportional to its cost. Determine the overall accuracy and the cost-weighted accuracy.
2. Repeat the exercise above for the quantitative attributes of the *KDD CUP 1999 Network Intrusion* data set of the *UCI Machine Learning Repository*.
3. Repeat each of the exercises above with the use of the *MetaCost* classifier, in which 100 different bagged executions are utilized in order to estimate the probability of relabeling. An unweighted 10-nearest neighbor classifier is used as the base learner.



For each bagged execution, use a 50% sample of the data set. Determine the overall accuracy and the cost-weighted accuracy.

4. Repeat Exercises 1 and 2 by sampling one-thirds the examples from the normal class, and including all examples from the other classes. An unweighted 20-nearest neighbor classifier is used as the base learner. Determine the overall accuracy and the cost-weighted accuracy.
5. Repeat Exercise 4, by using an ensemble of five classifiers, and using the majority vote.
6. Repeat Exercises 1 and 2 with the use of cost-sensitive boosting. An unweighted 10-nearest neighbor classifier is used as the base learner.

---

---

## Chapter 8

# Outlier Detection in Categorical, Text, and Mixed Attribute Data

---

---

“We become not a melting pot, but a mosaic. Different people, different beliefs, different yearnings, different hopes, different dreams.”—Jimmy Carter

### 8.1 Introduction

---

The discussion in the previous chapters has primarily focused on numerical data. However, the setting of numerical data represents a gross oversimplification because categorical attributes are ubiquitous in real-world data. For example, although demographic data may contain quantitative attributes such as the age, most other attributes such as gender, race, and ZIP code are categorical. Data collected from surveys may often contain responses to multiple-choice questions that are categorical. Similarly, many types of data such as the names of people and entities, IP-addresses, and URLs are inherently categorical. In many cases, categorical and numeric attributes are found in the same data set. Such *mixed-attribute data* are often challenging to machine-learning applications because of the difficulties in treating the various types of attributes in a homogeneous and consistent way.

Categorical attributes are also referred to as *discrete* attributes. An important characteristic of such attributes is that the underlying values are inherently unordered. Unlike numerical data, it is often hard to define similarity between different values of the same attribute. For example, if the attribute *Color* takes on the values “Red,” “Blue,” and “Orange,” the semantic distance between “Red” and “Orange” might be less than that between “Red” and “Blue,” because the former pair of colors look similar. This situation is common in real-world settings, but it is often not always straightforward to infer such relationships in a data-driven way. These nuances of categorical data create numerous methodological and technical challenges for developing outlier detection algorithms, which are as follows:

- Extreme-value analysis and statistical algorithms are dependent on the use of statistical quantifications such as the mean and standard deviation of numerical values. In

the case of categorical data, such statistical quantifications are no longer meaningful.

- Linear models such as principal component methods (PCA) are designed for numerical data. Although the data can be preprocessed into sparse binary representations, the number of entries in the data matrix might be too large and most of the entries might be 0s. This creates computational challenges.
- All proximity-based algorithms are dependent on some notion of distance. However, many proximity quantifications such as the Euclidean distance function are not meaningful or effective for categorical data. Therefore, proximity functions need to be re-defined for categorical data in order to enable the use of proximity-based algorithms.
- Many density-based models such as the Parzen-Rosenblatt method and volume-based methods (e.g., LOCI) cannot be easily adapted to categorical data because these methods are implicitly dependent on the notion of distances.

However, all classes of algorithms can be adapted for categorical data with suitable modifications. Because of the importance of similarity computations in outlier detection, this chapter will also review similarity measures for categorical data.

An important observation about categorical data is that it can be transformed into binary data by treating each value of the categorical attribute as a binary attribute. The binary attribute corresponding to the relevant categorical attribute value takes on the value of 1, whereas other binary attributes take on the value of 0. This allows the use of many algorithms that are designed for other sparse (but well-studied) domains such as text data. Furthermore, since binary data is a (rudimentary) special case of numerical data, the existing algorithms for numerical data can also be applied to this case. However, such an approach does not scale well with an increasing number of possible *values* of the categorical attribute. This is because a separate binary field needs to be dedicated to each *distinct* value of the categorical attribute. For an attribute such as the ZIP code, the number of possible values may be on the order of thousands, and therefore thousands of binary attributes may need to be created from a single categorical attribute. Furthermore, since the values of these binary attributes are heavily dependent on one another, it is inefficient to use the binary representation.

For ease in further discussion, the notations for categorical data are introduced here. It is assumed that the  $d$ -dimensional data set  $\mathcal{D}$  contains  $N$  records. When the data set is purely categorical, it is assumed that the  $i$ th attribute contains  $n_i$  possible distinct values. On the other hand, for a mixed-attribute data set, the first  $d_c$  attributes are assumed to be categorical, and the remaining are assumed to be numerical.

This chapter is organized as follows. Section 8.2 discusses the generalization of probabilistic models to categorical data. Section 8.3 discusses the extension of linear models to categorical data. The extension of proximity-based models to categorical data is studied in section 8.4. The special case of categorical data, corresponding to binary and transaction data, is discussed in section 8.5. The case of text data is studied in section 8.6. Section 8.7 presents the conclusions and summary.

## 8.2 Extending Probabilistic Models to Categorical Data

---

Like numeric data, probabilistic outlier detection models for categorical data [578] use generative models. The main difference is that the generating distribution is tailored to cate-

gical records rather than numerical records. Generative models have the advantage that the complexity of the specific data domain is captured with the choice of the underlying distribution and a specific generative process. Once this generative process has been defined, one can learn the parameters of the corresponding data distribution in a data-driven manner. A specific example of this generative process for numerical data is discussed in section 2.4 of Chapter 2. A *mixture model* is introduced in section 2.4 in which each data point is generated from one of the components of a mixture of Gaussian clusters. The mean and covariance matrix of each Gaussian cluster are then estimated in a data-driven manner.

The case of categorical data is not very different from that of numerical data. Just as a mixture model is defined for numerical data in Chapter 2, we can define a corresponding mixture model for categorical data. The *only* difference is that the probability distribution of each component of this mixture model is now defined for categorical data rather than the Gaussians, which are suited to numerical data. This difference primarily impacts the methodology and specific details of parameter estimation. Nevertheless, the overall framework is almost identical in both cases. Furthermore, such an approach can be easily extended to mixed-attribute data by defining separate probability distributions for the categorical and numerical attributes and combining them in product-wise fashion to create a single multivariate distribution.

Next, we will describe the generative process for categorical data. It is assumed that the mixture contains  $k$  components denoted by  $\mathcal{G}_1 \dots \mathcal{G}_k$ . Each point in the observed data set is generated from one of  $k$  components using the following process:

- Select the  $m$ th mixture component with probability  $\alpha_m$  for  $m \in \{1 \dots k\}$ .
- Generate a data point from  $\mathcal{G}_m$ .

The observed data points that have low probabilities of being generated by this process (or low *fit* probabilities) are deemed to be outliers.

The values of  $\alpha_m$  for  $m \in \{1 \dots k\}$  denote the prior probabilities, and may either be pre-specified to equal values of  $1/k$  as a simplification or may be learned in a data-driven manner. Furthermore, the parameters of the distributions of the mixture components need to be learned so as to maximize the log-likelihood of the observed data being generated by the model. Note that the generative process is similar to that discussed in section 2.4 of Chapter 2. The main difference is in the mathematical form of the generative model for the  $m$ th cluster (or mixture component)  $\mathcal{G}_m$  because these distributions now need to generate categorical points rather than numerical points. A common example of such a distribution is the Bernoulli distribution.

In the *generalized Bernoulli*<sup>1</sup> model, it is assumed that the  $j$ th possible categorical value of the  $i$ th attribute is generated by cluster  $m$  with probability  $p_{ijm}$ . The set of all model parameters is (collectively) denoted by the notation  $\Theta$ .

The expectation-maximization algorithm alternately estimates the generative (assignment) probabilities of points from mixture components (clusters), and also estimates the distribution parameters while fixing assignment probabilities. Consider a  $d$ -dimensional data point  $\bar{X}$  in which the  $r$ th attribute takes on the  $j_r$ th possible categorical value of this attribute. Then, the value of the generative probability  $g^{m,\Theta}(\bar{X})$  of  $\bar{X}$  from the  $m$ th mixture

---

<sup>1</sup>Strictly speaking, Bernoulli models are designed for the case of binary data containing only two possible values for each attribute. The generalized version of the Bernoulli distribution allows more than two outcomes for the random variable. This is particularly well-suited to categorical data with many distinct values.

component is given by the following expression:

$$g^{m,\Theta}(\bar{X}) = \prod_{r=1}^d p_{rj_r,m} \quad (8.1)$$

Note that the value of  $g^{m,\Theta}(\cdot)$  is a discrete probability, unlike the continuous density function  $f^{m,\Theta}(\cdot)$  discussed in Chapter 2. Nevertheless, this is the analogous probabilistic quantification for the cluster-specific fit probability. Correspondingly, the posterior probability  $P(\mathcal{G}_m|\bar{X},\Theta)$ , that the data point  $\bar{X}$  is generated by the  $m$ th mixture component (cluster) may be estimated as follows:

$$P(\mathcal{G}_m|\bar{X},\Theta) = \frac{\alpha_m \cdot g^{m,\Theta}(\bar{X})}{\sum_{r=1}^k \alpha_r \cdot g^{r,\Theta}(\bar{X})} \quad (8.2)$$

This defines the E-step for the categorical scenario. Note that this step provides a soft assignment probability of the data points to the different clusters.

Once the soft-assignment probability is estimated, the probability  $p_{ijm}$  is estimated to maximize the log-likelihood fit while treating the assignment probability as fixed. In this case, the maximization of the log-likelihood fit takes on a particularly simple form. While estimating the parameters for cluster  $m$ , the *weight* of a record is assumed to be equal to its generative (assignment) probability  $P(\mathcal{G}_m|\bar{X},\Theta)$  from mixture component (to cluster)  $m$ . The value  $\alpha_m$  is estimated in exactly the same way as discussed in Chapter 2. Specifically,  $\alpha_m$  is the weighted fraction of records assigned to cluster  $m$  on the basis of the soft probabilities computed in the E-step. In the event that the number of data points is small, we add a *Laplacian smoothing* parameter  $\gamma$  to the numerator and  $\gamma \cdot k$  to the denominator of this fraction for some small value of  $\gamma > 0$ . Let  $\mathcal{D}_{ij}$  be the subset of points in database  $\mathcal{D}$  for which the  $i$ th attribute takes on the  $j$ th value. For each cluster  $m$ , the aggregate *weight*  $w_{ijm}$  of the records for which attribute  $i$  takes on the  $j$ th value in cluster  $m$  is estimated as follows:

$$w_{ijm} = \sum_{\bar{X} \in \mathcal{D}_{ij}} P(\mathcal{G}_m|\bar{X},\Theta) \quad (8.3)$$

Then, the value of the probability  $p_{ijm}$  may be estimated as ratio of this weight in  $\mathcal{D}_{ij}$  to that in the entire database:

$$p_{ijm} = \frac{w_{ijm}}{\sum_{\bar{X} \in \mathcal{D}} P(\mathcal{G}_m|\bar{X},\Theta)} = \frac{\sum_{\bar{X} \in \mathcal{D}_{ij}} P(\mathcal{G}_m|\bar{X},\Theta)}{\sum_{\bar{X} \in \mathcal{D}} P(\mathcal{G}_m|\bar{X},\Theta)} \quad (8.4)$$

In practice, sufficient data may often not be available to estimate the parameters robustly, especially when some of the attribute values may not appear in a cluster (or  $w_{ijm} \approx 0$ ). This can lead to poor parameter estimations. Laplacian smoothing is commonly used in order to address such ill-conditioned probabilities. Let  $v_i$  be the number of distinct attribute values for the  $i$ th attribute. One can perform Laplacian smoothing by adding a small positive value  $\beta$  to the numerator and a value  $v_i \cdot \beta$  to denominator of the aforementioned estimation. The effect of Laplacian smoothing is to bias the estimated probability  $p_{ijm}$  towards its prior value of  $1/v_i$ . This smoothing is especially noticeable in cases where sufficient data is not available to properly estimate  $p_{ijm}$ . The smoothed estimation is as follows:

$$p_{ijm} = \frac{\beta + w_{ijm}}{v_i \cdot \beta + \sum_{\bar{X} \in \mathcal{D}} P(\bar{X} \in \mathcal{G}_m|\Theta)} \quad (8.5)$$

This completes the description of the M-step. As in the case of numerical data, the E-step and M-steps are iterated to convergence. Outliers may be declared as data points which either have low assignment probability to their best matching cluster, or have low absolute fit to the generative model. The use of absolute fit as the outlier score is more common, and it may be computed as follows:

$$\text{Score}(\bar{X}) = \sum_{m=1}^k \alpha_m \cdot g^{m,\Theta}(\bar{X}) \quad (8.6)$$

Lower values of the score are more indicative of outlierness. Note that this equation can be easily computed, because all the required parameters on the right-hand side have already been estimated by the expectation-maximization method. If desired, extreme value analysis can be used on the outlier scores in order to identify the relevant outliers. However, for reasons discussed in section 6.2.1 of Chapter 6, extreme-value analysis should be applied on the logarithms of the scores rather than the scores themselves. Large negative values would represent outliers.

### 8.2.1 Modeling Mixed Data

Mixed data is particularly easy to address with probabilistic models in a seamless way because all cluster assignments are evaluated in terms of probabilities. One common problem with the use of mixed data in other models is that of effective *normalization*; it is often difficult to ensure that all attributes are provided equal importance by an unsupervised algorithm in the absence of prior knowledge about the relative importance of attributes. In probabilistic models, all normalization issues that might arise in the context of mixed data sets are automatically addressed because of the use of probabilities to describe the distributions of various types of attributes. This provides a more homogeneous treatment of mixed data sets.

The EM-algorithm is modified by defining the generative probability of each component as a composite function of the different kinds of attributes. For example, the continuous attributes are modeled with a *density function*  $f^{m,\Theta}(\bar{X})$ , whereas the categorical attributes are modeled with a discrete probability function  $g^{m,\Theta}(\bar{X})$ . In a scenario containing both categorical and mixed attributes, it is possible to define the following joint density function  $h^{m,\Theta}(\bar{X})$ , which is a product of these values:

$$h^{m,\Theta}(\bar{X}) = f^{m,\Theta}(\bar{X}) \cdot g^{m,\Theta}(\bar{X}) \quad (8.7)$$

The E-step can then be performed with the use of this joint density function  $h^{m,\Theta}(\bar{X})$ . The M-step is performed separately on the categorical and numerical attributes in exactly the same way as discussed in Chapter 2 (for numerical attributes) and the discussion above (for categorical attributes).

A specific method for outlier analysis with mixed attribute data sets was proposed by [578]. This technique is an online approach in which temporal discounting is used in order to provide more importance to recent records. It has some differences with the more generic description provided above. For example, the technique in [578] uses the Hellinger scores in order to compute the outlier scores rather than the absolute fit probabilities. Here, a simplified and more general description of probabilistic models is provided. The interested reader may refer to [578] for a description optimized to the online scenario.

## 8.3 Extending Linear Models to Categorical and Mixed Data

---

Linear models can be extended to categorical data sets by using the conversion from the categorical data set to the binary scenario. For each categorical attribute, a set of *one-hot* binary dimensions are created, in which only one value is allowed to one corresponding to the relevant attribute value. Since the  $i$ th attribute contains  $n_i$  possible values, this leads to a data set with dimensionality  $\sum_{i=1}^d n_i$ . This process may lead to a significant expansion of the dimensionality of the data, especially if the numbers of distinct values of many attributes are large.

One problem is that the different attributes may be assigned different importance by such a transformation in an implicit way. This is because the values of  $n_i$  vary widely across different attributes as a result of which different attributes are assigned different numbers of dimensions. Furthermore, the frequencies of various attribute values also play a role in their relative presence. For example, a column in which 50% of the binary attributes take on the value of 1, is very different from a column in which only 1% of the attributes take on the value of 1. In such cases, the column that has 50% relative presence is implicitly given greater weight because of the greater variance in its (binary) value. The data set can be normalized by dividing each column by its standard deviation. Consider the  $j$ th value of the transformed binary data for which a fraction  $f_{ij}$  of the entries take in the value of 1. In such a case, the standard deviation is given by  $\sqrt{f_{ij} \cdot (1 - f_{ij})}$ . Therefore, one possibility would be to divide that column of the binary data by  $\sqrt{f_{ij} \cdot (1 - f_{ij})}$ . As in the case of numerical data, this corresponds to a normalization in which the variance of each *derived* binary dimension is set to the same value of 1. Unfortunately, such a normalization will favor attributes with large values of  $n_i$ , since their cumulative variance in the principal component matrix will be  $n_i$ . In order to account for this, all columns for the  $i$ th attribute are divided by  $\sqrt{n_i \cdot f_{ij} \cdot (1 - f_{ij})}$ . This ensures that the sum of the variances for all the columns corresponding to a particular attribute is equal to 1. The Principal Component Analysis (PCA) method of Chapter 3 can be directly applied to this representation in order to determine the outlying points in the data.

Such an approach can also be used for mixed attribute data sets by adding normalized numerical attributes to this representation. The normalization process for the numerical columns is relatively straightforward. The columns are normalized, so that the variance of each column is one. This ensures that such methods can be used over a diverse data set with mixed attributes without giving too much importance to only one of the attributes. Principal component analysis methods are particularly effective for sparse binary data sets because they can represent the data in a very small number of components.

After the principal components have been estimated, a similar methodology to that discussed in Chapter 3 can be used for outlier modeling. The deviations along the principal components are computed, and the sum of the squares of these values is modeled as a  $\chi^2$  distribution with  $d$  degrees of freedom. The extreme-values among the different point-specific deviations are reported as the outliers.

### 8.3.1 Leveraging Supervised Regression Models

It is also possible to use dependent variable modeling (section 3.2.1 of Chapter 3) in which one attribute is fixed as the predicted attribute and other attributes are used to model it. The root-mean squared (RMSE) error of the modeling provides an attribute-specific outlier

score. This approach is repeated with each predicted attribute in turn, and the scores of each data point are averaged over different columnwise predictors. Such an approach is discussed in detail in section 7.7 of Chapter 7, and its extension to mixed-attribute data is discussed in section 7.7.3.

## 8.4 Extending Proximity Models to Categorical Data

The design of effective similarity measures for categorical data is critical for the effective implementation of proximity-based algorithms such as  $k$ -nearest neighbor methods. Although the design of similarity measures for categorical data is a vast research area in its own right, this chapter will introduce the more common similarity measures that are used for outlier detection. For the case of categorical data, it is generally more natural to study similarities rather than distances because many of the measures are naturally based on matching discrete values [93].

Consider two data points  $\bar{X} = (x_1 \dots x_d)$  and  $\bar{Y} = (y_1 \dots y_d)$ . Then, the similarity between the points  $\bar{X}$  and  $\bar{Y}$  is the sum of the similarities on the individual attribute values. In other words, if  $S(x_i, y_i)$  denotes the similarity between the attributes values  $x_i$  and  $y_i$ , then the overall similarity is defined as follows:

$$Sim(\bar{X}, \bar{Y}) = \sum_{i=1}^d S(x_i, y_i) \quad (8.8)$$

This similarity measure is therefore dependent on the specific instantiation of  $S(x_i, y_i)$  that is used in the aforementioned definition.

The simplest possible option is to fix  $S(x_i, y_i)$  to 1 when  $x_i = y_i$  and 0, otherwise. This instantiation of  $S(x_i, y_i)$  is referred to as the *overlap* measure. This measure is statistically naive but popular because of its simplicity. Its main problem is that it does not account for the relative frequencies of different attributes. For example, a match between two rare attribute values (e.g., two patients with cancer) is statistically more significant than a match between two frequent attributes (e.g., two normal patients). This type of effect is particularly important in the context of applications like outlier analysis where uneven frequency distribution of attribute values might be indicative of abnormal characteristics.

There are two primary classes of methods for measuring similarity in categorical data:

- The aggregate statistical properties of the data (i.e., statistical frequencies of the attributes) can be used to enable better similarity computations [93]. For example, matches between rare attribute values are considered more significant, whereas mismatches between rare attributes are considered less significant.
- The statistical neighborhoods of data points are used to compute similarity. This approach implicitly models the inter-attribute correlations in the neighborhood of a point for similarity computations. For example, if the colors “Red” and “Orange” co-occur more frequently in the neighborhood of a point than the colors “Red” and “Blue,” it means that “Red” and “Orange” are semantically more similar than “Red” and “Blue” in that neighborhood. Of course, since the definition of a neighborhood is itself based on similarity, there is an inherent circularity in this type of definition. A typical approach is to initialize the computation with a simple method such as the *overlap* measure, which is subsequently followed by iterative refinement. An example of such a measure is provided in [154]. This type of approach has the advantage that it can capture semantic relationships between related attribute values.



The second approach for similarity computation is clearly much richer because it captures latent relationships between attribute values. However, it is also computationally more intensive. This section will discuss both types of methods.

### 8.4.1 Aggregate Statistical Similarity

In the context of categorical data, the *aggregate statistical properties* of the data set should be used in computing similarity. For example, consider a case in which an attribute takes on the value of “Normal” 99% of the time, and the value of “Abnormal” 1% of the time. In such a case, the similarity between frequent attribute values should be given less weight than infrequent attribute values. This principle forms the basis of many well-known normalization techniques such as the *Inverse Document Frequency (IDF)* in the information retrieval domain. Mismatches between attribute values should also be weighted differently based on the frequencies of the underlying attribute values.

Dissimilarity between  $x_i$  and  $y_i$  is more likely when the number  $n_i$  of the distinct values of the  $i$ th attribute is large or when  $x_i$  and  $y_i$  are rare. Correspondingly, the *Eskin* measure modifies the overlap measure by using the same value of  $S(x_i, y_i) = 1$  when  $x_i = y_i$ , but also using a non-zero similarity value of  $n_i^2 / (n_i^2 + 2)$  when  $x_i \neq y_i$ . Instead of using the number of distinct values of the  $i$ th attribute, one could directly use the raw frequency of  $x_i$  and  $y_i$  in penalizing mismatches between rare values to a smaller degree. Such an alternative is the *Inverse Occurrence Frequency (IOF)* measure, which uses the same value of 1 for matching values and a value of  $1 / (1 + \log[f(x_i)] \cdot \log[f(y_i)])$ , otherwise. Here,  $f(x_i)$  and  $f(y_i)$  are the raw numbers of records taking on the values of  $x_i$  and  $y_i$ , respectively, for the  $i$ th attribute.

Matches between rare attribute values can be given greater weight. The inverse occurrence frequency discussed above uses the rarity only in mismatches but not in matches. It is also possible to use the inverse occurrence frequency in matches, which makes the computation similar to text data. Let  $p_i(x)$  be the fraction of records in which the  $i$ th attribute takes on the value of  $x$  in the data set. Thus, when  $x_i = y_i$ , the similarity is equal to  $[\log(1/p_i(x_i))]^2$  and 0, otherwise. This type of measure is commonly used in the information retrieval domain to compute similarities in text data [472].

Another similarity computation method that uses the principle of assigning greater weight to (matching) infrequent values is the *Goodall* measure. The original method proposed in [222] is slow. Therefore, the work in [93] uses a number of heuristic approximations that capture the spirit of this measure in more efficient ways. One variant of the *Goodall* measure uses  $1 - p_i(x_i)^2$  as the similarity on the  $i$ th attribute, when  $x_i = y_i$ , and 0, otherwise. This particular (efficient and simplified) variant of the *Goodall* measure is referred to as *Goodall3* [93]. In some variants of the measure, multivariate dependencies between different attribute values are utilized in the final computation.

An important observation is that many of the above measures are intuitively similar in the broad principle of weighting different attribute values differently. The major differences lie in how this broad principle is applied by selecting a specific function for weighting attribute frequencies. A number of such measures are proposed in [54, 109, 209, 358, 498] and summarized very well in [93]. The work in [93] performed an extensive evaluation of a variety of categorical similarity measures, many of which are based on aggregate statistics. This work is especially relevant because the evaluations were performed in the context of the outlier detection problem. The broader conclusion of this work is that the use of statistical (aggregation-based) measures definitely improves the quality of the similarity. However, the experiments in the same work showed that no single measure was dominant over all the other measures.

Like the numerical similarity measures discussed in section 4.3.3 of Chapter 4, these similarity measures are heavily *data-dependent*. In the case of categorical data, such similarity measures assume a special importance because of the difficulty in computing numerical distances between points. An even more strongly data-dependent measure is contextual similarity, which is discussed in the next section.

## 8.4.2 Contextual Similarity

Contextual similarity measures provide a fundamentally different point of view, in which the relationships between data points are used to define relationships between attribute values and vice versa. For example, consider a setting in which we have a *Color* attribute with values like “Red,” “Blue,” and “Orange.” If the data points containing “Red” and “Orange” are very similar to one another on the other attributes, as compared to the data points containing “Red” and “Blue,” it is an indication of the greater semantic similarity between the former pair of attribute values. Therefore, a mismatch between “Red” and “Blue” should not be treated similarly to a mismatch between “Red” and “Orange.” Note that this observation also applies to the matches between values *across* different attributes. For example, one could binarize the categorical attributes into a binary data set, and try to develop a more general model in which we account not for the matches between individual attributes but also to the matches across different attributes. The simplest approach for measuring contextual similarity is to use the random-forest/hierarchical-clustering method discussed in section 4.3.3. This approach can be easily adapted to categorical data, because many off-the-shelf implementations of random forests and hierarchical clustering are available. The following will introduce an alternative method [154] for binary data, which can also be generalized to categorical data by binarizing the data set.

Consider a binary transaction database from a supermarket in which the attributes are binary, and represent the buying behavior of items such as *Coke*, *Pepsi*, *Mustard*, and so on. In such cases, it is evident that two customers who buy *Coke* and *Pepsi*, respectively, are more likely to show similar buying patterns on the other attributes, than a customer who buys *Mustard*. The converse is also true, where similar pairs of customers (on other attributes) are more likely to buy *Coke* and *Pepsi* rather than *Coke* and *Mustard*. Therefore, it is evident that similarities between attributes can often be inferred from similarities between data points, and vice versa. It is noteworthy that there is an inherent circularity in this relationship. Such a circularity is usually resolved in machine learning algorithms with iterative methods.

The set of rows in which the attribute *Coke* takes on the value of 1 represents a *sub-relation* of the data. When two attributes have similar sub-relations in terms of the underlying patterns, the corresponding attributes are also similar. Therefore, the work in [154] uses an *Iterative Contextual Distance* algorithm, in which the distances are determined by repeating the following steps in circular fashion, after an initialization, which is based on simpler aggregate measures of similarity:

- **Computing distances between points from distances between attributes:** The distances between attributes are used in order to construct a real valued representation of the rows in the data. This real valued representation encodes all useful information about the inter-attribute correlations. Therefore distances between the rows in this real-valued representation automatically encode inter-attribute similarity. Although a number of simpler ways could be used to achieve this, the work in [154] uses a kernel-mapping approach in order to define the real-valued representations of

the rows. The details of this approach may be found in [154]. In order to encode the distances between different attribute values, this approach defines real values for each entry that correspond to the similarity of a particular attribute to the other attributes. The reader is referred to [154] for the specific details of this similarity computation.

- **Computing distances between attributes from distances between points:** The distances between attribute values is defined on the basis of distances between their sub-relations. First, the sub-relations are isolated for each attribute value (e.g., *Coke* customers, and *Pepsi* customers). The real-valued centroid of each of the kernel mapping of the rows in the sub-relations is determined. The  $L_1$  distance between the centroids is used in order to measure the distance between the corresponding attributes. Other more complex measures such as the probabilistic differences between the distributions of the row values could also be used in order to achieve this goal in a more sophisticated way.

These steps are repeated to convergence. Although this algorithm was originally presented in the context of binary (market-basket) attributes, it can easily be generalized to any type of categorical data by using a preprocessing step of binarization.

#### 8.4.2.1 Connections to Linear Models

This approach has a natural connection with the linear models discussed earlier in this chapter. In fact, linear models (in general) and matrix factorization methods (in particular) provide a natural way to compute similarities between rows and columns simultaneously. Let  $D$  be an  $N \times d$  matrix of binary transaction data. This matrix can be factorized into two low-rank matrices of dimensions  $N \times k$  and  $d \times k$ , where  $k$  is the rank of the factorization. We replicate Equation 3.28 of Chapter 3 here:

$$D \approx UV^T \tag{8.9}$$

This form of matrix factorization provides a different way of inferring similarities between rows and columns of  $D$  simultaneously with the use of latent representations. The similarities among the  $N$  rows of  $U$  provide the latent similarities among the rows of  $D$  (taking into account inter-attribute correlations), whereas the similarities among the  $d$  rows of  $V$  provide the latent similarities among the columns of  $D$  (taking into account inter-point similarities). This suggests that we can achieve a roughly similar goal by factorizing the matrix  $D$  into two low-rank factors and using these factors to compute similarities between points and attributes.

It is possible to use PCA for matrix factorization. However, transaction matrices are very large and sparse, which make them unsuitable for off-the-shelf PCA. This is because the dimensionality of such data sets may be of the order of millions, whereas only a small proportion (e.g., 0.01%) of the entries might be nonzero. One can use any form of constrained or unconstrained matrix factorization using only the non-zero entries in  $D$  and a sample of the zero entries. In other words, entry-wise sampling is used on the zero entries in order to make these methods scalable. Such an approach is more efficient, and it amounts to the application of matrix factorization on incomplete data (cf. section 3.5.1 of Chapter 3). Nonzero entries with large differences from their predicted values can be viewed as outlier entries in the matrix. As discussed in Chapter 3, one can consolidate the entry-wise scores into row-wise scores. Furthermore, ensemble methods with the use of multiple samples can be used to make these methods more efficient. If efficiency is not a concern, one can even use kernel PCA to infer nonlinear relationships among the rows and columns.

### 8.4.3 Issues with Mixed Data

It is fairly straightforward to generalize any proximity-based approach to the case of mixed data, as long as appropriate weights can be assigned to the categorical and numerical components. For example, let us consider two records  $\bar{X} = (\bar{X}_n, \bar{X}_c)$  and  $\bar{Y} = (\bar{Y}_n, \bar{Y}_c)$ , where  $\bar{X}_n, \bar{Y}_n$  are the subsets of numerical attributes and  $\bar{X}_c, \bar{Y}_c$  are the subsets of categorical attributes. Then, the overall similarity between  $\bar{X}$  and  $\bar{Y}$  is defined as follows:

$$Sim(\bar{X}, \bar{Y}) = \lambda \cdot NumSim(\bar{X}_n, \bar{Y}_n) + (1 - \lambda) \cdot CatSim(\bar{X}_c, \bar{Y}_c) \quad (8.10)$$

The parameter  $\lambda$  regulates the relative importance of the categorical and numerical attributes. The choice of  $\lambda$  can sometimes be a challenging task, especially since the similarities on the two domains may not be of the same scale. In the absence of prior or domain knowledge about the relative importance of attributes, a natural choice would be to use a value of  $\lambda$  that is equal to the fraction of numerical attributes in the data. Furthermore, the proximity in numerical data is often computed with the use of distance functions rather than similarity functions. However, distance values can be converted to similarity values using a method suggested in [93]. For a distance value of  $dist$ , the corresponding similarity value is denoted by  $1/(1+dist)$ . Alternatively, one might choose to use the *heat-kernel* value of  $e^{-dist^2/t^2}$ , where  $t$  is a user-defined parameter.

Further normalization is required to meaningfully compare the similarity value components on the numerical and categorical attributes, which may be in completely different scales. One way of achieving this goal would be to determine the standard deviations in the similarity values over the two domains with the use of sample pairs of records. Each component of the similarity value (numerical or categorical) is divided by its standard deviation. Therefore, if  $\sigma_c$  and  $\sigma_n$  be the standard deviations of the similarity values in the categorical and numerical components, then Equation 8.10 needs to be modified as follows:

$$Sim(\bar{X}, \bar{Y}) = \lambda \cdot NumSim(\bar{X}_n, \bar{Y}_n)/\sigma_n + (1 - \lambda) \cdot CatSim(\bar{X}_c, \bar{Y}_c)/\sigma_c$$

By performing this normalization, the value of  $\lambda$  becomes more meaningful, as a true *relative weight* between the two components. By default, this weight can be set to be proportional to the number of attributes in each component, unless specific domain knowledge is available about the relative importance of attributes.

### 8.4.4 Density-Based Methods

Density-based methods can be naturally extended to discrete data, because numerical attributes are often discretized to create frequency profiles. In categorical data, these frequency profiles can be constructed directly on different combinations of values of the discrete attribute. The corresponding methods for numerical data have been discussed in section 4.4.3 of Chapter 4. In the case of categorical data, such methods are very natural to use, since the additional step of discretization does not need to be performed in order to convert the numerical values to discrete values. In the case of mixed attribute data, the numerical attributes may be discretized, whereas the categorical attributes may be retained in their original form. All other steps are identical to the methodology discussed in section 4.4.3 of Chapter 4.

### 8.4.5 Clustering Methods

As in the case of numerical data, outliers are defined as data points that are not members of clusters, or are not in sufficient proximity of clusters. While numerous clustering methods

exist for categorical data, such methods are often not applied to the categorical domain because of the greater difficulty in defining similarity between individual records and cluster representatives (centroids). A clustering method that is commonly used in the categorical domain for outlier analysis is the EM-method discussed earlier in this section. In that case, the negative logarithms of the fit probabilities can be used to quantify outlier scores. Although it is possible to define outliers on the basis of non-membership to clusters, such methods are not optimized to finding true anomalies in the data and may often discover noise. Stronger ways of scoring data points based on the distance to the nearest cluster and the size of the nearest cluster are discussed in sections 4.2 and 5.2.4. The only difference is that the distances need to be defined for categorical data, as discussed earlier in this chapter. Since clustering methods tend to be sensitive to different parameter choices such as the number of clusters, it is advisable to score each point in multiple ways with different clusterings. The scores from these different clusterings are averaged in order to create the final outlier score.

A discussion of common clustering methods for categorical data is also provided in the bibliographic section of this chapter, although most of these methods are optimized towards finding clusters rather than outliers. Several recent books [23, 33] discuss clustering methods and similarity measures for categorical data. A clustering method for categorical data is described in [29], and an outlier detection method is also integrated into this technique.

## 8.5 Outlier Detection in Binary and Transaction Data

---

A significant amount of categorical data is binary in nature. For example, transaction data, which contains customer buying patterns is almost always binary. Furthermore, all forms of categorical data and numerical data can always be transformed to binary data by various forms of discretization. Since binary data is a special case of all forms of categorical and numerical data, most of the methods discussed in this and other chapters can be applied to such data. Nevertheless, transaction data, which exists in a binary form in its natural state, is different from the binary data obtained by artificial conversion in terms of a number of practical aspects. The former type of data is typically very high dimensional and sparse with highly correlated attributes. Therefore, a number of methods have also been designed that are specifically optimized to these types of data. These methods can also be applied to categorical data sets after applying binarization, provided that the resulting binary data sets retain their sparsity.

### 8.5.1 Subspace Methods

Since transaction data is inherently high-dimensional, it is natural to utilize subspace methods [4] in order to identify the relevant outliers. The challenge in subspace methods is that it is often computationally impractical to define subspaces (or sets of items), which are sparse for outlier detection. For example, in a sparse transaction database containing hundreds of thousands of items, sparse itemsets are the norm rather than the rule. Therefore, a subspace exploration for sparse itemsets is likely to report the vast majority of patterns. The work in [254] addresses this challenge by working in terms of the relationship of transactions to dense subspaces rather than sparse subspaces. In other words, this is a reverse approach of determining transactions that are *not included* in most of the relevant dense subspace clusters of the data. In the context of transaction data, subspace clusters are essentially frequent patterns.

The idea in such methods is that frequent patterns are less likely to occur in outlier transactions, as compared to normal transactions. Therefore, a measure has been proposed in the FP-Outlier work [254], which sums up the support of all frequent patterns occurring in a given transaction in order to provide the outlier score of that transaction. The total sum is normalized by dividing with the number of frequent patterns. However, this term can be omitted from the final score because it is the same across all transactions and does not affect the relative ordering of outlier scores.

Let  $\mathcal{D}$  be a transaction database containing the transactions denoted by  $T_1 \dots T_N$ . Let  $s(T_i, \mathcal{D})$  represent the support of transaction  $T_i$  in  $\mathcal{D}$ . Therefore, if  $FPS(\mathcal{D}, s_m)$  represents the set of frequent patterns in the database  $\mathcal{D}$  at minimum support level  $s_m$ , the frequent pattern outlier factor  $FPOF(T_i)$  of a transaction  $T_i \in \mathcal{D}$  at minimum support  $s_m$  is defined as follows:

$$FPOF(T_i) = \frac{\sum_{X \in FPS(\mathcal{D}, s_m), X \subseteq T_i} s(T_i, \mathcal{D})}{|FPS(\mathcal{D}, s_m)|}$$

Intuitively, a transaction containing a large number of frequent patterns with high support will have high value of  $FPOF(T_i)$ . Such a transaction is unlikely to be an outlier, because it reflects the major patterns in the data.

As in other subspace methods, such an approach can also be used in order to explain why a data point may not be considered an outlier. Intuitively, the frequent patterns with the largest support, which are also not included in the transaction  $T_i$  are considered *contradictory patterns* to  $T_i$ . Let  $S$  be a frequent pattern not contained in  $T_i$ . Therefore,  $S - T_i$  is non-empty, and the *contradictiveness* of frequent pattern  $S$  to the transaction  $T_i$  is defined by  $s(S, \mathcal{D}) * |S - T_i|$ . Therefore, a transaction which does not have many items in common with a very frequent itemset is likely to be one of the explanatory patterns for the  $T_i$  being an outlier. The patterns with the top- $k$  values of contradictiveness are reported as the corresponding explanatory patterns.

At an intuitive level, such an approach is analogous to non-membership of data points in clusters in order to define outliers, rather than directly trying to determine the deviation or sparsity level of the transactions. As was discussed in the chapter on clustering-based methods, such an approach may sometimes not be able to distinguish between noise and anomalies in the data. However, the approach in [254] indirectly uses the weight and number of clusters in the outlier score. Furthermore, it uses *multiple* patterns in order to provide an ensemble score. This is at least partially able to alleviate the noise effects. In the context of very sparse transactional data, in which direct exploration of rare subspaces is infeasible, such an approach would seem to be a reasonable adaptation of subspace methods. One can view the FP-Outlier method as the unsupervised one-class analog of rule-based method in classification. It is noteworthy that frequent pattern mining methods are often used for rule-based methods in classification [364]. In fact, one can view many subspace methods as the one-class analogs of rule-based methods in classification; in the case of the FP-Outlier method, this similarity is particularly obvious. Similar methods have also been used for subspace outlier detection in high-dimensional and numeric data (cf. section 5.2.4 of Chapter 5).

Frequent pattern mining methods are closely related to information-theoretic measures for anomaly detection. This is because frequent patterns can be viewed as a code-book in terms of which to represent the data in a compressed form. It has been shown in [494], how frequent patterns can be used in order to create a compressed representation of the data set. Therefore, a natural extension is to use the description length [497] of the compressed data in order to compute the outlier scores. This approach was further improved in [42].

## 8.5.2 Novelties in Temporal Transactions

Transaction data are often temporal in nature, in which the individual transactions are associated with a time-stamp. A new transaction that is not consistent with the “normal” model of previous transactions can be viewed as a novelty. A method for novelty detection in fast binary data streams was proposed in [29]. In fact, this approach falls in a general class of proximity-based methods, which will be discussed for text data later in this chapter. Thus, this method is fairly general and can be applied to both text and categorical data.

The broad idea of the approach is to continuously maintain the clusters in the underlying temporal data stream. Data points that do not naturally fit into any of these clusters are regarded as novelties. Thus, for each incoming data point, its largest similarity to the centroids of the current set of clusters is evaluated. If this similarity is statistically too low, then the data point is flagged as a novelty, and it is placed in a cluster of its own. The determination of the statistical threshold on similarity is achieved on the basis of the average similarity of other points to the centroid of the cluster. Subsequently, it is possible that this data point may represent the beginning of a new trend or cluster in the data. Such situations are quite common in many novelty-detection applications in which a detected outlier eventually becomes a normal part of the data. However, in some cases, such data points may continue to remain outliers over time. This issue will be discussed in more detail in Chapter 9 on outlier detection in temporal data.

## 8.6 Outlier Detection in Text Data

---

Text data shares a number of conceptual similarities with high-dimensional and sparse transaction data. However, word frequencies are often associated with text attributes; therefore, the data set is typically not binary. Nevertheless, simplified binary representations are sometimes used for text in which only presence or absence of words are considered. Text data can be viewed as a sparse, high-dimensional, and non-negative form of multidimensional data. Therefore, most of the probabilistic, linear, and proximity-based methods for binary and multidimensional data can be generalized to text. However, there are some subtle differences in how these models are implemented because of the sparse, high-dimensional, and non-negative nature of text. There are also some interesting connections between probabilistic and linear models, because some linear (matrix factorization) models for text are also probabilistic in nature. In the following, we will use the same notation as used for multidimensional data throughout the book. Therefore, the frequency-vector of words for the  $i$ th document is denoted by  $\overline{X}_i$ , and it is possible for this vector to be binary.

### 8.6.1 Probabilistic Models

Probabilistic models are commonly used for *soft* clustering of text data in which documents are assigned probabilities of membership to various clusters. By “soft” clustering, we refer to the fact that documents are not clearly assigned to specific clusters, but have a probability of assignment to each cluster. This approach is essentially an application of the expectation-maximization (EM) algorithm (see Chapter 2) to the text domain.

The main idea is to represent a corpus as a mixture of various distributions, the parameters of which are estimated using a particular document collection. The number of documents is denoted by  $N$ , the number of mixture components by  $k$ , and the lexicon size (terms) by  $d$ . The primary assumptions of the mixture-modeling approach are as follows:

- Each document is assumed to have a probability of belonging to one of the  $k$  mixture components denoted by  $\mathcal{G}_1 \dots \mathcal{G}_k$ . The probability that the document  $\bar{X}_i$  belongs to the cluster  $\mathcal{G}_j$  is denoted by  $P(\mathcal{G}_j|\bar{X}_i)$ . Since documents belong to clusters in a probabilistic sense, this approach creates a soft partitioning. The value of  $P(\mathcal{G}_j|\bar{X}_i)$  is estimated using the expectation-maximization approach on a generative model, and is one of the primary outputs of the algorithm.
- Each cluster is associated with a probability vector that quantifies the probability of the different terms in the lexicon for that cluster. Let  $t_1 \dots t_d$  be the  $d$  terms in the lexicon. Then, if a document belongs to cluster  $\mathcal{G}_j$ , the probability that the term  $t_l$  occurs in it is given by  $P(t_l|\mathcal{G}_j)$ . The probability  $P(t_l|\mathcal{G}_j)$  is another parameter estimated by the expectation-maximization algorithm.

These assumptions are similar to those of the mixture model discussed in Chapter 2. In fact, the generative model is identical, and the same set of steps is used:

1. Select the  $r$ th mixture component with probability  $\alpha_r = P(\mathcal{G}_r)$ .
2. Generate the term frequencies of a document based on the predefined distribution of  $\mathcal{G}_r$ . A simple Bernoulli distribution is used for the binary representation of text, although the multinomial model is used to handle explicit word frequencies. The typical parameter used to describe this distribution represents a term generation frequency from the mixture component. For example, the parameter  $p_l^{(r)} = P(t_l|\mathcal{G}_r)$  is often used, which is the probability of term  $t_l$  being generated from mixture component  $\mathcal{G}_r$ .

The observed corpus is assumed to be an output of the generative process, and therefore the parameters of each of the mixture components are learned using the generative process. The prior probabilities  $\alpha_r$  and the parameters of the generative distribution of mixture component  $\mathcal{G}_r$  need to be estimated by this approach. It is noteworthy that the use of a Bernoulli model implicitly assumes that the frequencies of the documents are ignored, and the documents are generated in binary format. The following discussion will assume the use of a Bernoulli distribution, although the basic methodology remains unchanged for the multinomial distribution; the only difference is in the distribution-specific estimation of the underlying parameters.

The probability  $P(\mathcal{G}_j|\bar{X}_i)$  of the  $i$ th document being generated by the  $j$ th cluster can be computed using the Bayes rule as follows:

$$P(\mathcal{G}_j|\bar{X}_i) = \frac{P(\mathcal{G}_j)P(\bar{X}_i|\mathcal{G}_j)}{\sum_{r=1}^k P(\mathcal{G}_r)P(\bar{X}_i|\mathcal{G}_r)} = \frac{\alpha_j P(\bar{X}_i|\mathcal{G}_j)}{\sum_{r=1}^k \alpha_r P(\bar{X}_i|\mathcal{G}_r)} \quad (8.11)$$

The right-hand side of this equation is easy to estimate with the use of the probability distribution modeled for  $\mathcal{G}_j$ . For example, if a Bernoulli distribution is assumed for  $\mathcal{G}_j$ , the right-hand side may be expressed<sup>2</sup> as follows:

$$P(\mathcal{G}_j|\bar{X}_i) = \frac{\alpha_j \cdot \prod_{t_l \in \bar{X}_i} p_l^{(j)} \cdot \prod_{t_l \notin \bar{X}_i} (1 - p_l^{(j)})}{\sum_{r=1}^k \alpha_r \cdot \prod_{t_l \in \bar{X}_i} p_l^{(r)} \cdot \prod_{t_l \notin \bar{X}_i} (1 - p_l^{(r)})} \quad (8.12)$$

The main problem here is that we have no idea of what the values of the parameters  $p_l^{(j)}$  and  $\alpha_j$  ought to be in order to compute the aforementioned equation. After all, the values

<sup>2</sup>We have used notations somewhat loosely in Equation 8.12. Even though  $\bar{X}_i$  is not a set (but a binary vector), we are treating it as a bag of words when using expressions like  $t_l \in \bar{X}_i$  on the right-hand side.



of  $p_l^{(j)}$  can be estimated (using maximum-likelihood methods) only if we knew the probabilistic assignment probability  $P(\mathcal{G}_j|\bar{X}_i)$  of the documents to the various clusters. A similar observation holds for the parameter  $\alpha_j$ . In other words, the right-hand of Equation 8.12 can be estimated only if we already knew the value on the left-hand side. Therefore, there is an inherent circularity to this relationship, which is resolved with iterative methods.

We start with a random assignment of documents to mixture components, which gives us an initial value of each  $P(\mathcal{G}_j|\bar{X}_i) \in \{0, 1\}$ . Subsequently, we estimate each  $\alpha_j$  as the fraction of documents belonging to  $\mathcal{G}_j$  in the initial random assignment. The value of  $p_l^{(j)}$  is estimated as the fraction of documents in  $\mathcal{G}_j$  that contain the term  $t_l$ . Subsequently, the following iterative expectation-maximization steps are applied:

- **(E-Step):** The currently initialized values of the parameters are used to estimate the generative probability  $P(\mathcal{G}_j|\bar{X}_i)$  of each document from one of the clusters according to Equation 8.12.
- **(M-Step):** The value of  $P(\mathcal{G}_j|\bar{X}_i)$  is treated as the weight of document  $i$  in mixture-component  $j$  and is used to estimate various parameters as follows. The value of each  $p_l^{(j)}$  is estimated as the (weighted) fraction of documents in cluster  $j$  that contain term  $t_l$ . The value of  $\alpha_j$  is computed as the (weighted) fraction of the points belonging to the cluster  $j$ . In both cases, Laplacian smoothing may be used to avoid overfitting.

These steps are iterated to convergence. One can compute the fit probability of document  $\bar{X}_i$  being generated by the model as the sum of the generative probabilities over various mixture components:

$$P(\bar{X}_i) = \sum_{r=1}^k P(\mathcal{G}_r) \cdot P(\bar{X}_i|\mathcal{G}_r) \quad (8.13)$$

$$= \sum_{r=1}^k \alpha_r \cdot \prod_{t_l \in \bar{X}_i} p_l^{(r)} \cdot \prod_{t_l \notin \bar{X}_i} (1 - p_l^{(r)}) \quad (8.14)$$

Documents that have low fit probability can be declared outliers. One can also treat  $P(\bar{X}_i)$  as an outlier score in which smaller values are more indicative of outliers. Therefore, one can use extreme-value analysis on the score. However, for reasons discussed in section 6.2.1 of Chapter 6, extreme-value analysis should be applied on the logarithms of  $P(\bar{X}_i)$  because very low probability values lose numerical stability, and discrimination between the different values is not well-represented by probability values in the context of hypothesis testing.

The EM-algorithm can also be adapted to online scenarios. In such scenarios, the parameters of the model are iteratively updated in the context of an online data stream. In such cases, only a window of the current set of documents is used for the update equations, so that the model is updated over time in order to reflect the current trends in the data. Whenever a document is encountered which does not fit well into the current list of topics, a new topic can be initiated containing this document in a dominant way. Thus, this scenario corresponds to the case of growing clusters.

## 8.6.2 Linear Models: Latent Semantic Analysis

The goal of Latent Semantic Analysis (LSA) is largely to *improve the underlying data representation*, so as to reduce the impact of noise and outliers. One of the problematic

aspects of the text representation is that of *synonymy* and *polysemy*. Synonymy refers to the fact that multiple words might describe the same concept. For example, a “*car*” might also be referred to as an “*automobile*.” Therefore, two documents containing the words “*car*” and “*automobile*,” respectively, might not be considered sufficiently similar. In polysemy, the same word might correspond to different concepts. For example, the word “*Jaguar*” might either refer to a car or a cat. Clearly, the presence of such words can be viewed as noise, which can greatly reduce the quality of many applications such as similarity search. Latent semantic analysis is a method to improve the underlying data representation, so as to reduce the impact of such noise. This broader principle has also been discussed in the section on noise correction with PCA in Chapter 3.

LSA is the text-centric avatar of SVD. Specifically, let  $D$  be the  $N \times d$  term-document matrix in which the  $(i, j)$ th entry is the normalized frequency for term  $j$  in document  $i$ . Then,  $D^T D$  is a  $d \times d$  matrix, which can be viewed as a similarity matrix between pairs of dimensions. For mean-centered data, it would also be equivalent to a scaled version of the covariance matrix, although one rarely performs mean-centering<sup>3</sup> in sparse domains like text. The largest eigenvectors of  $D^T D$  are computed in order to represent the text. In typical text collections, only about 300 to 400 eigenvectors are required for the representation. One excellent characteristic of LSA is that the truncation of the dimensions removes the noise effects of synonymy and polysemy, and the similarity computations are better regulated by the semantic concepts in the data.

It should be further noted that documents that are incoherent, spam, or otherwise incongruent with the remaining data are more easily revealed in the new representation because they will have larger components along the small eigenvalues. Such documents can be more easily distinguished from the data. If desired, LSA can also be used in combination with proximity-based techniques for more effective outlier detection. This is because the quality of the similarity computations is improved in the latent space, and one can therefore identify outliers more easily.

### 8.6.2.1 Probabilistic Latent Semantic Analysis (PLSA)

PLSA can be considered both a probabilistic method and a linear method like LSA. However, this book has classified it among linear methods because of its natural connections to LSA. LSA is a straightforward application of SVD to the  $N \times d$  document-term matrix. As discussed in Chapter 3, one can write a low-rank matrix-factorization of SVD with data matrix  $D$  as follows:

$$D \approx (Q\Lambda)P^T = UV^T$$

In the case of LSA, the matrix  $D$  is the document-term matrix, the matrices  $Q$  and  $P$  are  $N \times k$  and  $d \times k$  matrices, respectively, and  $\Lambda$  is a  $k \times k$  diagonal matrix, where  $k$  is the rank of the decomposition (i.e., the number of eigenvectors selected by LSA). Therefore,  $U$  and  $V$  are  $N \times k$  and  $d \times k$  matrices, respectively. The rows of the matrix  $Q\Lambda$  provide the  $k$ -dimensional latent representations of the documents and the columns of the matrix  $P$  provide the  $k$  orthonormal basis vectors of representation. These basis vectors are also the  $d$ -dimensional eigenvectors of  $D^T D$ . One can write the corresponding optimization model

---

<sup>3</sup>This is a common difference between PCA and SVD. The two are equivalent when the data is mean-centered. However, SVD can be performed on any matrix, whereas PCA is always performed only on mean-centered matrices.

as follows:

$$\begin{aligned} \text{Minimize } J &= \|D - UV^T\|^2 \\ \text{subject to:} \\ \text{Columns of } U &\text{ are mutually orthogonal} \\ \text{Columns of } V &\text{ are mutually orthonormal} \end{aligned}$$

SVD (and LSA) can be shown to be solutions to this optimization model. PLSA is a variation of this optimization model. For example, the objective function of minimizing the Frobenius norm of the residual matrix ( $D - UV^T$ ) is a way of ensuring that  $D$  matches  $UV^T$  as closely as possible. There are other ways of changing the objective function to provide a more probabilistic interpretation. The specific optimization model solved by PLSA, which is a probabilistic avatar of nonnegative matrix factorization, is as follows:

$$\begin{aligned} \text{Maximize } J &= \text{Likelihood matching between } D \text{ and } UV^T \\ \text{subject to:} \\ U &\text{ is nonnegative} \\ V &\text{ is nonnegative} \end{aligned}$$

Whenever one talks of a “maximum likelihood matching” one is talking about a generative process. What is this generative process, and what are the parameters of this process?

The matrix  $D$  can be viewed as an outcome of an generative process. In fact, a set of matrices  $Q$ ,  $\Lambda$  and  $P$  can be extracted from  $U$  and  $V$ , which form the parameters for this generative process. The columns of  $U$  are scaled to sum to 1 to create  $Q$ . The columns of  $V$  are scaled to sum to 1 to create  $P$ . The  $(r, r)$ th diagonal matrix  $\Lambda$  is *proportional* to the product of the scaling factors of  $U$  and  $V$ , respectively, for the  $r$ th columns. We use the word “proportional” because the diagonal entries are scaled to sum to 1. Therefore, we have  $UV^T \propto Q\Lambda P^T$ . These parameters are used to define the following generative process for the data matrix  $D$ :

1. Select a mixture component  $\mathcal{G}_r$  with probability  $\Lambda_{rr}$ .
2. Select the row  $i$  of  $D$  with probability proportional to the  $(i, r)$ th entry of  $Q$  and the column  $j$  of  $D$  with probability proportional to the  $(j, r)$ th entry of  $P$ . Increment the  $(i, j)$ th entry of  $D$  by 1.

An implicit assumption is that the row  $i$  and column  $j$  are selected in a *conditionally independent* way in the second step after selecting the mixture component  $\mathcal{G}_r$ . One way of interpreting the parameters in  $Q$ ,  $P$  and  $\Lambda$  in the context of the generative process above is as follows. Let  $\bar{X}_i$  be the  $i$ th document and  $t_j$  be the  $j$ th term. The  $(r, r)$ th entry in the diagonal matrix  $\Lambda$  contains  $P(\mathcal{G}_r)$ , which is used in the first step of selecting the mixture component. The  $(i, r)$ th entry in  $Q$  contains  $P(\bar{X}_i|\mathcal{G}_r)$ , and the  $(j, r)$ th entry in  $P$  contains  $P(t_j|\mathcal{G}_r)$ . These probabilities are used in the aforementioned generative process, and are also *exactly* the parameters used by the generative process of section 8.6.1; however, the generative processes used in the two cases are different in spite of similarity in parametrization.

PLSA learns a matrix factorization (and underlying probabilistic parameters) that maximizes the likelihood of the observed document-term frequencies to be generated by this process. The EM steps for PLSA may be defined as follows:

- **(E-step):** Estimate  $P(\mathcal{G}_r|\bar{X}_i, t_j)$  for each document-term pair using the currently estimated parameters in conjunction with the Bayes rule:

$$P(\mathcal{G}_r|\bar{X}_i, t_j) = \frac{P(\mathcal{G}_r) \cdot P(\bar{X}_i|\mathcal{G}_r) \cdot P(t_j|\mathcal{G}_r)}{\sum_{s=1}^k P(\mathcal{G}_s) \cdot P(\bar{X}_i|\mathcal{G}_s) \cdot P(t_j|\mathcal{G}_s)} \quad \forall r \in \{1 \dots k\} \quad (8.15)$$

- **(M-step):** Estimate the current parameters in  $Q$ ,  $P$  and  $\Lambda$  by using the conditional probabilities in the first step as weights for entries belonging to each generative component. This is achieved as follows:

$$Q_{ir} = P(\bar{X}_i|\mathcal{G}_r) = \frac{\sum_j P(\bar{X}_i, t_j) \cdot P(\mathcal{G}_r|\bar{X}_i, t_j)}{P(\mathcal{G}_r)} \propto \sum_j D_{ij} P(\mathcal{G}_r|\bar{X}_i, t_j)$$

$$P_{jr} = P(t_j|\mathcal{G}_r) = \frac{\sum_i P(\bar{X}_i, t_j) \cdot P(\mathcal{G}_r|\bar{X}_i, t_j)}{P(\mathcal{G}_r)} \propto \sum_i D_{ij} P(\mathcal{G}_r|\bar{X}_i, t_j)$$

$$\Lambda_{rr} = P(\mathcal{G}_r) = \sum_{i,j} P(\bar{X}_i, t_j) \cdot P(\mathcal{G}_r|\bar{X}_i, t_j) \propto \sum_{i,j} D_{ij} P(\mathcal{G}_r|\bar{X}_i, t_j)$$

The constants of proportionality are set by ensuring that the probabilities in the columns of  $P$ ,  $Q$  and the diagonal of  $\Lambda$  each sum to 1.

The generative process of section 8.6.1 is somewhat different from the one discussed in this section. The technique in section 8.6.1 generates each document with a single iteration of the generative process, whereas the technique in this section fills up the *entries* of the document-term matrix with the generative process. Note that one can interpret the entries in the (scaled) matrix  $D$  as observed values of  $P(\bar{X}_i, t_j)$ . The  $(i, j)$ th entry of  $D$  can be interpreted in terms of the generative process to derive a matrix factorization as follows:

$$\begin{aligned} D_{ij} \propto P(\bar{X}_i, t_j) &= \sum_{r=1}^k \underbrace{P(\mathcal{G}_r)}_{\text{Select } r} \cdot \underbrace{P(\bar{X}_i, t_j|\mathcal{G}_r)}_{\text{Select } \bar{X}_i, t_j} \quad [\text{Generative probability of incrementing } (i, j)] \\ &= \sum_{r=1}^k P(\mathcal{G}_r) \cdot P(\bar{X}_i|\mathcal{G}_r) \cdot P(t_j|\mathcal{G}_r) \quad [\text{Conditional independence}] \\ &= \sum_{r=1}^k P(\bar{X}_i|\mathcal{G}_r) \cdot P(\mathcal{G}_r) \cdot P(t_j|\mathcal{G}_r) \quad [\text{Rearranging product}] \\ &= \sum_{r=1}^k Q_{ir} \cdot \Lambda_{rr} \cdot P_{jr} = (Q\Lambda P^T)_{ij} \quad [\text{Note the similarity to LSA}] \end{aligned}$$

One can generate outlier scores not only for each document but also for each entry. For each entry, one might use the (absolute value of the) entry-wise residuals of the matrix factorization as an approximation to the outlier score. The document-wise scores are the sum of the squares of the residuals in each row. Similarly, the term-wise scores are the sum of the squares of the residuals in each column. Larger values of the scores are indicative of a greater tendency of a point to be an outlier. Therefore, one can identify document-wise, term-wise, and entry-wise outliers. This is because the PLSA technique is well-suited to explaining the generative process of the entire matrix in an entry-wise fashion, rather than as a generative process in document-wise fashion.

It is noteworthy that PLSA is a special case of nonnegative matrix factorization, in which the objective function uses maximum-likelihood matching. Traditional forms of nonnegative matrix factorization use the Frobenius norm of the residual matrix (like PCA and SVD). As a practical matter, the difference between these two forms of factorization is very limited; however, exactly the same solution is not obtained in the two cases because of differences in the underlying objective function.

PLSA was the first technique in the category of *topic-modeling* techniques. Many other related probabilistic methods such as Latent Dirichlet Allocation (LDA) are commonly used for topic modeling [90, 91]. All classes of topic-modeling methods can be adapted to outlier detection. LDA is particularly useful if the outlier scores of new documents (i.e., those not included in the training data) are to be computed; this situation arises commonly in a temporal or streaming setting. PLSA can compute the scores of only in-sample documents.

### 8.6.3 Proximity-Based Models

As in all data types, the design of the similarity function is important in text data. The word frequencies need to be normalized in terms of their relative frequency of presence in the document and over the entire collection. For example, a common word such as “*the*” is not as discriminative (or informative) for outlier detection, as compared to a word such as “*car*.” A common approach for text processing is the *vector-space based tf-idf* representation [472]. In the tf-idf representation, the term frequency for each word is normalized by the *inverse document frequency*, or *idf*. The inverse document frequency normalization reduces the weight of common terms which occur more frequently in the collection. This reduces the importance of common terms in the collection, ensuring that the matching of documents is more influenced by that of more discriminative words which have relatively low frequencies in the collection. Therefore, if  $n_i$  is the number of documents in which the  $i$ th word occurs, and  $N$  is the total number of documents, the term frequency of the  $i$ th word is multiplied with  $\log(N/n_i)$  in order to emphasize the importance of infrequent words.

After this normalization has been achieved, the cosine similarity function is applied to the vector-space representations in order to measure similarity. The cosine measure is equal to the dot product of two vectors, once they have been normalized to unit length. The idea of normalization is to compute similarities in a fair way between documents of varying lengths. Thus, for two normalized document frequency vectors  $\bar{X}$  and  $\bar{Y}$ , the cosine function  $Cosine(\bar{X}, \bar{Y})$  can be defined as follows:

$$Cosine(\bar{X}, \bar{Y}) = \frac{\bar{X} \cdot \bar{Y}}{\|\bar{X}\| \cdot \|\bar{Y}\|} \quad (8.16)$$

Here,  $\|\cdot\|$  represents the  $L_2$ -norm. The earliest work on proximity-based models was done in the context of the topic detection and tracking (TDT) project [49, 622]. Most of the methods [47, 48, 49, 622] determine key novelties in the stream by using the following broad two step framework for each incoming document, with respect to a current summary which is maintained for the corpus:

- Compute the similarity of the incoming document to the current summary of the corpus. The summary of the corpus could correspond to either a fine-grained clustering, or a sample of the documents which have been received so far. Report any and all documents that have low similarity with the current summary as anomalies. Alternatively, the similarity to the closest cluster centroid or to the nearest document in the

sample could be reported as the outlier score. Smaller values indicate greater degree of outlierness.

- Update the summary of the incoming documents in the data stream by incorporating the incoming document into an appropriate component of the summary. For example, a clustering approach might adjust the centroid of the closest cluster to reflect the incorporation of the document in the cluster.

Recently, a method has also been proposed for novelty detection in fast data streams [29]. This approach uses an exponential decaying model in order to determine clusters and novelties simultaneously from the data stream. Eventually such novelties are converted into clusters, if more data points arrive which match the content of the document. Such methods are also able to identify novelties that are eventually converted into broad trends in the data stream.

The method of [29] makes the assumption of fixed-memory availability, because of which the number of clusters in the data remain fixed. As a result, a current cluster needs to be ejected from the buffer when a new cluster is inserted. Typically, such ejected clusters are chosen as those (faded) trends that have not been updated for a while. However, it is possible that at a future time, such a trend will re-appear. As a result, a similar cluster may need to be created at a future point in time. Such outliers are referred to as *infrequently recurring outliers*, which are distinct from novelties that were never seen before. Under a *fixed space* constraint, it is often not possible for summarization methods to distinguish between novelties and infrequently recurring outliers. This is because recurring outliers are often dropped from the stream summary to conserve space. However, if the number of clusters is allowed to grow over time, it may be possible to distinguish between such events. Nevertheless, such an approach is likely to slow down over time and may no longer be relevant to the streaming scenario.

### 8.6.3.1 First Story Detection

The detection of novelties in a stream of text documents is closely related to the problem of *first story detection*. Much of the work on outlier detection in the text domain has been studied in the context of the problem of first-story detection in a streaming context such as a news wire service. Although some related methods for temporal data are studied in Chapter 9, the case of text is presented here, because it is more closely related to the other material in this chapter. Furthermore, there are very few methods for text outlier detection in non-temporal scenarios, and most of the methods for *first-story detection* can trivially be generalized to the non-temporal scenario. In the context of temporal data, such data points are also referred to as *novelties*. Two broad classes of methods exist for this problem, corresponding to proximity-based models and probabilistic models. Although both of these methods have been proposed in the context of temporal data, they can trivially be generalized to the non-temporal scenario, since the temporal component is utilized only from the perspective of data subset selection in the modeling process. In the non-temporal scenario, these methods simply need to be applied to the entire data instead of the previous history. Many of the non-temporal methods in this chapter can also be generalized to the temporal scenario.

## 8.7 Conclusions and Summary

---

This chapter studies methods for outlier detection in categorical, text, transaction, and mixed data sets. Such data sets share a number of broad similarities with one another. Therefore, it is natural to study the outlier detection methods for these different problems in a single chapter. Probabilistic models can be adapted to various data domains by defining appropriate generative models. Most of the existing proximity-based outlier detection methods can also be easily adapted to these different domains by defining appropriate similarity functions. Mixed-attribute data sets can be addressed by proximity-methods, as long as appropriate methods are defined for weighting the similarity of each component. Both text and binary data tend to be high dimensional and sparse. Therefore, many of the techniques for one domain can be used for the other, and vice versa. Linear models and matrix factorization methods are often used in such settings.

## 8.8 Bibliographic Survey

---

The earliest work on categorical outlier detection was performed in the context of clustering algorithms [210, 220, 226, 283, 307]. In many clustering algorithms, the outliers are identified as a side-product of clustering algorithms. Some of the earliest work on probabilistic clustering for outlier detection in categorical data is proposed in [578]. This approach uses a joint probability model that can address both categorical and mixed attributes. The approach can be effectively used for mixed data, since probabilistic models are able to normalize in an intuitively uniform way over numerical and categorical attributes. Furthermore, the approach uses a time-decaying model for the clustering component. This ensures that it can be used effectively in an evolving data set.

Although linear models have not been studied extensively in the context of outlier detection of categorical data, they are particularly useful because they can naturally account for inter-attribute correlations. A number of such correspondence-analysis methods [265] are specifically optimized for categorical data. However, the potential of these methods has not been fully explored in the context of outlier detection. A particularly important challenge with categorical data surfaces when a large number of potential values exists for a categorical attribute. In such cases, the increasing sparsity of the representation can make correspondence-analysis methods ineffective.

Proximity-based algorithms can be naturally generalized to categorical data as long as a similarity or neighborhood measure is available in order to perform the distance (similarity) computations. However, in most of these cases, the similarity computation is performed with the use of relatively straightforward overlap measures. Categorical similarity measures are crucial to all proximity-based applications such as clustering. Similarity measures in categorical data are closely related to corresponding measures in text data, because the conversion of categorical data to binary data often resembles sparse text data. In both cases, the representation is normalized, so that aggregate statistics in order to provide lower values to attributes which are rare in occurrence. The most common example in the case of text is the use of the inverse document frequency in order to normalize the similarity measures [472]. The corresponding measure in the case of categorical data is referred to as the inverse occurrence frequency.

Many of the similarity measures defined in the literature [39, 87, 154, 346, 423], are *contextual* in the sense that they use the neighborhood of a point in order to compute similarity. Of course, the relationship between similarity derivation and neighborhood computation is

circular, because neighborhood computation itself requires the computation of similarity. Therefore, many of these techniques use simpler measures such as the *Overlap* in order to define the initial neighborhood for similarity computation. Subsequently, iterative methods [154] are used in order to simultaneously refine the neighborhood and the similarity computation. Therefore, it is sometimes also helpful to develop similarity measures, which are based on the aggregate statistics, rather than on specific neighborhoods of data points.

Such aggregate methods either measure similarity only between same values of an attribute (using different kinds of normalizations), or they compute similarity between different values of the same attribute [93, 358]. Some of the common techniques that do not use similarity between values of different attributes are discussed in [209, 222, 472, 109]. Other measures that use functions of both matches and mismatches on different attribute values are proposed in [54, 358, 498]. Many of these similarity measures have been tested in the context of the outlier detection problem [93]. Contextual measures [154] have also been proposed, but have not been widely explored in the context of the outlier detection problem. Similarity measures have also been provided that can be used in the context of categorical and mixed data [596]. The effectiveness of these measures in the context of the outlier detection problem was studied in the same work.

Link-based methods [563] model common attribute values between data instances. The work in [563] models the categorical values as a graph, and distances between data instances are modeled by using the connectivity of this graph. This type of modeling can be used to identify outliers. Distributed link-based methods for finding outliers in categorical and mixed data are proposed in [421]. Two pattern-based methods for anomaly detection are discussed in [42, 155], the first of which is information-theoretic in nature.

The problem of binary and transaction data is especially challenging because of its high dimensionality. As shown in [254], direct adaptations of various methods such as replicator neural networks [567] and clustering [255] do not work effectively. In such data, a given transaction may contain only a very tiny fraction of the available items. The work in [254] shows how to adapt subspace methods to transaction data by finding transactions which do *not* lie in dense subspaces (or do not contain frequent patterns), rather than determining transactions that contain sparse subspaces. Another approach that is based on frequent patterns is proposed in [410]. Methods that use frequent patterns for information-theoretic anomaly detection methods are discussed in [42, 323, 497].

In the context of text data, latent semantic analysis is commonly used to identify and reduce noise. Latent semantic analysis [162] (LSA) is closely related to principal component analysis [296], which is studied in detail in Chapter 3. LSA was originally proposed as a method for retrieval of text [162], and its effects on improving the quality of similarity in text were observed in [425].

The problem of topic detection and tracking has been studied both in the unsupervised scenario [29, 47, 48, 49, 95, 306, 585, 608] and in the supervised scenario [584, 586]. Some of these methods have also been generalized to social streams, which contain a combination of text, linkage, and other side information, such as spatiotemporal data [30, 194, 435]. Most of the methods for novelty detection are based on proximity models, though a few methods [608] are also based on probabilistic models. A variety of probabilistic models such as PLSA and LDA [90, 91, 271] can be used for outlier analysis in text data. In the supervised scenario, it is assumed that training data are available in order to enhance the novelty detection process. This can be achieved either by providing examples of the rare class, the normal class or both. Such methods are often quite useful for detecting novel events in conventional news streams or social media such as *Twitter* [435]. A probabilistic model for determining novelties in text streams is proposed in [608]. Another aggregate



method for detection of correlated bursty topic patterns from coordinated text streams is proposed in [562].

## 8.9 Exercises

---

1. Download the *Adult data* from the UCI Machine Learning Repository [203]. Determine outliers with the use of  $k$ -nearest neighbor algorithm on the categorical attributes only by using:
  - Match-based similarity measures.
  - Inverse occurrence frequency similarity measure.

How do the outliers compare to one another?

2. Repeat Exercise 1 by including the numerical attributes in the analysis. Use a Euclidean distance measure on the numerical attributes. Do you obtain the same outliers?
3. Compute the principal components of the mixed representation of the *Adult data* with and without attribute-specific normalization. Determine the outliers by using the  $\chi^2$  statistic on the sum of the squares of the deviations along the different principal components. Do you obtain the same set of outliers in the two cases?
4. Repeat the Exercises 1, 2, and 3 with the use of EM-based modeling. How do the outliers compare with the ones obtained in the previous exercises?
5. Repeat the Exercises 1, 2 and 3 with the use of the *Breast Cancer* data set. Use the rare classes in the data set as ground truth in order to construct an ROC curve in these cases. In which case do you obtain the best results?
6. Download the 20-newsgroups data set [624]. Use the following methods to determine the outliers:
  - Use a  $k$ -nearest neighbor approach with cosine similarity.
  - Use the probabilistic modeling approach, and report the data points with the least fit as the outliers.

Do you obtain the same outliers in the two cases?

7. Repeat Exercise 6 with the Reuters-215788 data set [623].

---

---

## Chapter 9

# Time Series and Multidimensional Streaming Outlier Detection

---

---

“To improve is to change; to be perfect is to change often.”– Winston Churchill

### 9.1 Introduction

---

The temporal and streaming outlier-detection scenarios arise in the context of many applications such as sensor data, mechanical systems diagnosis, medical data, network intrusion data, newswire text posts, or financial posts. In such problem settings, the assumption of *temporal continuity* plays a critical role in identifying outliers. Temporal continuity refers to the fact that the patterns in the data are not expected to change abruptly, unless there are abnormal processes at work. It is worth noting that outlier analysis has diverse formulations in the context of temporal data, in some of which temporal continuity is more important than others. In *time-series data*, temporal continuity is immediate, and expected to be very strong. In multidimensional data with a temporal component (e.g., text streams), temporal continuity is much weaker, and is present only from the perspective of *aggregate trends*. Therefore, two different scenarios arise:

- **Abrupt change detection in time series:** These correspond to sudden changes in the trends in the underlying data stream. In such cases, the issues of temporal continuity are critical, and the outlier is defined as unusual because it exhibits a *lack of continuity* with its immediate [579] or long-term history. For example, sudden changes in time-series values (with respect to immediate history), or distinctive shapes of subsequences of the time series (with respect to long-term history) are identified as outliers. In cases where the entire time series is available offline, the advantage of hindsight may be leveraged to identify abnormal time-series values or shapes. In time-series analysis, *vertical* analysis is more important where each individual series (or dimension) is treated as a unit, and the analysis is primarily performed on this unit.

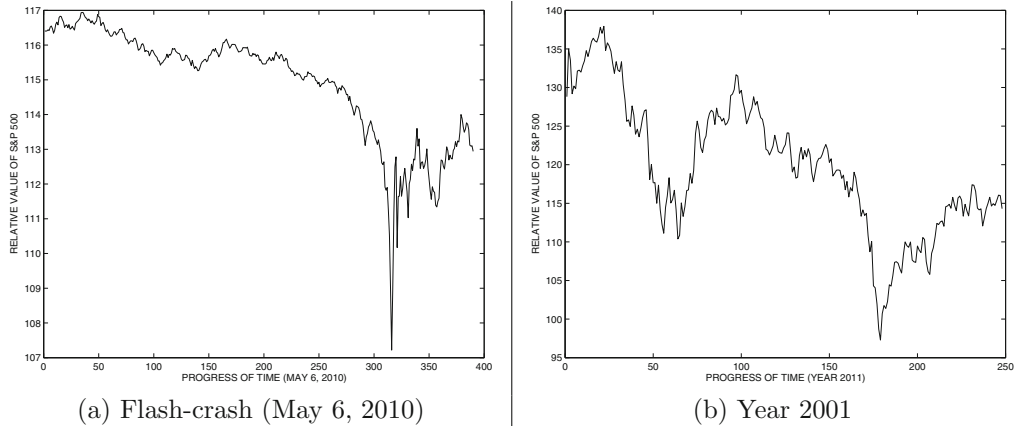


Figure 9.1: Behavior of the S&P 500 on (a) the day of the flash crash (May 6, 2010), and (b) year 2001

In the event that multiple series are available, cross-correlations may be leveraged, although they typically play a secondary role to the analysis of each individual series. This is because time-series data is *contextual*, which imposes strong temporal locality on series values.

- Novelty and change detection in multidimensional data:** In this case, the data contains individual multidimensional points that are independent of one another, and the issue of temporal continuity is much weaker as compared to time-series data. For example, in a time series derived from sensor data, two successive data values are often almost identical. On the other hand, an individual text document (multidimensional data point) in a stream of newswire articles may normally be quite different from its immediately preceding article. The data point needs to be compared to a much larger history of the documents in order to construct a robust outlier model. The anomalies in multidimensional streaming data could correspond to either *time-instants* at which *aggregate* trends have changed or novelties in *individual* data points. The latter requires the identification of incoming data points in the stream that are very different from previously seen points. Such cases are almost identical to offline outlier analysis. The temporal aspects of the stream are important only to the extent that the novelties are defined with respect to the *past* history rather than the entire data set. In such cases, all the dimensions of a record are treated as a unit, and the anomalies are identified by aggregating the temporal trends in these units. This type of analysis is *horizontal*.

For example, a first story on a particular topic in a text stream is considered a novelty outlier, while a change in the aggregate trend of the topics in the text stream is a change point outlier. It is worth noting that novelties are often *trend-setters*. Therefore, at a later stage, similar data points may no longer be considered novelties, but may become a normal part of the data. Thus, while the temporal aspects of the stream are used, they are slightly less important from an *immediate* continuity perspective. However, significant changes from aggregate trends continue to be important.

Both the aforementioned definitions are consistent with Hawkins's notion of outliers, which was introduced at the very beginning of the book.

Outliers in a time series are either *contextual* or *collective* anomalies. Outliers are contextual when the values at specific time stamps suddenly change with respect to their temporally adjacent values, whereas outliers are collective when entire time series or large subsequences within a time series have unusual shapes. For example, consider the two cases illustrated in Figures 9.1(a) and (b). The first time series illustrates the relative behavior<sup>1</sup> of the S&P 500, on May 16, 2010 which was the date of the stock market flash crash. This is a very unusual event *both* from the perspective of the deviation during the stock value drop, *and* from the perspective of the time-series shape. On the other hand, Figure 9.1(b) illustrates the behavior of the S&P 500 during the year 2001. There are two significant drops over the course of the year both because of stock-market weakness and also because of the 9/11 terrorist attacks. Although the *specific time-stamps of drop* may be considered somewhat abnormal, the *shape* of this time series is not unusual because it is frequently encountered during stock-market drops. The detection of unusual points in time or unusual shapes arises in different types of applications.

It is evident from the aforementioned example that temporal data allows multiple ways of defining anomalies. This is consistent with the observation in the first chapter: “*The more complex the data is, the more the analyst has to make prior inferences of what is considered normal for modeling purposes.*” The appropriate way of defining anomalies is highly application-dependent, and therefore it is important for the analyst to have a good understanding of the application domain at hand. For example, in the case of sensor data, it may sometimes be helpful to use deviation detection methods for noise outlier filtering, whereas in other cases, unusual shapes in a medical stream can diagnose heart conditions such as arrhythmia.

There are also some subtle differences between the *offline* and *online* (streaming) settings. In the former case, the entire history of the stream may be available for analysis, whereas only the stream up to the current time is available in the latter case. In the offline setting, the advantage of hindsight allows the discovery of better outliers with more sophisticated models.

Labels may be available to supervise the anomaly detection process in both the time-series or multidimensional outlier detection settings. In the time-series setting, the labels may be associated with time-instants, with time intervals, or they may be associated with the entire series. In the multidimensional setting, labels are associated with the individual data points. In such cases, the problem reduces to a special case of rare-class detection (cf. Chapter 7). In general, supervised methods almost always perform better than unsupervised methods because of their ability to discover *application-specific* abnormalities. The general recommendation is to always use supervision when it is available.

Even though temporal data may comprise either continuous data or discrete sequences, this chapter focuses on continuous data in order to preserve homogeneity in presentation. This is because the concept of temporal continuity is defined differently in discrete data than in continuous data. In the case of discrete data, a lack of ordering in the data values significantly influences the nature of the methods used for outlier analysis. It should be noted that a continuous series can always be discretized into symbolic data, albeit at the loss of some information. Thus, some of the methods for outlier detection are common to both types of data. The case of discrete data will be discussed separately in the next chapter. The material in the two chapters on temporal and sequence-based outlier detection is carefully organized, so as to point out the relationships among these scenarios.

---

<sup>1</sup>The tracking ETF SPY was used.

This chapter is organized as follows. In the next section, algorithms for detection of outlier *instants* in streaming time series will be presented. Typically, these methods are based on deviation-detection from predicted values at time instants. The detected anomalies are contextual outliers. In section 9.3, methods for detecting unusual shapes in time-series data will be presented. These are collective outliers. Methods for multidimensional streaming outlier detection are discussed in section 9.4. Section 9.5 presents the conclusions and summary.

## 9.2 Prediction-Based Outlier Detection in Streaming Time Series

---

The most common application of temporal outlier detection is that of detecting *deviation-based* outliers of specific time-instants with the use of regression-based forecasting models. These anomalies are contextual anomalies, because they define abnormalities at specific instants of the data, on the basis of relationships between data values at adjacent time instants. Such an approach can either be used to detect sudden changes in the underlying process, or to filter noise from the underlying streams. Deviation-based outliers in time series are very closely related to the problem of time-series forecasting, since outliers are declared on the basis of deviations from expected (or forecasted) values. This point of view is closely related to the discussion in section 7.7 of Chapter 7, which establishes a relationship between prediction and outlier detection.

In these methods, *temporal continuity* plays a significant role, since it is assumed that time-series data values are highly correlated over successive instants, and the temporal trends do not change abruptly. Deviation-based outliers use the predicted value at the next time-stamp through a variety of regression models. The correlations in a single time series, or across multiple series, may be used in order to perform the prediction. Thus, two types of correlations are used:

- **Correlations across time:** This is the same principle as temporal continuity, which is typically implemented using autoregressive modeling and forecasting. *Significant* deviations from the *expected* (i.e., forecasted) predictions are defined as outliers. Such *significant* deviations are therefore defined by violations of temporal continuity.
- **Correlations across series:** Many sensor applications result in time series that are often closely correlated with one another. For example, a bird call at one sensor will typically also be recorded by a nearby sensor. In such cases, one series can frequently be used in order to predict another. Deviations from such expected predictions can be reported as outliers.

Regression modeling techniques for *non-temporal data* are discussed in Chapter 3. This chapter will study their application in the context of temporal data, which has a number of unique characteristics in terms of data specification and modeling. While the core theory behind both domains is essentially the same, the way in which it is applied is different. The subsequent discussion assumes familiarity with the regression modeling techniques presented in section 3.2.1 of Chapter 3.

### 9.2.1 Autoregressive Models

Autoregressive models (AR) are particularly useful in the context of univariate time series. Let  $X_1 \dots X_t \dots$  be the values in the univariate time series. In the autoregressive model,

the value of  $X_t$  is defined in terms of the values in the immediately preceding window of length  $p$ .

$$X_t = \sum_{i=1}^p a_i \cdot X_{t-i} + c + \epsilon_t \tag{9.1}$$

A model that uses the preceding window of length  $p$  is referred to as an  $AR(p)$  model. The values of the regression coefficients  $a_1 \dots a_p, c$  need to be learned from the training data, which is the previous history of the time series. Here, the values of  $\epsilon_t$  are assumed to be error terms, which are uncorrelated with one another. Since these error terms represent unexpected behavior, they are natural candidates for being considered outlier scores.

A set of linear equations between the coefficients can be created by using Equation 9.1 on each time stamp in the training data. Therefore, for a time series of length  $n$ , one can extract  $(n - p)$  such linear equations. When this number is much larger than the number of variables  $(p + 1)$ , this is a severely over-determined system of equations with no exact solution. The coefficients  $a_1, \dots, a_p, c$  can be approximated with *least-squares regression* in which the squared-error of the over-determined system is minimized. The details of the solution are provided in section 3.2.1 of Chapter 3. From the point of view of the framework in section 3.2.1 of Chapter 3, one can generate an  $(n - p) \times (p + 1)$  independent-variable matrix  $D$  with  $(p + 1)$  dimensions (including the constant term) and  $(n - p)$  points for regression modeling. The rows of matrix  $D$  contain the behavioral attribute values in windows of size  $p$  along with a value of 1 appended at the end in order to handle the constant term:

$$D = \begin{bmatrix} X_1 & X_2 & \dots & X_p & 1 \\ X_2 & X_3 & \dots & X_{p+1} & 1 \\ \dots & \dots & \dots & \dots & \dots \\ X_{n-p} & X_{n-p+1} & \dots & X_{n-1} & 1 \end{bmatrix} \tag{9.2}$$

The  $(n - p)$ -dimensional column vector  $\bar{y}$  contains the dependent variables, which are the behavioral attribute values at each of the  $(n - p)$  modeled time-stamps at ticks  $p + 1 \dots n$ . In other words, the column-vector  $\bar{y}$  is defined as follows:

$$\bar{y} = \begin{bmatrix} X_{p+1} \\ X_{p+2} \\ \dots \\ X_n \end{bmatrix} \tag{9.3}$$

We would like to learn the values of  $a_1 \dots a_p, c$  that minimize the least-squares error of the following system of equations, which is a matrix representation of Equation 9.1 over the  $(n - p)$  modeled time-stamps:

$$\bar{y} \approx D \begin{bmatrix} a_p \\ a_{p-1} \\ \dots \\ a_1 \\ c \end{bmatrix} \tag{9.4}$$

Note the use of approximate equality “ $\approx$ ” because we have omitted the errors  $\epsilon_t^j$  in Equation 9.1. These errors need to be minimized by linear-regression modeling. As discussed in section 3.2.1 of Chapter 3, a matrix  $D^T D$  of size  $(p + 1) \times (p + 1)$  needs to be inverted in order to determine<sup>2</sup> the coefficients  $a_1 \dots a_p, c$  so that the least-squares error of approximation is

<sup>2</sup>Many alternatives and derivatives, such as the Yule-Walker equations [309], can also be used. Most off-the-shelf forecasting packages contain highly refined versions of these algorithms.

minimized. The corresponding solution is as follows:

$$[a_p, a_{p-1} \dots a_1, c]^T = (D^T D)^{-1} D^T \bar{y} \quad (9.5)$$

Regularization is helpful in cases where the number of time stamps is not significantly larger than  $p$ . In such cases, the matrix  $D^T D$  might not be invertible. If regularization is used, then the matrix  $(D^T D + \alpha I)$  is inverted instead of  $D^T D$  for some small value of the regularization parameter  $\alpha > 0$ . It is noteworthy that the matrices  $D^T D$  and  $D^T \bar{y}$  can be maintained incrementally as new rows are added to  $D$  and  $\bar{y}$  with the arrival of new time-stamps. However, one must still perform the inversion of the matrix  $D^T D$  in each iteration. This can be performed more efficiently in an incremental manner with the use of the *matrix-inversion lemma* [592].

In Equation 9.1, the value of  $\epsilon_t$  represents the *deviation* from the expected value, which corresponds to the outlier score of each time stamp. One can use the learned coefficients to estimate these deviations, and large absolute values correspond to anomalous time stamps. For simplified analysis, these values are assumed to be independent and identically distributed (i.i.d.) random variables from a normal distribution. This assumption can be used for hypothesis testing in converting outlier scores to outlier labels on time-stamps.

The autoregressive model can be made more robust by combining it with a moving-average model (MA Model). This model predicts subsequent values in the time series as a function of the past history of deviations. The moving-average model is defined as follows:

$$X_t = \sum_{i=1}^q b_i \cdot \epsilon_{t-i} + \mu + \epsilon_t \quad (9.6)$$

In other words, a value is predicted using the past history of *shocks* in the time series. The aforementioned model is also referred to as  $MA(q)$ . The value of  $\mu$  is the mean of the time series. The values of  $b_1 \dots b_t$  are the coefficients, which need to be learned from the data. The moving-average model is quite different from the autoregressive model, in that it relates the current value to the mean of the series and the previous history of deviations rather than the previous history of values. Note that the term  $\sum_{i=1}^q b_i \cdot \epsilon_{t-i}$  represents a linear combination of historical *shocks* or outlier scores. In this sense, the moving average model is very interesting because it expresses the current value of the series as a function of the level of unexpected behavior in the past. It is also noteworthy that the values of  $\epsilon_{t-i}$  are not a part of the observed data, but they represent the deviations from the expected values (which are themselves computed from historical deviations). Therefore, this particular system of equations is inherently nonlinear. In general, closed-form solutions cannot be found for such systems, and iterative non-linear fitting procedures are used [467].

In practice, both the previous history of deviations and values may be important for calculating expected values. The two models can then be combined with  $p$  autoregressive terms and  $q$  moving-average terms to create the following *Autoregressive Moving Average (ARMA)* model:

$$X_t = \sum_{i=1}^p a_i \cdot X_{t-i} + \sum_{i=1}^q b_i \cdot \epsilon_{t-i} + c + \epsilon_t \quad (9.7)$$

The aforementioned model is the  $ARMA(p, q)$  model. A key question here is about the choice of the parameters  $p$  and  $q$  in these models. If the values of  $p$  and  $q$  are selected to be too small, then the model will not fit the data well, and the absolute values of all noise terms will be too large to provide information about true anomalies. On the other hand, if the values of  $p$  and  $q$  are selected to be too large, then the model is likely to overfit the data. In

such cases, all noise terms will be close to 0. In general, it is good to select the values of  $p$  and  $q$  as small as possible, so that the model fits the data reasonably well. If one chooses to use larger values of  $p$  and  $q$ , then it becomes increasingly important to use regularization within the linear regression. The optimal choices of the model and parameters can be identified by minimizing the forecasting error on the observed data using leave-one-out cross-validation.

In some cases, the time series may have some persistent trends, as a result of which it may drift away from the mean. A random-walk time series would be an example of such a situation. This is referred to as the *non-stationarity* of the time series. Non-stationarity is a problem from the perspective of forecasting because the older data becomes stale over time, and it may no longer remain relevant for regression-based modeling. For example, one cannot expect to predict prices today based on the prices from a hundred years back, using a window-based regression model. This is because the prices show a clear trend, and the statistics of the price-based time series today may be very different from that a hundred years back. In such cases, the series can be de-trended by first differencing the time series before ARMA modeling. Such a modeling approach is referred to as *Autoregressive Integrated Moving Average Model (ARIMA)*. In other cases, a function such as the logarithm is applied to the series before the differencing operation. Specific details of the solution techniques for these models are beyond the scope of this book, and are discussed in detail in [467].

## 9.2.2 Multiple Time Series Regression Models

In many applications, multiple time series are available which can be used in order to perform the prediction of time-series values in a more robust way. The idea is that different time series may often contain the same information, and may sometimes contain *lag correlations*. For example, a bird-call at one sensor will also be heard at a nearby sensor, albeit with a small lag. Such lag correlations can be used in order to make more robust predictions. The standard regression-based model can be generalized to this case, by defining the variable  $X_t^j$  in terms of its past history, as well as the history of other time series. A number of common methods are discussed in the following.

### 9.2.2.1 Direct Generalization of Autoregressive Models

The basic idea in multivariate autoregressive models is to predict the values at each time-stamp with the past window of length  $p$ . The main difference from univariate regressive models is that the value at each time-stamp (for any particular series) is predicted as a linear function of all the  $d \cdot p$  values in *all* the streams in the previous window of length  $p$ . Therefore, by sliding the window over the series, one can generate a system of linear equations with  $d \cdot p$  coefficients. The coefficient  $a_i^{kj}$  represents the predictive power of the  $i$ th previous time-stamp (from current time-stamp) of series  $k$  on the current time-stamp of the  $j$ th series. Let  $t$  be the current time-stamp. Therefore, if  $X_t^1 \dots X_t^d$  represent the  $t$ th values of all the  $d$  different series, the simple autoregressive model expresses  $X_t^j$  as follows:

$$X_t^j = \left[ \sum_{k=1}^d \sum_{i=1}^p a_i^{kj} \cdot X_{t-i}^k \right] + c^j + \epsilon_t^j \quad (9.8)$$

Note that the main difference between Equations 9.1 and 9.8 is that the former expresses a time series as a linear function of its own immediate history, whereas the latter expresses a time series as a linear function of not only its own recent history but also that of the



other time series. Therefore, for  $n$  time-stamps, one can now construct a matrix  $D$  of size  $(n-p) \times [(d \cdot p) + 1]$ , where the  $i$ th row of  $D$  is the following  $(d \cdot p + 1)$ -dimensional vector:

$$[X_i^1 \dots X_{i+p-1}^1, X_i^2 \dots X_{i+p-1}^2, \dots, X_i^d \dots X_{i+p-1}^d, 1]$$

Therefore,  $D$  is an  $(n-p) \times (d \cdot p + 1)$ -dimensional matrix. Similarly, we define  $Y$  to be an  $(n-p) \times d$  dimensional matrix in which the  $j$ th column contains the values of the time-stamp for the  $j$ th time series at ticks  $p+1, \dots, n$ . In other words, the  $j$ th column of  $Y$  contains the  $(n-p)$  entries corresponding to  $X_{p+1}^j \dots X_n^j$ . Therefore, we have:

$$Y = \begin{bmatrix} X_{p+1}^1 & \dots & X_{p+1}^d \\ X_{p+2}^1 & \dots & X_{p+2}^d \\ \dots & \dots & \dots \\ X_n^1 & \dots & X_n^d \end{bmatrix} \quad (9.9)$$

One can write Equation 9.8 in matrix form as follows:

$$Y \approx D \begin{bmatrix} a_p^{11} & \dots & a_p^{1d} \\ a_{p-1}^{11} & \dots & a_{p-1}^{1d} \\ \dots & \dots & \dots \\ a_1^{11} & \dots & a_1^{1d} \\ a_p^{21} & \dots & a_p^{2d} \\ a_{p-1}^{21} & \dots & a_{p-1}^{2d} \\ \dots & \dots & \dots \\ a_1^{21} & \dots & a_1^{2d} \\ \dots & \dots & \dots \\ a_p^{d1} & \dots & a_p^{dd} \\ a_{p-1}^{d1} & \dots & a_{p-1}^{dd} \\ \dots & \dots & \dots \\ a_1^{d1} & \dots & a_1^{dd} \\ c^1 & \dots & c^d \end{bmatrix} \quad (9.10)$$

It is helpful to compare this equation with the univariate case of Equation 9.4. Note the use of approximate equality “ $\approx$ ” because we have omitted the errors  $\epsilon_t^j$  in Equation 9.8. These errors need to be minimized with least-squares optimization.

We can denote the matrix of coefficients in Equation 9.10 by  $A$ . Therefore, one needs to learn a  $(d \cdot p + 1) \times d$  matrix of coefficients  $A$ , which minimizes the least-squares error of the following relationship:

$$Y \approx DA \quad (9.11)$$

As in the previous case, the least-squares solution for  $A$  can be expressed as follows:

$$A = (D^T D)^{-1} D^T Y \quad (9.12)$$

One can perform regularization by adding  $\alpha I$  to  $D^T D$  for some small value of  $\alpha > 0$ . It is noteworthy that the matrices  $D^T D$  and  $D^T Y$  can be maintained incrementally as new rows are added to  $D$  and  $Y$  with each time-stamp, although one still needs to invert a matrix of size  $(d \cdot p + 1) \times (d \cdot p + 1)$  in each iteration. However, it is also possible to perform the inversion incrementally with the use of the matrix-inversion lemma [592].

At each time instant, a total of  $d$  different residuals denoted by  $\epsilon_t^j$  (for different values of  $j$ ), which correspond to the outlier scores of the  $d$  different series. Large absolute values

of  $\epsilon_t^j$  are reported as anomalies. Note that the scores within a series are comparable to one another but the scores across different series might not be comparable to one another if the series are not normalized to unit variance as a preprocessing step. This is because the values of  $\epsilon_t^j$  across different values of  $j$  (which might represent different physical quantities such as temperature and pressure) are likely to have different means and standard deviations. Normalization is often not possible in online settings. Therefore, the normal distribution assumption (or the  $t$ -distribution) can be used in order to determine the level of significance of the different anomalies. In general, the different values of  $\epsilon_t^j$  for a *fixed* value of  $j$  are assumed to be drawn from a normal or  $t$ -distribution.

This broad formulation can also be extended to the autoregressive moving average (ARMA) and autoregressive integrated moving-average models (ARIMA). An exponential forgetting mechanism can also be incorporated in order to give more importance to the recent history of the stream in learning the cross-stream and autoregressive coefficients. This is because the stream auto-correlations and cross-correlations may also change significantly over time.

One problem with the approach is that of increased computational complexity because of the inversion of a matrix of size  $(d \cdot p + 1) \times (d \cdot p + 1)$ . How can one use the basic principle of multivariate regression for forecasting, while keeping the complexity to a manageable level? Two such methods have been proposed in the literature:

1. One can select a subset of streams in order to perform the regression with respect to a smaller set of variables.
2. A fundamentally different approach is to use the notion of *hidden variables* to decompose the multivariate forecasting problem into a (more easily solvable) set of univariate forecasting problems.

In the following, we will discuss each of these mechanisms.

### 9.2.2.2 Time-Series Selection Methods

The *Muscles* and *Selective Muscles* techniques [592] can speed up the regression by using recursive and selection-based tricks. The *Muscles* approach is a relatively straightforward application of the least-squares model for multivariate regression. A variation known as *Recursive Least Squares* is employed to solve the regression more efficiently. The basic idea is that the solution to linear regression requires the inversion of a  $(p \cdot d + 1) \times (p \cdot d + 1)$  matrix. In a real-time application one would need to perform the inversion at each time-stamp. The work in [592] shows how to do the inversion incrementally in an efficient way with the use of the matrix-inversion lemma [592]. Refer to [592] for details of this lemma. Another difference from the standard multivariate autoregressive model is that the current values of the other series are also used for the purposes of prediction. This provides a more accurate estimation of the time stamps, and is also therefore likely to improve the accuracy of anomaly detection. It is noteworthy, however, that the current values of the other series might not always be available in a given application.

In spite of its leveraging of the matrix inversion lemma, the *Muscles* technique is slow because too many series are used. In practice, most of the series do not have predictive correlations towards one another and the presence of irrelevant series in the model can cause overfitting. The *Selective Muscles* technique therefore uses only a small subset of the series for predictive modeling. For each time series  $\overline{X^j}$ , which needs to be predicted, a subset  $S_j$  of predictor streams (satisfying  $|S_j| \ll d$ ) needs to be identified.

A greedy algorithm forms the basis of the *Selective Muscles* technique [592]. The first series to be added to  $S_j$  is the one with the highest correlation coefficient to  $\overline{X^j}$ . Subsequently, the next series to be selected minimizes the expected prediction error of the values in  $\overline{X^j}$ , when added to the current set of series in  $S_j$ . This process is continued until  $k$  series have been selected or the prediction error cannot be reduced any further. These series are then used in order to make the predictions for the  $j$ th time series (in addition to an autoregressive model on the  $j$ th series). These predictions are used for anomaly detection by finding values at time stamps which deviate significantly from expected values. The subset selection process can be performed periodically in the context of an evolving stream, in order to minimize the overhead resulting from the selection algorithm itself.

### 9.2.2.3 Principal Component Analysis and Hidden Variable-Based Models

The aforementioned models predict the values in each stream both from its own history and those of other streams. Even with the use of *Selective Muscles*, the approach can be slow. Furthermore, such an approach is sometimes not effective because of the increased number of regression coefficients, which makes the regression modeling more susceptible to noise and outliers. A specific example is presented in Figure 3.3 of Chapter 3, where a complete breakdown of regression analysis is caused by a single outlier.

As discussed in Chapter 3, PCA-based techniques are generally more robust to the presence of noise and outliers. Such PCA-based methods are able to express a large number of correlated data streams into a small number of uncorrelated data streams, which facilitates compact and noise-resistant autoregressive analysis. The most well-known technique among these methods is SPIRIT [427]. The basic idea here is that one can construct a  $d \times d$  covariance matrix between the various streams. The projections of the  $d$ -dimensional values at each time-stamp on the top- $k$  eigenvectors of this matrix provide a new set of  $k$  uncorrelated time series. These time series are also referred to as the *hidden variables*. The remaining  $(d - k)$  time series have very little variance and can be treated as constant time series, which do not need to be explicitly forecasted with an autoregressive model (since their constant values provide a trivial forecast). Furthermore, one only needs to perform the analysis on the larger eigenvectors, with a *single auto-correlation model*. Once the  $k$  values of the (hidden) time series have been predicted at a particular time-stamp, one can combine them with the remaining  $(d - k)$  (constant) values and transform the predictions to the original space. The use of auto-correlation on hidden variables provides a more robust prediction than multivariate regression. One can report the deviations from forecasted values (in each stream) as the outlier scores for the corresponding stream/time-stamp pair. This deviation can also be fitted to the normal distribution or  $t$ -distribution in order to convert the scores to binary labels. Another possibility is to create a composite deviation by computing the sum of the squares of the  $k$  components after scaling each deviation to unit variance. Since this is the sum of the squares of  $k$  normal distributions, a  $\chi^2$ -distribution can also be used to measure the significance of the composite level of deviation.

In the time-series streaming setting, the incremental implementation of PCA-based methods also requires the incremental maintenance of a covariance matrix. This can be easily achieved with the observation that all covariances can be maintained as functions of additive stream statistics. Note that the covariance between the time-series  $\overline{X^j}$  and  $\overline{X^k}$  at time  $t$  can be expressed as follows:

$$Cov(\overline{X^j}, \overline{X^k}) = \frac{\sum_{i=1}^t X_i^j \cdot X_i^k}{t} - \frac{\sum_{i=1}^t X_i^j}{t} \cdot \frac{\sum_{i=1}^t X_i^k}{t} \quad (9.13)$$

The computation above requires the maintenance of (i) the  $d$  additive sums of the values of each variable in the stream, and (ii) the  $d^2$  additive sums of the pairwise products of the time series values in the stream. This can easily be achieved in an incremental setting. Furthermore, an exponential forgetting mechanism can be incorporated in order to ensure that the co-variance matrix is based on the recent history of the stream, rather than the entire stream of data points. Therefore, the overall approach may be summarized as follows:

1. Transform the  $d$  time series into  $k \ll d$  uncorrelated hidden time series with PCA. The remaining  $(d - k)$  (hidden) time series have little variance and, therefore, their constant values<sup>3</sup> can be noted.

Let  $t$  be the current time-stamp. If  $P_k$  is the  $d \times k$  matrix with columns containing the top- $k$  eigenvectors, then the values in the  $k$  hidden time series at the  $r$ th time-stamp (for each  $r < t$ ) are contained in the  $k$ -dimensional row-vector  $[Y_r^1 \dots Y_r^k]$  using the transformation discussed in Equation 3.15 of Chapter 3:

$$[Y_r^1 \dots Y_r^k] = [X_r^1 \dots X_r^d] P_k \quad \forall r < t \quad (9.14)$$

2. For each of the  $k$  time series, use univariate auto-correlation forecasting methods to predict the value at time-stamp  $t$ . For the remaining  $(d - k)$  hidden time series, their (roughly) constant values at time  $t$  are already known. This results in the  $d$ -dimensional hidden representation of the time series in the transformed space. Note that to create the forecasts in real-world applications, one can only use values up to time  $(t - 1)$ . However, if the entire time series is available for offline analysis, it is possible to use future values as well, although the predicted value would not be technically referred to as a forecast.
3. The new  $d$ -dimensional representations are transformed back to the original space to create *forecasts* at time  $t$ . The forecasted values at time  $t$  can be expressed as follows:

$$[\hat{X}_t^1 \dots \hat{X}_t^d] = [Y_t^1 \dots Y_t^d] P_d^T \quad (9.15)$$

Note that  $P_d^T$  is the reverse transformation of  $P_d$  because we have  $P_d P_d^T = I$ . The “hat” (circumflex) symbol on top of  $\hat{X}_t^j$  denotes that it is a forecasted value.

4. Compute the absolute deviations  $|\hat{X}_t^j - X_t^j|$  as the outlier scores. One can standardize these deviations to zero mean and unit variance over a particular time series and use a  $t$ -value distribution to report specific time-stamps at which a time series shows abnormal behavior. Note that the outlier score is specific to the combination of time-stamp and time series.

One can also create a composite score across all the time series by summing up the squares of the scores across all the time series. This score provides outlier scores for time-instants, rather than time-stamp/time-series pairs. Examples of the hidden series for four precious metal exchange-traded funds (ETFs) and their hidden series are illustrated in Figure 9.2. Each of the four series can be approximately expressed as a linear combination of the top-2 hidden series. It is noteworthy that the main trends in the four series can be summarized with only two series using PCA. Furthermore, these series are uncorrelated with one another and each of them can be forecasted efficiently with the use of univariate methods. These forecasts can then be transformed to the original space in order to discover series that are behaving anomalously.

---

<sup>3</sup>For mean-centered time series these values are zero. For non-mean centered time series, one can transform the  $d$ -dimensional mean of the all time series to the new space using Equation 9.14 with all  $k = d$  eigenvectors included.

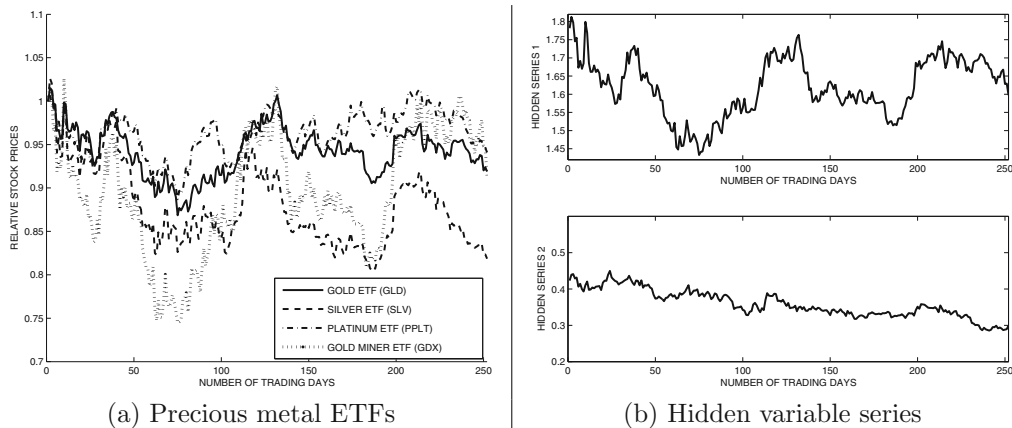


Figure 9.2: Example of four precious metal exchange traded funds (ETFs) and their top-2 hidden series [33]. Note the greater variability in the top hidden series and its ability to summarize the broader/common trends in all four series. Each of the four series can be approximately expressed as a different linear combination of the two hidden series.

### 9.2.3 Relationship between Unsupervised Outlier Detection and Prediction

The methods in this section use supervised prediction and forecasting methods for unsupervised outlier detection. The close relationship between prediction and outlier detection is not restricted to the time-series domain; the method in section 7.7 of Chapter 7 shows how one can transform a generic instance of multidimensional outlier detection to a set of supervised prediction problems [429]. Outliers are, after all, violations of the “normal” model of data dependencies. A prediction model, therefore, helps in modeling these dependencies as they apply to a specific data point. Violations of these dependencies represent violation of the model of normal data and therefore correspond to outliers.

### 9.2.4 Supervised Point Outlier Detection in Time Series

The aforementioned methods determine the significant deviations from the expected values and report them as outliers. In many cases, such deviations could have many causes that are not necessarily indicative of events of interest. For example, in the context of an environmental monitoring applications, many deviations may be result of the failure of the sensor equipment or another spurious event that causes deviations in sensor values. This may not necessarily reflect an anomaly of interest. Although anomalous events often correspond to extreme deviations in sensor-stream values, the precise causality of different kinds of deviations may be quite different. Therefore, in the context of noisy time-series data, the anomalies of interest may be embedded among a number of spurious abnormalities, which may not be of any interest to an analyst. For example, consider the case illustrated in Figure 9.3, in which we have illustrated the temperature and pressure values inside pressurized pipes containing heating fluids. Figures 9.3(a) and (b) illustrate values on two sensors in a pipe-rupture scenario. Figures 9.3(c) and (d) illustrate the values of the two sensors in a scenario where the pressure sensor malfunctions, and this results in a value of 0 at each tick. In the first case, the readings of both pressure and temperature sensors are affected by the

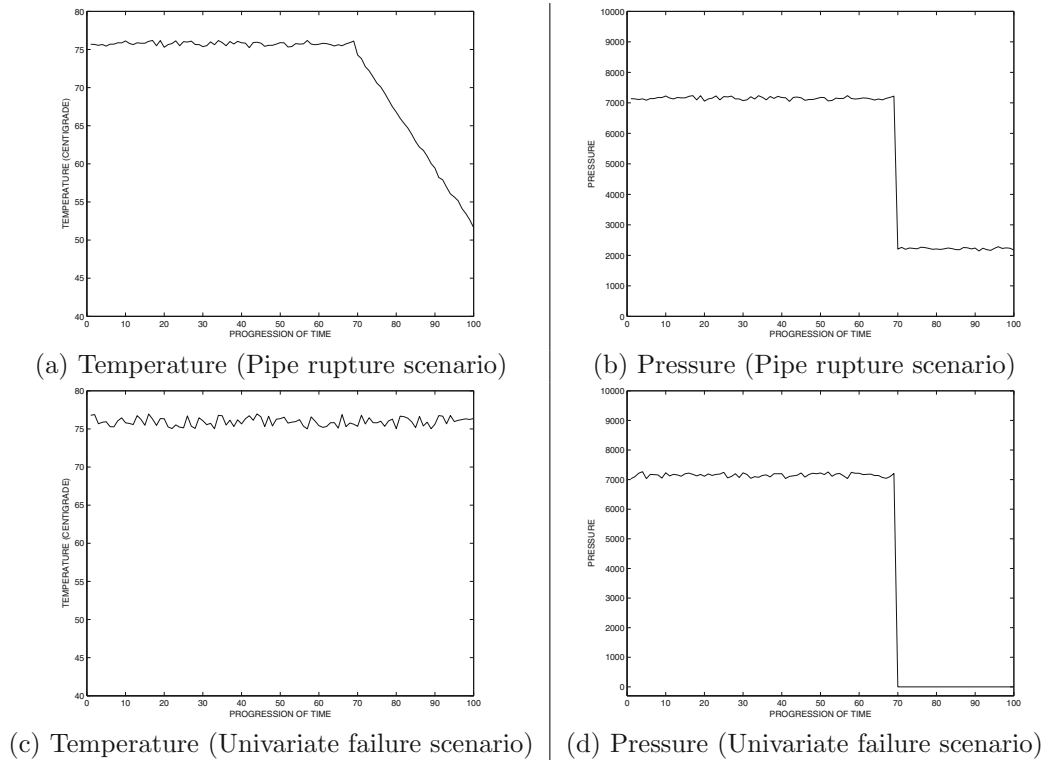


Figure 9.3: Readings of temperature and pressure sensors in various scenarios

malfunction, although the final pressure values are not zero; rather, they reflect the pressure in the external surroundings. The readings on the temperature sensor are not affected at all in the second scenario, since the malfunction is specific to the pressure sensor.

So how does one distinguish between noise and true anomalies of interest to the analyst? The time-tested method for making the approach more sensitive to analyst interest is to use *supervision* from previous examples. In the multivariate scenario, the truly anomalous events of interest may be detected only from the *differential* behavior of the deviations across different time-series data streams. In such scenarios, supervision can be very helpful in distinguishing the true anomalies from the spurious abnormalities in the time-series data stream. The approach discussed in [9] proposes a method for abnormality detection in spuriously populated data streams. It is assumed that the true events of interest are available as the *ground-truth time stamps*,  $T_1 \dots T_r$ , which are used for supervision. These are referred to as *primary abnormal events*. Note that  $T_1 \dots T_r$  might include consecutive lag periods because time-series abnormalities often show up over small lag windows.

In addition, it is possible that some other causative factors might cause secondary abnormal events. Although the training data for such spurious events can also be continuously collected in some settings [9], this may not always be possible. Nevertheless, one can still discriminate between the primary and secondary events by using the normal time stamps as baseline periods. The following description is a simplified and generalized version of the approach in [9], which does not use specific knowledge of the secondary events.

The overall process of event prediction is to create a composite alarm level from the error terms in the time-series prediction. The first step is to use a univariate time-series

prediction model in order to compute the error terms for each series/time-stamp pair. This step is identical to the methods discussed earlier in this section, and any of these methods can be used, with or without exponential forgetting factors. These error terms are then normalized to  $Z$ -values for the  $d$  streams, and their *absolute* values at the  $t$ th time-stamp are denoted by  $z_t^1 \dots z_t^d$ . Then, one can use a set of coefficients,  $\alpha_1 \dots \alpha_d$  to create the composite alarm level  $Z_t$  at time  $t$ :

$$Z_t = \sum_{i=1}^d \alpha_i \cdot z_t^i \quad (9.16)$$

This vector of *discrimination coefficients*  $\alpha_1 \dots \alpha_s$  should be learned with the objective of maximizing the differences in the alarm level between the primary events and normal time periods. The alarm level  $Q^p(\alpha_1 \dots \alpha_d)$  at the time of the primary events is as follows:

$$Q^p(\alpha_1 \dots \alpha_d) = \frac{\sum_{i=1}^r Z_{T_i}}{r} \quad (9.17)$$

We can also compute the normal alarm level  $Q^n(\alpha_1 \dots \alpha_d)$  for the entire time series of length  $n$  is as follows:

$$Q^n(\alpha_1 \dots \alpha_d) = \frac{\sum_{i=1}^n Z_i}{n} \quad (9.18)$$

Then, we would like to learn the coefficients to maximize the discrimination between the alarm at the primary events and normal periods:

$$\begin{aligned} &\text{Maximize } Q^p(\alpha_1 \dots \alpha_d) - Q^n(\alpha_1 \dots \alpha_d) \\ &\text{subject to:} \\ &\sum_{i=1}^d \alpha_i^2 = 1 \end{aligned}$$

The normalization constraint on the coefficients is necessary to prevent unbounded solutions. This objective function essentially provides the maximum discrimination between the primary events and normal periods. One can use any off-the-shelf optimization solver in order to learn  $\alpha_1 \dots \alpha_d$ . The learned coefficients are used to construct the outlier scores of each time-stamp according to Equation 9.16. In practice, the anomaly detection and learning processes are executed simultaneously, as new events are encountered. As the result, the vector of discrimination coefficients can be learned more accurately over time.

### 9.3 Time-Series of Unusual Shapes

---

Much of the work on time-series outlier detection is to determine *unusual changes or very large deviations* from the underlying series. However, certain types of deviations are based not only on the individual deviations of the data points, but also on the *shapes* of specific portions of the time series with respect to the other extracted portions. For example, consider the case of the flash-crash illustrated in Figure 9.1(a). In this case, the stock market showed very unusual behavior over *multiple time stamps* by first dropping precipitously over a very short period, and then recovering very quickly to almost the original level. This unusual behavior had a different causality from most other drops in the stock market. Therefore, the *shape* of the series is different from other large deviations. It is clear that

determining large deviations from previous time-stamps cannot *differentially* discover such anomalies, because the entire series (or subsequence) needs to be viewed from the perspective of other normal series in the database. On other words, the set of time-stamps must be viewed *collectively* in order to learn whether or not they should be considered anomalies.

The goal in such methods is to determine windows of the data (or subsequences) in which a given series behaves differently from a database of multiple sequences. Unlike the cases discussed earlier in this chapter where outliers are defined by *a single position*, such outliers correspond to multiple consecutive time stamps; the unusual co-occurrence of specific patterns at these time stamps represent outliers. Thus, the previous cases correspond to *contextual* outliers, whereas this case corresponds to *collective* outliers. Two possibilities may exist within this case of anomaly detection:

- **Full-series anomaly:** In this case, the shape of the entire series is treated as an anomaly. This shape is compared against a database of similar time series. However, in most cases, unless the database of sequences corresponds to a relatively short segment of time-stamps, the noise variations within the series will mask the anomalous shape. This is analogous to the problems of noise encountered in detecting outliers in high-dimensional data.
- **Subsequence-based anomaly:** If the time series is collected over long periods of time, then we might have a single time series that shows typical trends over shorter time-periods. Therefore, the anomalous shape is detected over small windows of the time series as deviations from these typical trends.

The distinction between these two types of problems is somewhat artificial because most of the methods for subsequence anomaly detection can also be used for full-series anomaly detection. This is because the first step in subsequence anomaly detection is to extract windows from the time series and then treat the extracted subsequences as whole series for the purpose of anomaly detection. Subsequently, various types of distance-based, probabilistic, or linear methods can be applied on the extracted windows.

There are, however, some subtle implementation differences between subsequence anomalies and whole-series anomalies. This is because one must always account for the overlap between adjacent windows when addressing subsequence anomalies. For full-series anomalies the longer length of the series is a greater concern. In all cases, it is assumed that the series has been normalized to zero mean and unit standard deviation because it is not meaningful to compare series of different means and amplitudes. Furthermore, in many models, it is assumed that all the underlying series are of the same length  $n$ .

### 9.3.1 Transformation to Other Representations

There are two types of transformations that are used to compare the various segments of a time series:

- **Numeric multidimensional transformation:** In one case, the series (or each subsequence of the series) is transformed into a multidimensional vector. The representation of each dimension corresponds to numeric coefficients in standard vector space. Proximity can be computed on this representation with the Euclidean distance. Therefore, many of the methods discussed in Chapter 4 of this book can be used in order to determine proximity-based anomalies.



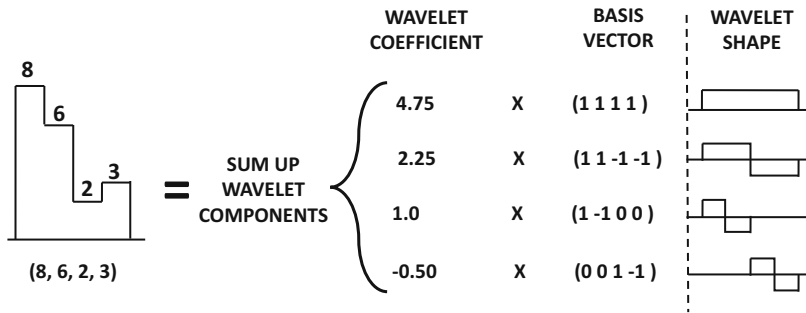


Figure 9.4: Wavelet decomposition of series of length 4. The wavelet coefficients implicitly capture long-term and short-term contextual dependencies.

- **Discrete sequence transformation:** In the second case, the series can be transformed to symbolic representations by using discretization methods. In such cases, the methods discussed in Chapter 10 can be used in order to determine unusual shapes in the time series.

This section will discuss transformations of various types.

### 9.3.1.1 Numeric Multidimensional Transformations

The simplest possible transformation would be to consider each window of length  $n$  in the time series as a multidimensional vector of length  $n$ . Other methods such as *Discrete Wavelet Transform (DWT)* and *Discrete Fourier Transform* are available for compressing the series with the multi-scale approach into numeric coefficients [439]. The most common type of wavelet is the *Haar wavelet*. The global average of the series is stored as the first coefficient. Then, for a time series of length  $n$ , one can generate the remaining  $(n - 1)$  wavelet coefficients recursively as follows:

1. Report half the difference between the averages of the first-half and second-half of the series as the first wavelet coefficient. This provides the most basic global trend between the first-half and second-half of the series.
2. Recursively compute the  $(n/2 - 1)$  wavelet coefficients for *each* of the first half and second half of the series in order to determine the remaining  $(n - 2)$  coefficients. These provide local trends in each of the two halves of the series.

Note that the number of wavelet coefficients is exactly equal to the length of the time series. It can be shown [33] that each point in the series can be reconstructed *exactly* with this set of wavelet coefficients. The time series can be exactly reconstructed as a coefficient-weighted sum of  $n$  primitive, wavelet-like, time series, each of which looks like a step function drawn on  $\{-1, 0, 1\}$ . Note that each coefficient represents the difference between the first half and second half of a particular segment of a series. The wavelet-like primitive series for a particular coefficient is defined by (i) setting the first-half of the relevant segment of the series to  $+1$ , (ii) setting the second half of the relevant segment of the series to  $-1$ , and (iii) setting all other portions of the time series to 0.

For example, consider the time series  $(8, 6, 2, 3)$ . The global average of the series is 4.75. The top-level coefficient is  $(7 - 2.5)/2 = 2.25$ . The two second-order coefficients for the

first-half and second-half of the series are  $(8 - 6)/2 = 1$  and  $(2 - 3)/2 = -0.5$ . Therefore, the wavelet representation is  $(4.75, 2.25, 1, -0.5)$ . One can reconstruct the original series as an additive sum of primitive wavelet series as follows:

$$[8, 6, 2, 3] = \underbrace{4.75 * [1, 1, 1, 1] + 2.25 * [1, 1, -1, -1]}_{\text{Long-term trends}} + \underbrace{1 * [1, -1, 0, 0] + (-0.5) * [0, 0, 1, -1]}_{\text{Short-term trends}}$$

A pictorial illustration of the decomposition of the time series into its wavelet coefficients is shown in Figure 9.4. Note that the basis vectors are mutually orthogonal and the coefficients of higher granularity capture detailed (local) trends, whereas coefficients of lower granularity capture long-term (global) trends. In this particular example, we only have two levels of granularity. Wavelet decomposition is inherently a *multiresolution* decomposition, in which the number of levels of resolution is  $O(\log_2(n))$  for a series of length  $n$ .

For a database containing  $N$  time series of length  $n$ , one can use the wavelet transform to create  $N$  new multidimensional data points with dimensionality  $n$ . Although it is common to drop small coefficients while using *single series*, it is much harder to prune small wavelet coefficients in a database of *multiple series* because a particular coefficient might be small in *most series* but a large value of that coefficient in even a single series might be relevant from the point of view of outlier detection. Therefore, all coefficients are retained.

Since the transformed representation is of the same size (i.e.,  $N \times n$ ) as the original data, what is the advantage of the wavelet representation? The key point of the wavelet transform is that one can treat the new representation *as a multidimensional data set rather than as a dependency-oriented data set because the short-term and long-term dependencies in the data are already encoded inside the wavelet coefficients*. Therefore, one can use traditional models for outlier detection of multidimensional data (like one-class SVMs or subspace methods) on the wavelet representation in a straightforward way. Other than a few distance-based models, it is generally not possible to (effectively) use off-the-shelf models with the original time-series representation (because they ignore the underlying dependencies among data values). Distance functions such as the Euclidean function are preserved, when using<sup>4</sup> the wavelet coefficients rather than the time series. This is because the new set of basis vectors are mutually orthonormal, and rotating the axis system does not affect Euclidean distances.

The discrete Fourier transform is an alternative way to perform dimensionality reduction and transformation into a new representation in which implicit dependencies are encoded in coefficients. However, in the case of the Fourier transform, the focus is on capturing *periodic* dependencies rather than local variations. Interested readers may refer to [33] for details of the Fourier transform. Which of these transformations is more effective and when should one use each? In general, a transformation is more effective when only a small number of coefficients are dominant in the representation. This tends to magnify outliers when (typically) non-dominant coefficients show large values. There are a few general guidelines in selecting the correct transformation:

1. Time series that have a significant amount of seasonality or periodicity built into them generally work well with the DFT.
2. Time series that tend to show local continuity (i.e., values that do not vary much over small ranges) generally work well with the DWT.

<sup>4</sup>The coefficients need to be normalized so that the basis vectors have unit norm. The basis vectors in Figure 9.4 are not normalized. The normalized coefficients in this case would be  $(4.75\sqrt{4}, 2.25\sqrt{4}, 1\sqrt{2}, -0.5\sqrt{2})$ .

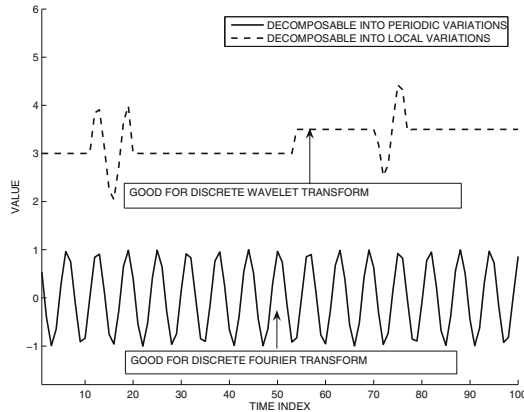


Figure 9.5: Effectiveness of different transformations in different settings

Therefore, it is often useful to have a deeper semantic understanding of the problem domain at hand before choosing the specific transformation to be used. Examples of two types of series in which the two transformations will work well are shown in Figure 9.5. A precise explanation of why such types of transformations are more relevant in various scenarios is provided in [33]. It is also possible to apply some of these transformations to multivariate series and spatial data with appropriate modifications.

Euclidean distances can be utilized on these representations to compute outlier scores. In particular, given two subsequences (or transformed representations) of length  $n$  denoted by  $A = (a_1 \dots a_n)$  and  $B = (b_1 \dots b_n)$ , the Euclidean distance between them can be computed as follows:

$$Dist(A, B) = \sqrt{\sum_{i=1}^n (a_i - b_i)^2} \quad (9.19)$$

The  $k$ th nearest-neighbor distance to a series can be used as its outlier score. A major challenge is that all these methods require  $O(N^2)$  time to compute pairwise distances, where  $N$  is the total number of subsequences. As discussed in Chapter 4, this is a problem with all such distance-based models. As in the case of multidimensional data, section 9.3.2 will discuss analogous pruning methods for improving computational efficiency.

### 9.3.1.2 Discrete Sequence Transformations

In theory, it is possible to convert continuous time series to discrete data, and then use the methods discussed in the next chapter for anomalous shape discovery. This transformation is typically performed on windows of the data, and it leads to a compressed and approximate representation of the underlying time series. Such methods can either be used for *stand-alone* anomaly detection of the discrete sequences with the methods discussed in Chapter 10, or for improving the *efficiency* of the nearest-neighbor detectors discussed above by quick approximate representations and pruning. The latter case will be discussed in the next subsection. A variety of discrete transformations are possible such as (symbolically discretized representations of) the means over specific windows, slopes over specific windows, discrete wavelet coefficients, and Fourier transform coefficients. The specific representation which is used should depend upon the application domain at hand.

A commonly used discretization technique is the *Symbolic Aggregate Approximation (SAX)* [361]. In this method, *Piecewise Aggregate Approximations (PAA)* are used in order to represent the time series. This method comprises two steps:

- **Window-based averaging:** The series is divided into windows of length  $w$ , and the average time-series value over each window is computed.
- **Value-based discretization:** The (already averaged) time-series values are discretized into a smaller number of approximately *equi-depth* intervals. The idea is to ensure that each symbol has an approximately equal frequency in the time series. The actual interval boundaries are constructed by assuming that the time-series values are distributed with a Gaussian assumption. It is to be noted that the mean and standard deviation of the (windowed) time-series values need to be computed in order to construct the Gaussian distribution. The quantiles of the Gaussian distribution are used to determine the boundaries of the intervals. Typically, the values are discretized into 3 or 4 intervals for the best results [311]. Each such equi-depth interval is mapped to a symbolic value. This creates a symbolic representation of the time series.

It is important to note that symbolic discretization does lose some information about the underlying time series. For example, the symbols provide no information about how close or far the different intervals are from one another. Nevertheless, such approximations are useful in streaming scenarios because of their simplicity, ease in construction, and their ability to facilitate the use of various pattern- and rule-based models. Such models are discussed in the next chapter.

### 9.3.1.3 Leveraging Trajectory Representations of Time Series

The case of finding unusual shapes from multivariate series is much more challenging. Here, different behavioral attributes such as temperature, pressure, or the same behavioral attribute such as the temperature may be measured *at the same instant* by different sensors. The problem of finding unusual shapes therefore needs to be very carefully defined in this case. As will be evident from the subsequent discussion, this problem maps directly to that of trajectory outlier detection, which will be discussed in detail in Chapter 11.

In multivariate temporal data, the different behavioral attributes are typically measured with the use of multiple sensors simultaneously. An example is the *Intel Research Berkeley Sensor data* described in [427], which measures different behavioral attributes over time. For example, the behavior of one of the temperature and pressure sensors at the same segment of time is illustrated in Figures 9.6(a) and (b), respectively.

Existing work on trajectory outlier detection can be leveraged to detect outlier shapes in this scenario. Even though the existing work [347] has been designed for spatial trajectories, it can also be extended to the non-spatial case with arbitrary values on the  $X$ -coordinates and  $Y$ -coordinates. In this case, it is possible to visualize the variation of the two behavioral attributes by eliminating the common time attribute, or by creating a 3-dimensional trajectory containing the time and the other two behavioral attributes. Examples of such trajectories are illustrated in Figures 9.6(c) and (d), respectively. The most generic of these trajectories is illustrated in Figure 9.6(d), which shows the simultaneous variation between all three attributes. In general, a multivariate time series with  $n$  behavioral attributes can be mapped to a  $(n+1)$ -dimensional trajectory. One issue is that when the number of behavioral attributes increases, the dimensionality of the corresponding trajectory also increases. This leads to challenges arising from the curse of dimensionality. In such cases, it may be

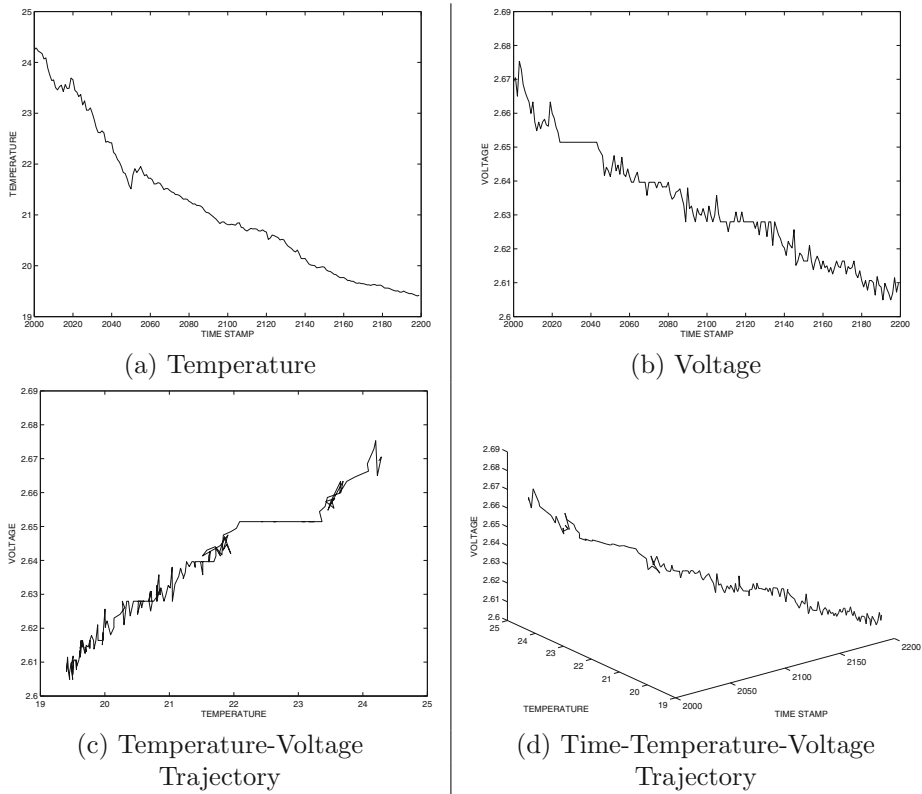


Figure 9.6: Multivariate time series can be mapped to trajectory data

better to explore subsets of behavioral attributes in order to determine outliers. This corresponds to the subspace methods discussed in Chapter 5. This is an extremely difficult problem because it combines trajectory analysis with multivariate time-series analysis, and is still an open problem in the literature.

The TROAD method [347] can be used in order to determine unusual shaped trajectories in such cases. This approach is described in detail in section 11.4 of Chapter 11. While this method was designed for 2-dimensional spatial data, a generalized version of the method can be used for higher dimensional trajectory data containing any kind of attributes. Care needs to be taken to normalize the data along each attribute to unit variance. This is particularly important in the non-spatial scenario, since the different axes of the trajectory may be drawn on very different scales.

### 9.3.2 Distance-Based Methods

The *Hotsax* approach [311] uses the standard numeric Euclidean distances introduced earlier in this section in order to define distance-based outliers. Furthermore, it uses the discrete approximations in order to perform pruning. The Euclidean distance to the  $k$ -nearest neighbor is used in order to determine the outlier score of each subsequence. In order to facilitate pruning, it is assumed that only the scores of the top- $r$  outliers need to be reported.

The outlier analysis is performed over windows of length  $p$ . These subsequences are extracted from the time series and the nearest Euclidean distances are determined. The Euclidean distance between two series is computed using Equation 9.19. Care must be taken to compare a window of values with non-overlapping windows, in order to minimize the self-similarity bias of overlapping windows [311]. Therefore, an overlapping window is not included in the computation of the  $k$ -nearest neighbor. Therefore, the basic version of the scheme (without pruning) may be described as follows:

1. Extract overlapping subsequences of length  $p$  from the time series. For a time series of length  $n$ , a total of  $(n - p + 1)$  subsequences can be extracted.
2. Report outlier scores as the  $k$ -nearest neighbor distance of the subsequences, while excluding overlapping windows in the computation.

In principle, a variety of other similarity or distance functions exist for time-series data, the most prominent of which is *Dynamic Time Warping*. Different types of similarity functions may be qualitatively more effective in different application settings, although the Euclidean distance function has the advantage that it can be efficiently computed and lends itself to an effective pruning methodology. Therefore, more general distance functions may require the use of sophisticated indexes for efficient retrieval. The reader is referred to [33, 229] for reviews of time-series similarity measures and time-series indexing.

Next, we discuss the pruning methodology that is used in the context of Euclidean distance functions. In this case, the assumption is that we only wish to discover the top- $r$  outliers (rather than report all the outlier scores). The goal of pruning is to improve the *efficiency* of the algorithm, but the same outliers will be reported as one would obtain without pruning.

A nested loop approach is used to implement the method. The algorithm examines the candidate subsequences iteratively in an outer loop. For each such candidate subsequence, the  $k$ -nearest neighbors are computed progressively in an inner loop with distance computations to other subsequences. Each candidate subsequence is either included in the current set of best  $r$  outlier estimates at the end of an outer loop iteration, or discarded via *early*

*abandonment* of the inner loop without computing the *exact* value of the  $k$ -nearest neighbor. This inner loop can be terminated early, when the currently approximated  $k$ -nearest neighbor distance for that candidate subsequence is less than the score for the  $r$ th best outlier found so far. Clearly, such a subsequence cannot be an outlier. In order to obtain the best pruning results, the subsequences need to be heuristically ordered, so that the earliest candidate subsequences examined in the outer loop have the greatest tendency to be outliers. Furthermore, the pruning performance is also most effective when the subsequences are ordered in the inner loop such that the  $k$ -nearest neighbors of the candidate subsequence are found early. It remains to explain how the heuristic orderings required for good pruning are achieved.

Pruning is facilitated by an approach that can measure the clustering behavior of the underlying subsequences. Clustering has a well-known relationship of complementarity with outlier analysis, and therefore it is useful to examine those subsequences first in the outer loop, which are members of clusters containing very few (or one) members. The SAX representation is used in order to create a simple mapping of the subsequences into clusters. The piecewise aggregate approximations of SAX are performed over windows of length  $w < p$ . Then, each window for outlier analysis examines a sequence of symbols (or words) of length  $p/w$ . Subsequences which map to unique words are more likely to be discordants than those which map to the same word. This is because the latter case corresponds to subsequences that are members of the same cluster. This observation is leveraged in [311] in order to design a pruning mechanism for outlier analysis. In the discrete case, since the number of distinct words is often limited, a trie data structure can be used in order to maintain the counts of the distinct words in the subsequences, and identify the more unusual ones. Furthermore, the identities of the set of actual subsequences to which these series map to are also maintained in this data structure. Subsequences that are anomalous can be identified by words with low frequency count. This provides an *ordering* in which to examine the different candidate subsequences. For each candidate subsequence, those subsequences which map to the same word may be considered first for computing the nearest neighbor distances in order to provide quick and tight upper bounds on the nearest neighbor distances. As these distances are computed one by one, a tighter and tighter upper bound on the nearest neighbor distance is computed. *A candidate can be pruned, when an upper bound on its nearest neighbor distance is guaranteed to be smaller (i.e., more similar) than the  $r$ th best outlier distance found so far.* Therefore, for any given series, it is not necessary to determine its exact nearest neighbor by comparing to all subsequences. Rather, early termination of the inner loop is often possible during the computation of the nearest neighbor distance. This forms the core of the pruning methodology used in [311], and is similar in principle to one of the pruning methodologies used in multidimensional distance-based methods [456].

The SAX representation is used in order to provide a good *ordering* of candidates for outliers, as well the ordering in which to iteratively compute the nearest neighbor distance for a candidate. A good ordering of the candidates ensures that strong outliers are found early, and therefore a tight *lower bound* on the outlier score is obtained early, as the  $r$ th best outlier found *so far*. A good ordering for nearest neighbor computation of a candidate ensures that the iterative nearest neighbor computation process reaches a tight *upper bound* on the outlier score of a particular candidate early. The processing of the candidate can be terminated early, when its upper bound distances are less than the lower bounds on the  $r$ th best outlier score. The effectiveness of the pruning is dependent on the tightness of the approximations, which in turn is dependent on the quality of the ordering created by SAX-based clustering. It is relatively easy to extend this approach to multivariate time series by using a multivariate distance function.

### 9.3.2.1 Single Series versus Multiple Series

The beginning of this section introduced the problem scenario where the subsequence-based time series-based anomalies needed to be detected over a database of multiple time series of the same type. For example, one may need to determine anomalies from a database of a few thousand electrocardiogram (ECG) time series. However, in many cases, a single very long series may be available, and it may be desirable to determine unusual segments of the series from this large series. The case of a single large series is not very different from that of multiple series because one can always extract subsequences of length  $p$  in both cases. The anomalous series are discovered from this set of extracted series subsequences. Although some methods such as *Hotsax* assume a single long time series in the original problem definition, others assume multiple series. The (approximate) equivalence of the two definitions ensures that methods for one can be used for the other and vice versa.

The presence of a larger number of series makes the modeling process more robust. Furthermore, if some of these series are known to be normal instances (non-outliers), the data modeling process should use only the normal series because it provides a cleaner model. For example, in a nearest-neighbor anomaly detector, a candidate sequence should be compared only to subsequences from the normal series. It is noteworthy that the case of multiple series of the same type is different from the *multivariate* scenario, in which the different *types* of time series are available, which are synchronized by their time stamps.

### 9.3.3 Probabilistic Models

Probabilistic models use a generative process to model the generation of a set of time series. For example, one can extract a set of subsequences from a given time series and then model these time series to be generated from one of the components of this mixture. The most common generative model for clustering time series is that of a hidden Markov model (HMM). A HMM can be viewed as an analog of the mixture model discussed in section 2.4 of Chapter 2 except it allows temporal dependencies among different components of the mixture in order to generate the values on successive time-stamps. As in Chapter 2, one can compute the probabilistic fit of each subsequence with respect to the probabilistic model. Data points with low levels of fit are declared outliers. Hidden Markov models are generally more suitable for discrete sequences, although they are sometimes also used for continuous data [500]. Because of the preponderance of these models in discrete data (as opposed to continuous data), a detailed discussion of HMMs will be deferred to the next chapter on outlier detection in discrete sequences.

### 9.3.4 Linear Models

It is also possible use the linear models discussed in Chapter 3 to discover unusual shapes in the underlying time series. One can use this approach both for the univariate and the multivariate case.

#### 9.3.4.1 Univariate Series

One can repeatedly extract overlapping windows of length  $p$  from the time series of length  $n$  to create  $(n - p + 1)$  objects. Each such window of length  $p$  can be treated as a  $p$ -dimensional data point. Therefore, one can now extract a data matrix of size  $(n - p + 1) \times p$ . Once the data matrix has been extracted, a  $p \times p$  covariance matrix can be constructed on this representation in order to execute PCA. Note that the covariance matrix effectively



contains information about the auto-correlations in the series at maximum lag value of  $p$ . Deviations from the normal pattern of auto-correlations in a particular window should be viewed as a window-outlier.

In order to achieve this goal, one can determine the eigenvectors of this covariance matrix to determine a new  $p$ -dimensional representation in the transformed space. Each of these  $p$  directions is standardized to zero mean and unit variance. The squared distance of each point to the  $p$ -dimensional mean of the transformed data provides the outlier score. Note that this approach is a direct application of the Mahalanobis method discussed in section 2.3.4 of Chapter 2 (with a PCA-based interpretation provided in section 3.3.1 of Chapter 3).

As discussed in section 3.5 of Chapter 3, PCA methods are special cases of matrix factorization. One can, in fact, use any type of matrix factorization and not just PCA. In such a case, the approach can also be used for anomaly detection in incompletely specified time series. This is because the method in section 3.5 can be used for anomaly detection in an incompletely specified multidimensional data set. An incompletely specified time series maps to an incompletely specified multidimensional data set using the aforementioned transform. This type of approach can be very useful because time series are often incompletely specified in many settings. It is also noteworthy that matrix factorization provides an anomaly score for each *entry* in the matrix; this can be used in order to flag specific *positions* in the time series as outliers. Therefore, the approach can also be used to determine point outliers. However, such point outliers cannot be used in an *online* setting because anomaly scores at specific positions might be using the values of the time series at future positions. Nevertheless, the approach can still be used to flag outliers in the context of retrospective and offline analysis.

### 9.3.4.2 Multivariate Series

The approach can also be applied to multivariate time series. The main difference is in terms of *feature engineering*; in other words, the multidimensional representation is created differently. Instead of extracting  $p$ -dimensional points from a univariate time series, we now extract  $p \cdot d$ -dimensional points from the set of  $d$  different time series. For the  $d$  different time series there are  $d \cdot p$  different values in the most recent window of length  $p$ . One can treat these values as a single  $(d \cdot p)$ -dimensional data point, and slide a window across the data stream to generate  $(n - p + 1)$  such points.

These points can be used in order to create a  $d \cdot p \times d \cdot p$  covariance matrix. Each entry of this covariance matrix provides the relationship both across streams and time over the past window of length  $p$ . As a result, the covariance matrix computes the covariances not only between different time-points in the same series (auto-correlations), but also between across different time series. The value of  $p$  represents the maximum lag at which such covariances are tracked. Window-based anomalies are defined as those windows in which the correlations across different time series and time-points are different from the “usual” trends. These usual trends are summarized in the covariance matrix, and PCA can be used to extract the key hidden directions of correlation.

The remaining steps of generating the outlier scores are similar to the univariate case. The eigenvectors of this covariance matrix can be used to transform the  $(d \cdot p)$ -dimensional data points into a new space. One can standardize the  $d \cdot p$  transformed coordinates in this new space. The distance of each of these  $d \cdot p$  transformed points from the mean of all the points provides the Mahalanobis score for each window of length  $p$ . Note that this approach provides an outlier score for a window of length  $p$  *across all different series*. In other words,

if the entire set of series are *collectively* behaving unusually over a window of length  $p$ , then that window should be considered an outlier. This approach requires the computation of the eigenvectors of a covariance matrix of size  $(d \cdot p) \times (d \cdot p)$ . This can sometimes be computationally expensive.

One can also use matrix factorization as a generalization of PCA (cf. section 3.5 of Chapter 3) on the extracted multidimensional data set with dimensionality  $d \cdot p$ . As in the univariate case, the matrix factorization methodology has the advantage that it can be used for incompletely specified time series. Furthermore, specific positions in the time series can be flagged as outliers because matrix factorization associates each *entry* in the matrix with an outlier score.

### 9.3.4.3 Incorporating Arbitrary Similarity Functions

It is often overlooked that PCA makes the implicit assumption that the distances between various series are represented by the Euclidean distance function. This may not always be the case in real applications. For example, in the univariate case, one might want to use distance functions like edit distance or dynamic time-warping (DTW). Such distance functions can be addressed by using kernel PCA methods discussed in section 3.3.8 of Chapter 3. In this case, one generates the embedding *directly*<sup>5</sup> from the similarity matrices, rather than creating a basis system from the covariance matrices. Consider a setting in which  $(n - p + 1)$  multidimensional points have been extracted from the time series. The steps are as follows:

1. Construct an  $(n - p + 1) \times (n - p + 1)$  similarity matrix on the data points. In cases in which distances are available<sup>6</sup> instead of similarities, one might apply a Gaussian kernel on these distances to convert the distance values  $\delta_{ij}$  into similarity values  $s_{ij}$ .

$$s_{ij} = \exp(-\delta_{ij}^2/t^2) \quad (9.20)$$

Here,  $t$  is a parameter defining the kernel width. This results in an  $(n-p+1) \times (n-p+1)$ -dimensional similarity matrix  $S$ .

2. Extract the all strictly positive eigenvectors of this similarity matrix and standardize each of the eigenvectors to zero mean and unit variance. This provides a  $k$ -dimensional representation of each of the  $(n - p + 1)$  objects. While extracting strictly positive eigenvectors, care must be taken to exclude zero eigenvectors that appear positive because of numerical errors in the eigenvector computation. A very small threshold such as  $10^{-6}$  on the eigenvalues is usually sufficient to exclude such eigenvectors.
3. The squared distance of each point from the origin in this mean-centered representation is reported as the Mahalanobis outlier score.

For the case of multivariate series, the only difference is that the distance functions now need to be computed using multivariate dynamic time-warping. Such distance functions are discussed in [33].

<sup>5</sup>A linear basis system no longer exists in the original space because a kernel transformation is used.

<sup>6</sup>Examples include dynamic time-warping and edit distance. Note that the specific choice of the distance function (e.g., DTW) depends on the application at hand. However, not all distance functions will result in a positive semi-definite kernel. In such a case, some of the tricks discussed in section 3.3.8.3 of Chapter 3 may need to be used.

#### 9.3.4.4 Leveraging Kernel Methods with Linear Models

It is also possible to apply methods such as one-class support vector machines, as long as an appropriate kernel function can be defined to measure similarity between objects [379]. One challenge with the use of this approach is that kernels need to be positive semi-definite in order for a valid embedding to exist in the transformed space. Nevertheless, at a *heuristic* level, it is often possible to use similarity functions that do not satisfy the positive-semidefinite property to obtain reasonably good results (see section 3.3.8.3).

Kernel methods are often combined with linear models because linear models in the transformed space correspond to complex boundaries in the original space. The one-class support-vector machine (cf. section 3.4 of Chapter 3) is one example of such a linear model. Another example is the use of kernel PCA methods for arbitrary data types as discussed in section 3.3.8.3. Although such an approach has not been used widely in the literature, it remains a viable option because of the simplicity in implementation of kernel PCA methods.

### 9.3.5 Supervised Methods for Finding Unusual Time-Series Shapes

In many medical applications, characteristic pathological properties may be captured by unusual shapes of the underlying time series. In such cases, training data is often available about either the normal or the pathological behavior or both. Thus, such an approach requires the detection of unusual shapes in time series in a supervised manner. The labels may either be associated with the entire time series, with portions of the time series (subsequences). The simplest possible approach in such a scenario is develop subsequence profiles for both the normal class and the anomalous class. For a given test subsequence, a  $k$ -nearest neighbor approach may be used for classification purposes. Methods for querying and classifying data streams may be found in [171].

Feature extraction and transformation forms the core of all supervised methods for time-series classification. If the time series is represented in the form of discriminative features, it becomes much easier to classify it. One possibility is to transform the series to a discrete representation, and then use Hidden Markov Models (HMM) of Chapter 10 for the purposes of classification. A much larger number of models are available for sequence classification because of the natural ease in developing pattern-based classification models in the context of discrete data. A second possibility proposed in [408, 590], is to mine *shapelets* from the data, which are highly discriminative features for classification purposes. While many of these methods have been proposed for generic time-series classification, they can usually be generalized to the case where the classes are imbalanced. Detailed surveys on time-series classification may be found in [11, 575].

## 9.4 Multidimensional Streaming Outlier Detection

---

The previous sections discussed the case, where outliers were determined from time series based on either deviations from expected values or unusual shapes. Even when multiple correlated time series are available, each time series is processed as a unit for analysis. In traditional time-series analysis, *autocorrelation* is extremely important in the prediction and outlier computation process because of a strong assumption of temporal continuity.

On the other hand, in the case of multidimensional data, each record contains  $d$ -dimensions which form an indivisible unit. Furthermore, in the case of time series, a very high level of temporal continuity is observed in the individual series. This is not necessarily the case for multidimensional data streams, in which the temporal continuity is much

weaker. For example, in a stream of multidimensional text records, the individual frequency of an attribute in a text record cannot be reliably predicted from its immediately preceding records. On the other hand, the words present in the document can be compared at an *aggregate* level with the history of the stream in order to predict outliers. Thus, outlier analysis in multidimensional data streams is very different from outlier analysis in (possibly multivariate) time-series data because of the differences in the expected level of temporal continuity in these scenarios. The multidimensional streaming scenario is much closer to traditional methods for multidimensional outlier analysis. The only difference is the addition of a temporal component to the analysis, although this temporal component is much weaker than in the case of time-series data. In the context of multidimensional data streams, efficiency is a core concern, because the outliers need to be discovered quickly. There are two types of outliers that may arise in multidimensional data streams:

- One is based on outlier detection of individual records. For example, a first news story on a specific topic represents an outlier of this type. Such an outlier is also referred to as a *novelty* because it represents a point that was not seen before.
- The second is based on changes in the *aggregate trends* of the multidimensional data. For example, an unusual event such as a terrorist attack may lead to a burst of news stories on a specific topic. This essentially represents a higher level and aggregated outlier based on a specific time window. The second type of change point almost always begins with an individual outlier of the first type. However, an individual outlier of the first type may not always develop into an aggregate change point.

Both types of outliers will be discussed in this section. Furthermore, supervised methods for detecting rare and novel classes from multidimensional data streams will also be discussed.

### 9.4.1 Individual Data Points as Outliers

The problem of detecting individual data points as outliers is closely related to the problem of *unsupervised novelty detection*, especially when the entire history of the data stream is used. This problem is studied extensively in the text domain in the context of the problem of *first story detection* [622]. Such novelties are often trend-setters, and may eventually become a part of the normal data. However, when an individual record is declared an outlier in the context of a *window* of data points, this may not necessarily correspond to a novelty. In this context, proximity-based algorithms are particularly easy to generalize to the incremental scenario, by an almost direct applications of the algorithm to the window of data points. Numerous variations of proximity-based algorithms have been generalized to the temporal scenario in the literature.

#### 9.4.1.1 Proximity-Based Algorithms

A distance-based method to detect such outliers was proposed in [60]. The original distance-based definition of outliers is modified in the following way:

The outlier score of a data point is defined in terms of its  $k$ -nearest neighbor distance to data points in a time window of length  $W$ .

Note that this is a relatively straightforward modification of the original distance-based definition. The work in [60] proposes the STORM algorithm for distance-based outlier detection. When the entire window of data points can be maintained in main memory, it is

fairly easy to determine the outliers. On the other hand, in many interesting cases, it may not be possible to hold the entire window of data points in main memory. In such cases, the streaming scenario is more challenging because it is difficult to create efficient indexes for distance-based pruning. For this case, approximate outliers are returned because the exact determination of outliers is too expensive. Accuracy guarantees are provided for the outlier detection process. It is noteworthy that the averaged ensemble score from small subsamples of the previous window can often be more effective than an exact outlier score because of variance reduction effects. A detailed discussion of this effect is discussed in Chapter 6 on ensemble methods.

The LOF algorithm has also been extended to the incremental scenario [443]. Furthermore, the approach can handle both insertion and deletion of data points. Two steps are performed in the insertion process:

- The statistics of the newly inserted data point are computed with respect to the existing LOF model.
- The existing LOF model is updated in terms of densities, reachability distances, and the outlierness of the underlying data points. In other words, the parameters of many of the existing data points need to be updated, because they are affected by the addition of a new data point. However, not all points need to be updated, because only the locality of the new data point is affected. The work in [443] performs these updates in a judicious way, so as to efficiently determine the outliers.

Since distance-based methods are well known to be computationally expensive, many of the aforementioned methods are still quite expensive in the context of the data stream. Therefore, the complexity of the outlier detection process can be greatly improved by using a clustering-based approach.

Clustering-based methods are particularly effective when a large number of data points are available. In the context of a data stream, a sufficient number of data points are typically available in order to maintain the clusters at a very high level of granularity. *In the context of a streaming clustering algorithms, the formation of new clusters is often associated with unsupervised novelties.* Of course, this may not always be the case, when the entire history of the stream is not reflected in the limited-space summary provided by the current set of clusters. For example, the work in [28] explicitly regulates the creation of new clusters in the data stream, when an incoming data point does not lie within a specified statistical radius of the existing clusters in the data. Such data points may be considered outliers. In many cases, this is the beginning of a new trend, as more data points are added to the cluster at later stages of the algorithm. In some cases, such data points may correspond to novelties. In other cases, such points may correspond to trends that were seen a long time back, but are no longer reflected in the clusters. In either case, such data points are interesting outliers. However, it is not possible to distinguish between these different types of outliers, unless one is willing to allow the number of clusters in the stream to increase over time. The work in [322] leverages the clustering process discussed in [28] to improve the efficiency of the outlier analysis process.

This approach has also been generalized to the case of text data [29]. In these cases, it can be used in order to detect the first story [622] from a possibly new trend of more stories on the topic. Other distance-based algorithms for first story detection from text streams are discussed in section 8.6.3.1 of Chapter 8. Therefore, the reader is referred to Chapter 8 for a discussion on how such proximity-based methods for applied to first-story detection in text streams. Such methods can also be combined with window-based strategies in order to detect outliers by performing the clustering on specific chunks of the data stream [181].

### 9.4.1.2 Probabilistic Algorithms

The clustering approach discussed above can also be used in the context of probabilistic learning algorithms. As discussed in Chapter 2, probabilistic learning algorithms are generally not advisable in the context of limited data. However, in the context of data streams, a much larger amount of data is available for learning, and this is less of an issue, as long as the learning algorithm can be implemented *efficiently*. A method in [578] proposes methods for creating mixture models from mixed attribute data sets. The idea is to compute a fit of the incoming data point to the model both before and after the data point is added to the model. This is achieved with the use of an expectation-maximization algorithm.

Expectation-maximization algorithms are naturally suited to the streaming setting because of their iterative approach to the optimization process. As new data points arrive, it is relatively easy to include the adjust the parameters with these points. In addition, such methods have also been extended to first-story detection in text streams [608]. These approaches are discussed in detail in Chapter 8 of this book, and the reader is referred to that chapter.

### 9.4.1.3 High-Dimensional Scenario

The combination of the streaming scenario and high-dimensional data is particularly challenging because of the complexity of high dimensional projected clustering algorithms. Clearly, computational efficiency is a primary concern in the context of high-dimensional algorithms.

Many of the high-dimensional projected stream clustering algorithms can also be used in order to determine anomalies in the data stream. This is because many of these algorithms report outliers as side products of the clustering algorithm-[6, 139]. Data points which start new clusters in the stream can typically be reported as outliers.

A method called SPOT [604] is designed for detecting outliers in high-dimensional data streams. Unlike clustering methods, this method is designed to directly find sparse subspaces of the data. This approach uses a decaying cell based summary of the underlying data stream. This is then used in order to determine projected cell based summaries. The sparse subspaces in the underlying data stream. The data points which lie in the cells corresponding to the sparse subspaces are reported as outliers.

An approach that has rarely been explored, but holds significant promise is that of ensemble methods. For example, the variable subsampling methods and rotated bagging methods [32] are particularly designed for efficiency in large data sets and high-dimensional data sets, respectively. For a streaming data set, one can create multiple subsamples of variable size and combine with rotated bagging to provide outlier scores of data points in real time. The resulting data sets are often extremely compact, and it is possible to maintain a small history of multiple data sets. The scores across different components can be combined using any of the techniques discussed in section 6.6 of Chapter 6. Note that this approach is a meta-algorithm and it can be used in combination with virtually any base outlier detector. Furthermore, the ensemble approach is likely to provide it with numerous accuracy advantages.

## 9.4.2 Aggregate Change Points as Outliers

Numerous methods have been proposed in the literature in order to determine significant change points in a multidimensional data stream. Many change point detection techniques

have been studied independently in the database literature and the outlier detection literature. The reality is that these two areas are too closely related to be treated separately. The sudden changes in aggregate local and global trends in the underlying data are often indicative of anomalous events in the data. Many methods also provide statistical ways of quantifying the level of the changes in the underlying data stream. Therefore, we will discuss some of the significant methods for change detection, which can also be used for outlier detection. Some of these methods have also been used in the anomaly detection literature for finding significant changes in stock order data streams [372], or for finding anomalies in network data streams [334, 335, 377].

#### 9.4.2.1 Velocity Density Estimation Method

The idea in velocity density estimation [19] is to construct a density-based velocity profile of the data. This is analogous to the concept of kernel density estimation in static data sets. In kernel-density estimation [496], we provide a continuous estimate of the density of the data at a given point. The value of the density at a given point is estimated as the sum of the smoothed values of kernel functions  $K'_h(\cdot)$  associated with each point in the data set. Each kernel function is associated with a kernel width  $h$  which determines the level of smoothing created by the function. The kernel estimation  $\hat{f}(\bar{X})$  based on  $N$  data points and kernel function  $K'_h(\cdot)$  is defined as follows:

$$\hat{f}(\bar{X}) = \frac{1}{N} \cdot \sum_{i=1}^N K'_h(\bar{X} - \bar{X}_i) \quad (9.21)$$

Thus, each discrete point  $\bar{X}_i$  in the data set is replaced by a continuous function  $K'_h(\cdot)$  that peaks at  $\bar{X}_i$  and has a variance determined by the smoothing parameter  $h$ . An example of such a distribution is the Gaussian kernel with width  $h$ .

$$K'_h(\bar{X} - \bar{X}_i) = \left( \frac{1}{\sqrt{2\pi} \cdot h} \right)^d \cdot \exp\left( -\frac{\|\bar{X} - \bar{X}_i\|^2}{2h^2} \right) \quad (9.22)$$

The estimation error is defined by the kernel width  $h$  which is chosen in a data driven manner. It has been shown [496] that for most smooth functions  $K'_h(\cdot)$ , when the number of data points goes to infinity, the estimator  $\hat{f}(\bar{X})$  asymptotically converges to the true density function  $f(\bar{X})$ , provided that the width  $h$  is chosen appropriately.

In order to compute the velocity density, a temporal window  $h_t$  was used in order to perform the calculations. Intuitively, the temporal window  $h_t$  is associated with the time horizon over which the rate of change is measured. Thus, if  $h_t$  is chosen to be large, then the velocity density estimation technique provides long term trends, whereas if  $h_t$  is chosen to be small then the trends are relatively short term. This provides the user flexibility in analyzing the changes in the data over different kinds of time horizons. In addition, a spatial smoothing vector  $h_s$  is used, whose function is quite similar to the standard spatial smoothing vector which is used in kernel density estimation.

Let  $t$  be the current instant and  $S$  be the set of data points that have arrived in the time window  $(t - h_t, t)$ . We intend to estimate the rate of increase in density at spatial location  $X$  and time  $t$  by using two sets of estimates: the *forward time-slice density estimate* and the *reverse time-slice density estimate*. Intuitively, the forward time-slice estimate measures the density function for all spatial locations at a given time  $t$  based on the set of data points that have arrived in the *past* time window  $(t - h_t, t)$ . Similarly, the reverse time-slice

estimate measures the density function at a given time  $t$  based on the set of data points which will arrive in the *future* time window  $(t, t + h_t)$ . Let us assume that the  $i$ th data point in  $S$  is denoted by  $(\bar{X}_i, t_i)$ , where  $i$  varies from 1 to  $|S|$ . Then, the forward time-slice estimate  $F_{(h_s, h_t)}(X, t)$  of the set  $S$  at the spatial location  $\bar{X}$  and time  $t$  is given by:

$$F_{(h_s, h_t)}(\bar{X}, t) = C_f \cdot \sum_{i=1}^{|S|} K_{(h_s, h_t)}(\bar{X} - \bar{X}_i, t - t_i) \tag{9.23}$$

Here,  $K_{(h_s, h_t)}(\cdot, \cdot)$  is a spatiotemporal kernel smoothing function,  $h_s$  is the spatial kernel vector, and  $h_t$  is temporal kernel width. The kernel function  $K_{(h_s, h_t)}(\bar{X} - \bar{X}_i, t - t_i)$  is a smooth distribution which decreases with increasing value of  $t - t_i$ . The value of  $C_f$  is a suitably chosen normalization constant, so that the entire density over the spatial plane is one unit. Thus,  $C_f$  is defined as follows:

$$\int_{\text{All } \bar{X}} F_{(h_s, h_t)}(\bar{X}, t) \delta \bar{X} = 1 \tag{9.24}$$

The reverse time-slice density estimate is also calculated in a somewhat different way to the forward time-slice density estimate. We assume that the set of points that have arrived in the time interval  $(t, t + h_t)$  is given by  $U$ . As before, the value of  $C_r$  is chosen as a normalization constant. Correspondingly, the value of the reverse time-slice density estimate  $R_{(h_s, h_t)}(\bar{X}, t)$  is defined as follows:

$$R_{(h_s, h_t)}(\bar{X}, t) = C_r \cdot \sum_{i=1}^{|U|} K_{(h_s, h_t)}(\bar{X} - \bar{X}_i, t_i - t) \tag{9.25}$$

In this case,  $t_i - t$  is being used in the argument instead of  $t - t_i$ . Thus, the reverse time-slice density in the interval  $(t, t + h_t)$  would be exactly the same as the forward time-slice density, if time was reversed, and the data stream arrived in reverse order, starting at  $t + h_t$  and ending at  $t$ .

The velocity density  $V_{(h_s, h_t)}(\bar{X}, T)$  at spatial location  $\bar{X}$  and time  $T$  is defined as follows:

$$V_{(h_s, h_t)}(\bar{X}, T) = \frac{F_{(h_s, h_t)}(\bar{X}, T) - R_{(h_s, h_t)}(\bar{X}, T - h_t)}{h_t} \tag{9.26}$$

A positive value of the velocity density corresponds to a increase in the data density of a given point. A negative value of the velocity density corresponds to a reduction in the data density a given point. It has been shown [19] that the velocity density is directly proportional to a rate of density change at a given point with the following choice of the spatiotemporal kernel function:

$$K_{(h_s, h_t)}(X, t) = (1 - t/h_t) \cdot K'_{h_s}(X) \tag{9.27}$$

This kernel function is only defined for values of  $t$  in the range  $(0, h_t)$ . The Gaussian spatial kernel  $K'_{h_s}(\cdot)$  was used because of its well-known effectiveness [496]. Specifically,  $K'_{h_s}(\cdot)$  is the product of  $d$  identical Gaussian kernel functions, and  $h_s = (h_s^1, \dots, h_s^d)$ , where  $h_s^i$  is the smoothing parameter for dimension  $i$ .

The velocity density is associated with a data point as well a time-instant, and therefore this definition allows the labeling of both data points and time-instants as outliers. However, the interpretation of a data point as an outlier in the context of aggregate change analysis



is slightly different from the previous definitions in this section. An outlier is defined on an aggregate basis, rather than in a specific way for that point. Since outliers are data points in regions where abrupt change has occurred, *outliers are defined as data points  $\bar{X}$  at time-instants  $t$  with unusually large absolute values of the local velocity density*. If desired, a normal distribution or Student's  $t$ -distribution could be used to determine the extreme values among the absolute velocity density values. Thus, the velocity density approach is able to convert the multidimensional data distributions into a quantification, which can be used in conjunction with extreme-value analysis.

It is important to note that the data point  $\bar{X}$  is an outlier only in the context of *aggregate* changes occurring in its locality, rather than its own properties as an outlier. In the context of the news-story example, this corresponds to a news story belonging to a particular burst of related articles. Thus, such an approach could detect the sudden emergence of local clusters in the data, and report the corresponding data points in a timely fashion. Furthermore, it is also possible to compute the aggregate absolute level of change (over all regions) occurring in the underlying data stream, by computing the average *absolute* velocity density over the entire data space by summing the changes at sample points in the space [19]. Time instants with large values of the aggregate velocity density may be declared outliers.

#### 9.4.2.2 Statistically Significant Changes in Aggregate Distributions

A different way to characterizing aggregate changes in multidimensional data streams would be to estimate the aggregate distributions in these time windows. Significant changes in these time windows can be reported as the unusual changes in the data stream. We note that the velocity-density method also estimates the aggregate distributions with the use of kernel-density estimation. However, in the context of change detection, it is also sometimes useful to be able to perform statistical tests directly on the underlying distributions in order to determine significant change points.

The work in [315] proposes a non-parametric framework for change detection in data streams. The key contribution in the work is a definition of the distance between two probability distributions. Generalizations of the Wilcoxon and Kolmogorov-Smirnoff tests are used in order to determine the significant change points. The work is, however, proposed for the case of 1-dimensional data streams, and the generalization to higher dimensions is not discussed.

The work in [159] is a bit more general, in that it can address multidimensional data streams effectively. Furthermore, it can be applied to different data types, because the computational aspects of the problem are different from the type of the underlying data. Since change detection is essentially relevant to the concept of finding distances between distributions, a very general way of representing this distance is the relative entropy from information theory. This is also known as the Kullback-Leibler (or KL) distance.

The broad principle is as follows. Given a set of data that should be fit to a family of distributions, the maximum likelihood estimator is the one that minimizes the KL-distance to the true distribution. This is a very general form of many other kinds of change analysis. For example, the  $t$ -test is equivalent to the KL-distance between two normal distributions. The KL-distance also allows for better intuitive interpretability, and is therefore particularly appealing for outlier detection. The definition of the KL-distance is also independent of the specific dimensionality or the type of the data representation itself. Therefore, this abstracts out the modeling of the change from the computational process. Details of this approach are presented in [159].

### 9.4.3 Rare and Novel Class Detection in Multidimensional Data Streams

An interesting case is when supervision is available in order to guide the temporal outlier-detection process. In such cases, class labels are attached to the individual data points. Many different types of supervised outliers may be of interest in such scenarios. The supervised scenario is a very rich one in the temporal domain, because *different kinds of temporal and frequency-based aspects of the classes could correspond to outliers*. These could correspond to novel class outliers, rare-class outliers, or infrequently recurring class outliers. Thus, a combination of methods for concept drift analysis, outlier detection, and rare-class detection may need to be used to determine interesting outliers in the streaming scenario. Such scenarios arise often in applications such as intrusion detection in which some known intrusions may be labeled, but new intrusions may also arise over time. Therefore, it is critical for the anomaly detection algorithm to use a combination of supervised and unsupervised methods to detect outliers.

The different types of outliers in such scenarios are as follows:

- **Rare-class outliers:** The identification of such outliers is similar to that in the static supervised scenario, except that it needs to be done more efficiently in the streaming setting. In such cases, a small fraction of the records may belong to a rare class, but they may not necessarily be distributed in a non-homogenous way from a temporal perspective. Although some amount of concept drift may also need to be accounted for, rare-class variations of stream classification algorithms may be used.
- **Novel-class outliers:** These classes were not encountered earlier in the data stream. Therefore, they may not be reflected in the training model at all. Eventually, such classes may become a normal part of the data over time. This scenario is somewhat similar to semi-supervised outlier detection in the static scenario, although the addition of the temporal component brings a number of challenges associated with it.
- **Infrequently recurring class outliers:** These are classes that have not been encountered for a while, but may reappear in the stream. Such classes are different from the first type of outliers, because they arrive in *temporally rare bursts*. Since most data stream classification algorithms use some form of discounting in order to address concept drift, they may sometimes completely age out information about old classes. Such classes cannot be distinguished from novel classes, if the infrequently recurring classes are not reflected in the training model. Therefore, issues of model update and discounting are important in the detection of such classes. This type of outlier was first proposed in [391].

We discuss each of the aforementioned cases below.

#### 9.4.3.1 Detecting Rare Classes

Numerous classifiers are available for the streaming scenario [10], especially in the presence of concept drift. For detecting *rare classes*, the only change to be made to these classifiers is to add methods that are intended to handle the *class imbalance*. Such changes are not very different from those of addressing class imbalance in the static context. The reader is referred to Chapter 7 for a detailed description of these methods and also to the bibliography section of the current chapter for references to specific methods in the streaming context. Since the broad principles of detecting rare classes do not change very much between the

static and dynamic scenario, the discussion in this section will focus on the other two types of outliers.

#### 9.4.3.2 Detecting Novel Classes

Detecting novel classes is also a form of *semi-supervision*, because models are available about many of the other classes, but not the class that is being detected. Furthermore, the problem is often encountered in an online setting. Important special cases include the first story detection setting [29, 608, 622] discussed in Chapter 8, and the streaming outlier-detection setting [28] in which no labels are available at all. In the latter case, the models are created in an unsupervised way with the use of clustering or other unsupervised methods. The work in [391, 392] addresses the problem of novel class detection in the streaming scenario with the use of labels on a subset of the classes.

Much of the traditional work on novel class detection [388] is focused only on finding novel classes that are different from the current models. However, this approach does not distinguish between the different novel classes that may be encountered over time. A more general way of understanding the novel class detection problem is to view it as a combination of supervised (classification) and unsupervised (clustering) models. Thus, as in unsupervised novel class detection models such as first story detection [29, 622], the degree of cohesion between the test instances of a novel class is important in determining whether they belong to the same novel class or not. The work in [391, 392] combines both supervised and semi-supervised models by the following methods:

- Maintaining a supervised model of the classes available in the training data as an ensemble of classification models.
- Maintaining an unsupervised model of the (unlabeled) novel classes received so far as cohesive groups of tightly knit clusters.

When a new test instance is received, the classification model is first applied to it to test whether it belongs to a currently existing (labeled) class. If this is not the case, it is tested whether it naturally belongs to one of the novel classes. The relationship of the test instance to a statistical boundary of the clusters representing the novel classes is used for this purpose. If neither of these conditions hold, it is assumed that the new data point should be in a novel class of its own. Thus, this approach combines supervised and unsupervised methods for novel class detection in a flexible way.

#### 9.4.3.3 Detecting Infrequently Recurring Classes

Many outliers in real applications often arrive in *infrequent temporal bursts*. Many classifiers cannot distinguish between novel classes and rare classes, especially if the old models have aged out in the data stream. Therefore, one solution is to simply report a recurring outlier as a novel outlier.

This is, however, not a very desirable solution because novel-class detection is a semi-supervised problem, whereas the detection of recurring classes is a fully supervised problem. Therefore, by remembering the distribution of the recurring class over time, it is possible to improve the classification accuracy. The second issue is related to computational and resource efficiency. Since novel class detection is computationally and memory intensive, it is inefficient to treat recurring classes as novel classes. The work in [391] is able to identify the recurring classes in the data, and is also able to distinguish between the different types of recurring classes, when they are distinct from one another by examining their relationships

in the feature space. For example, two types of recurring intrusions in a network intrusion detection application may form distinct clusters in the stream. The work in [391] is able to distinguish between the different types of recurring classes as well.

## 9.5 Conclusions and Summary

---

This chapter studies the problem of outlier detection in streaming time-series data and multidimensional data streams. The nature of the outliers is very different in the two cases because time-series data requires the analysis of each series as a unit, whereas multidimensional data requires the analysis of each multidimensional point as a unit. Different types of outliers can also be defined in time-series data, depending on whether it is desirable to identify deviating points in the series, or whether it is desirable to identify unusual shape subsequences. For the case of multidimensional data, individual data points can be classified as novelties, or aggregate change points in the data may be defined as outliers. In cases where some of the labels are available, supervised and unsupervised methods can be combined for effective novel and rare class detection. Thus, the area of temporal outlier detection provides a wide variety of different problem definitions.

## 9.6 Bibliographic Survey

---

Outlier detection has been studied extensively in the context of traditional time-series data analysis [231, 232], especially from the perspective of removing noise from the underlying data [101, 129, 136, 202, 467]. Much of this work (such as Kalman Filtering [101]) has focused on the removal of noise from and smoothing of temporal time series in order to facilitate more accurate regression and prediction. Outliers are defined as deviations from predicted values in such cases. Detailed surveys on outlier detection in temporal data may be found in [231, 232].

Significant changes in the time-series trends can be detected as changes and outliers in the data [579]. This includes many traditional methods for regression modeling such as Autoregressive Modeling (AR), Autoregressive Moving Average (ARMA), and Autoregressive Integrated Moving Average (ARIMA). Methods such as principal component analysis [296] have also commonly been used in order to track correlations among multiple streams. The robustness of the prediction process is helpful for accurate outlier detection. Many methods have also been designed to speed up the regression modeling in the context of large number of data streams and real-time data [27, 278, 292, 427, 592]. An information-theoretic method for determining anomalous points in a time series was proposed in [284], where an outlier is defined as a point in a time series, whose removal results in a better histogram representation in the same storage. The storage in the two cases is compared after explicitly accounting for the space required by the outlier point. This is essentially an information-theoretic approach. Supervised methods can be used to perform discriminative analysis between different types of deviations and outliers in the time-series setting [9].

Whereas time-series outlier detection is often understood in terms of real-time change detection and deviations, the problem of finding unusual shapes in time series is an entirely different scenario. Whereas the former is related to extreme value analysis, the determination of the latter is more subtle, since it requires a careful analysis of the regularities in the series over different windows of the data. One of the earliest methods for anomaly detection in time series using ideas from immunology was proposed in [157]. In such cases, the overall

magnitude of the deviations matters less than the shape of the overall time series. The discovery of anomalies in dynamic product ratings is discussed in [228].

A variety of methods can be used in order to compute unusual shapes. The use of the Haar transform for anomaly detection in time series has been explored in [205]. The most common method [311] is the use of the Euclidean distance on the fixed windows of the time series. It has been shown in [311, 360] that this problem can be further sped up with the use of symbolic aggregate approximation. These methods have been scaled up to be disk-aware and work with terabyte-scale data sets in [587]. Methods for anomaly detection in multivariate time series are discussed in [73, 140]. Another interesting approach proposed in [76] was to determine anomalous *regimes* in time-series data, where the correlations and dependencies among the different time series have changed over time. Discrete transformations can also be used on a stand-alone basis [310] in order to perform the anomaly detection directly on the discrete representation, although this is rarely done because of the information loss associated with discretization.

Methods for finding anomalies in discrete sequences are discussed in Chapter 10. A detailed survey for the discrete case may be found in [126] and integrated perspectives in the context of both discrete and continuous data may be found in [231, 232]. Some semi-supervised and supervised methods have also been proposed for unusual shape detection in the literature. In the semi-supervised scenario, it is assumed that examples of normal time series are available. In the fully supervised scenario, examples of both normal and anomalous shapes are available [171, 289, 408, 574, 590]. The transformation to the discrete case also enables the use of supervised string-based models such as hidden Markov models (HMM) [126]. Other transformation methods include feature transformation methods, which construct relevant *shapelets* on the underlying time series [408, 590]. These are patterns that can discriminate between different classes of instances. Another well-known method is the use of distance function such as dynamic time-warping [289]. The design of effective distance functions is enables the development of proximity-based classification techniques. While most of the aforementioned techniques have been developed in the literature for the *generic* version of the classification problem, most of these methods can be easily adapted to the rare class scenario by using methods discussed in Chapter 7.

Multivariate temporal data can also be represented as trajectories. Methods for trajectory outlier detection are discussed in the next chapter. Significant *changes* in trajectory directions are useful for many applications, such as hurricane tracking [117]. In such cases, the trajectory can be treated as bivariate temporal data, and prediction-based deviation detection techniques can be applied to this representation. The works in [102, 215] determine anomalies in moving object streams in real time by examining patterns of evolution. On the other hand, the detection of anomalous trajectory *shapes* is a very different problem. The earliest methods for trajectory shape outlier detection were proposed in [319]. However, this method transforms the trajectories into point data by using a set of features describing meta-information about the trajectories. Unsupervised methods for trajectory outlier detection, which actually use the sequence information explicitly, were first investigated in [409, 347]. The work in [409] uses the Fourier transform in order to represent the trajectories in terms of the leading coefficients and find anomalies. In the second method [347], trajectories are divided into different line segments and anomalous patterns are identified in order to determine outliers. Supervised methods for anomaly detection in trajectory data may be found in [355]. These methods transform the data into discrete sequences, and a classifier is learned in order to relate the trajectories to the class labels.

Outlier detection has also been studied extensively in the context of sensor data [2, 108, 94, 204, 517, 614]. Sensor streams are one of the most common applications of anomaly

detection in temporal data [614]. Sensor data is also noisy because of errors and deviations in the data collection and transmission process. Therefore, the outlier detection problem has dual applicability to this scenario, both in terms of removing the underlying noise, and in terms of detecting unusual events from the sensor stream. The same techniques are typically used for both cases. In this context, supervision [9] can sometimes be useful in distinguishing between noise and anomalies. A number of unique technological issues arise in the context of outlier detection in sensor data, because of the large amounts of data involved. Consequently, *real-time* and *in-network* processing is important in order to minimize delays and communication costs. A detailed survey on outlier detection in sensor networks may be found in [614].

The problem of outlier detection in multidimensional data streams has also been studied extensively in the literature [28, 29, 476, 60, 63, 414, 443, 532, 608] in the context of different types of multidimensional and text data. The works in [532, 227, 571] are particularly notable for their extensions of the ideas in isolation forests to the case of streaming data. Recently, a subspace histogram technique, referred to as *RS-Hash*, has been proposed in [476] (cf. section 5.2.5 of Chapter 5). The extension to the streaming setting is referred to as *RS-Stream*. This technique averages the log-density in grid regions of varying sizes and shapes in order to provide the final score. The approach uses randomized hashing for efficiency and requires linear time. This approach is suitable for subspace outlier detection in high-dimensional data and therefore its efficiency is notable. Efficiency is particularly important in these cases because of the stream scenario. The problem of novelty detection is also closely related to clustering in the stream scenario, since the formation of new clusters corresponds to novelties in the data. Many of the aforementioned techniques use either clustering or distance-based methods to detect novelties. For example, the work in [322] uses stream-clustering algorithms [28] in order to improve the efficiency of the outlier analysis process. A method in [414] uses data editing techniques that progressively remove data points with the smallest nearest neighbor distance. This is referred to as *numerosity reduction*, which reduces the size of the data while retaining certain desirable properties for outlier detection. Recently, streaming methods for outlier detection have also been extended to the case of probabilistic data [560].

The determination of aggregate changes in data streams is another type of multidimensional outlier detection [193]. This requires the determination of whether the *aggregate distribution* of the multidimensional data has changed enough to be considered significant. Methods for characterizing the change in the form of velocity density contours may be found in [19]. These methods can also be generalized to the high-dimensional case. Other methods which uses different forms of statistical testing for change detection such as the KL-distance, Wilcoxon, and the Kolmogorov-Smirnoff test, may be found in [159, 315]. A number of other statistical methods [333, 504] use log-likelihood criteria in order to quantify the change. A martingale framework for change detection in data streams is proposed in [267]. Such change points can be very useful in determining anomalies in stock market order distributions [372] or temporal network traffic data streams [334, 335, 377]. The latter class of methods can sometimes also be used to diagnose network intrusions. Methods for aggregate change point detection in the form of correlated bursty topic patterns in coordinated text streams are proposed in [562].

A detailed survey of traditional methods for semi-supervised novelty detection may be found in [388, 389]. Methods for using support vector regression models for online novelty detection are discussed in [379]. The effectiveness of novelty detection can be enhanced by using class labels. A combination of streaming classification and clustering models are proposed in [46, 391, 392] for distinguishing between normal classes, novel classes, rare

classes, and rarely recurring classes.

## 9.7 Exercises

---

1. Develop a closed-form solution to the autoregressive model (AR) in this chapter using the relationship discussed in Chapter 3. What is the size of the matrix which needs to be inverted in order to solve this model?
2. Develop a solution to the ARMA model for deviation detection. How does the complexity compare to the simple AR model?
3. Apply each of the algorithms in Exercises 1 and 2 to detect point (time-stamp) outliers in the *Ozone Level Detection* data set of the UCI Machine Learning Repository [203], for each attribute (series) separately. How do the outliers compare for different attributes in the two cases?
4. Apply the unusual shape detection algorithm discussed in this chapter in order to detect unusual shape subsequences from the *Ozone level detection* data set of the UCI Machine Learning Repository [203]. Do the unusual shapes occur at time stamps that are in any way related to the time-stamps of unusual point deviations found in Exercises 2 and 3?
5. Apply a multivariate PCA technique across the different attributes of the *Ozone Level Data Set*, where each time-stamp is treated as a multi-attribute data point corresponding to the values over different series. Note that the time-stamps are treated independently of one another and temporal continuity is not used. Determine points of significant deviations from the regression model. Which time-stamps are the outliers?
6. Repeat Exercise 5 using windows of length  $p$  in order to generate each data point. Thus, a data point of dimensionality  $p * d$  is generated at each time-stamp for multivariate series with  $d$  attributes.
7. Use the methodology of Exercise 5, in order to create a multidimensional data set from the time-series data set. Determine outliers with:
  - A  $k$ -nearest neighbor algorithm over the entire data set.
  - A  $k$ -nearest neighbor algorithm over the segment of the data set containing only earlier time stamps.

How do the outliers found relate to each other, and to those found in Exercise 5?

8. Repeat Exercise 7 using windows of length  $p$  according to the approach in Exercise 6 rather than Exercise 5.

---

---

# Chapter 10

## Outlier Detection in Discrete Sequences

---

---

“Poets write the words you have heard before, but in a new sequence.” – Brian Harris

### 10.1 Introduction

---

The previous chapter discusses anomaly detection from the perspective of continuous time series. A related setting is one in which the individual elements at each time stamp are discrete-valued (i.e., categorical). Such discrete time-series are also referred to as *sequences*. Discrete-valued temporal scenarios arise in numerous systems diagnosis, intrusion detection, and biological applications. In some domains such as intrusion detection and systems diagnosis, the discrete sequences are caused by *temporal ordering*, whereas in other domains such as biological data, the discrete sequences are caused by *physical ordering*. Nevertheless, at a logical level, the differences in the problem definitions for the two cases are relatively minor. The primary difference is that temporal data often has a specific direction to the analysis in real scenarios (i.e., forward in time), whereas this may not be the case for data based on placement relationships. At the analytical level, the models for the two cases are different in minimal ways and typically have cross-applicability. Some examples of applications that generate discrete data sequences are as follows:

- **Systems diagnosis:** Many automated systems continuously generate data about the system state. These could correspond to UNIX system calls, aircraft system states, mechanical systems, or host-based intrusion detection systems. The last case is particularly common, and is an important research area in its own right.
- **Biological data:** Biological data typically contains sequences of amino-acids, in which anomalous subsequences can provide interesting information about unusual properties of the genome sequences and their impact on different types of genetic conditions [365].
- **User-action sequences:** Such sequences are created by user *actions* in different application domains:



- Web logs contain long sequences of visits to Web sites by individuals. It is often desired to identify interesting subsequences that are indicative of anomalous or intrusive activity.
- Customer transactions may contain sequences of buying behavior. The symbols may correspond to the identifiers of the different items that are bought. Unusual temporal patterns may correspond to changes in buying patterns [122].
- User actions on Web sites such as online banking sites are frequently logged. This is similar to the aforementioned case of Web logs, except that the logs of banking sites are often more detailed. Anomalous subsequences of actions provide insights into unusual user activity, such as an attempt to break into the system.

In order to ease further discussion, some notations will be defined. A sequence is defined as an ordered set of symbols  $a_1 a_2 \dots a_r$  drawn from the symbol set  $\Sigma = \{\sigma_1 \dots \sigma_{|\Sigma|}\}$ . In the most general form, a sequence can also be defined as an ordered set of sets  $S_1 S_2 \dots S_r$ , where each  $S_i$  is a subset of  $\Sigma$ . In this chapter, the more common (and simpler) case will be examined in greater detail, where each element  $a_i$  is drawn directly from  $\Sigma$ . The more complex case of set-based sequences will also be examined briefly.

Discrete sequences are different from continuous time-series data, because it is difficult to directly compare two different symbols from the alphabet  $\Sigma$  in terms of distances. Therefore, commonly used regression-modeling methods for deviation detection in continuous data are not easily generalizable to this case. Nevertheless, even in this case, outliers can be defined in terms of either deviations from predicted values at specific time stamps, or in terms of unusual *successive combinations of sequence values*. The key is to define an appropriate predictive or regularity model in the discrete case that is analogous to its continuous counterpart. As in the case of continuous data, outliers are of two types, depending on whether *specific positions* are considered outliers, or whether *combinations of symbols* are considered outliers.

- **Position outliers:** In position-based outliers, the values at specific positions are predicted by a model. This prediction is used in order to determine the *deviation* from the model, and predict specific *positions* as outliers. Typically, Markovian methods are used for predictive outlier detection. This is analogous to deviation-based outliers that are discovered in time-series data with the use of regression models. Unlike regression models, Markovian models are better suited to discrete data. These correspond to *contextual* outliers.
- **Combination outliers:** In combination outliers, an entire test sequence is deemed to be unusual because of the combination of symbols in it. This is because this combination may rarely occur in a sequence database (frequency-based), or its distance (similarity) to most other subsequences of similar size may be very large (small). More complex models such as hidden Markov models can also be used to model the frequency of presence in terms of generative probabilities. For a longer test sequence, smaller subsequences are extracted from it for testing, and then the outlier score of the entire sequence is predicted as a combination of these values. This is analogous to the determination of unusual shapes in time-series data. These correspond to *collective* outliers.

The rarity in the combination of values can be defined in different ways depending on the specific choice of model. The most commonly used models are as follows:

- **Distance-based:** In this case, the nearest-neighbor distance of a subsequence to other candidate subsequences in the base data is computed. This provides an outlier score for the subsequence. The score of a longer sequence can be computed by combining the outlier scores of its smaller subsequences.

For the case of shorter sequences, methods such as clustering can be used for the outlier analysis process. Such methods are analogous to distance-based techniques for outlier analysis in multidimensional data.

- **Frequency-based:** In this case, the frequency of the occurrence of different subsequences of values is compared between a test instance and the base training data. Sequences that show unusual differences in frequency distributions of subsequences between the test and training sequence are presumed to be outliers. Such methods are useful in cases in which the compared subsequence is relatively small, and the alphabet size is small. In such cases, a limited number of distinct values of the subsequence is possible, and it is meaningful to talk in terms of frequencies. As in the case of distance-based models, smaller subsequences are extracted for meaningful frequency comparisons.
- **Model-based:** In this case, a probabilistic generative model is constructed that generates the subsequences. A specific example is the *hidden Markov model*. Subsequences that have low probability of being generated by the model are predicted to be outliers. Such methods are analogous to expectation-maximization algorithms for outlier detection in multidimensional data. The advantage of such models in the discrete case is that a well-designed generative Markov model can encode both the user understanding of the sequences, and can also capture highly likely sequences that are not explicitly present in the data.
- **Transformation-based:** The sequences are transformed into a new space with the use of kernel methods. A pairwise object-to-object similarity matrix is constructed. Methods like kernel-PCA and one-class support vector machines can be used in conjunction with the resulting similarity matrix.

Supervision can be incorporated when labels are available. This requires the design of rare-class adaptations of sequence classification algorithms.

This chapter is organized as follows. Section 10.2 discusses predictive models for outlier detection of individual positions in sequences. Section 10.3 discusses methods for determining combination outliers in discrete sequences. Complex sequences correspond to set-valued symbols, and multivariate sequences. The detection of outliers in such sequences is discussed in section 10.4. Online and early outlier detection is also discussed in this section. Methods for supervised outlier detection are discussed in section 10.5. The conclusions and summary are presented in section 10.6.

## 10.2 Position Outliers

---

In continuous time-series data (Chapter 9), it is discussed how one can identify point outliers as significant deviations from *expected* values at time stamps. Thus, these methods intimately combine the problems of *forecasting* and *deviation-detection*. In the case of continuous data, regression models are utilized for forecasting. A similar principle applies to discrete sequence data, in which the discrete values at specific positions can be predicted.

The main difference is in the choice of models used for predicting *discrete* values rather than continuous ones. When a position has very low probability of matching its forecasted value, it is reported as an outlier. For example, consider a Radio-Frequency Identification (RFID) application, in which event sequences are associated with product items in a superstore with the use of semantic extraction from RFID tags. A typical (normal) event sequence might appear as follows:

PlacedOnShelf, RemovedFromShelf, CheckOut, ExitStore.

On the other hand, in a shoplifting scenario, the event sequence may be *unusually* different. An example of event sequence in the case of the shoplifting scenario is as follows:

PlacedOnShelf, RemovedFromShelf, ExitStore.

Clearly, the sequence symbol *ExitStore* is anomalous in the second case but not in the first case, because it does not depict the *expected* or *forecasted* value for that position. It is often desirable to detect such anomalous *positions* on the basis of expected values. Such anomalous positions may appear anywhere in the sequence and not necessarily in the final element of the sequence (as in the aforementioned example). The basic problem definition for position outlier detection is as follows:

**Definition 10.2.1 (Semi-Supervised)** *Given a set of training sequences  $\mathcal{D} = T_1 \dots T_N$ , and a test sequence  $V = a_1 \dots a_n$ , determine if the position  $a_i$  in the test sequence should be considered an anomaly based on its expected value.*

Some formulations do not explicitly distinguish between training and test sequences. This is because a sequence can be used for both model construction and prediction, especially when it is very long. This is analogous to the case of unsupervised models in which training and test data are not differentiated. For example, such an unsupervised formulation may be as follows:

**Definition 10.2.2 (Unsupervised)** *Given a long sequence  $V = a_1 \dots a_n$ , determine if the position  $a_i$  in the test sequence should be considered an anomaly based on its expected value.*

The differences between these two models are relatively small, because the latter model simply needs to use the sequence  $V$  in order to construct the model. For ease in discussion, the first definition will be used in this chapter.

Typically, position  $a_i$  can be predicted in temporal domains only from the positions before  $a_i$ , whereas in other domains such as biological data, both directions may be relevant. The discussion below will assume the temporal scenario, although generalization to the placement scenario (as in biological data) is achieved in a straightforward way by examining windows on both sides of the position.

In many applications such as Web logs (for anomalous Web page-access prediction), the training sequence might exist as a single long sequence. Furthermore, the training and test sequences may not be cleanly separated. In such settings, it may be desirable to use the recent history of accesses to identify anomalous positions. The models discussed in this chapter are general enough to be applicable to these variations with small modifications.

Just as regression modeling of continuous streams uses small windows of past history, discrete-sequence prediction also uses small windows of the symbols. It is assumed that the prediction of the values at a position depends upon this short history. This is known as the *short-memory property* of discrete sequences, which generally holds true across a wide variety of temporal application domains [465]:

**Definition 10.2.3 (Short Memory Property)** For a sequence of symbols  $V = a_1 \dots a_i \dots$ , the value of the probability  $P(a_i|a_1 \dots a_{i-1})$  can usually be accurately approximated as  $P(a_i|a_{i-k} \dots a_{i-1})$  for some small value of  $k$  in most application domains.

Another observation is that the influence of  $a_{i-j}$  on  $a_i$  is typically greater at smaller values of  $j$ . Intuitively, this is because recent sequence symbols are more likely to be relevant for predicting the current symbol. After the value of  $P(a_n|a_{n-k} \dots a_{n-1})$  has been estimated, a position can be flagged as an outlier when the *observed* symbol in a test sequence has very low predicted probability of occurrence.

This section will discuss two types of methods for position outlier detection, known as *rule-based* and *Markovian* models. Both Markovian and rule-based models, which are technically equivalent, exploit the *small memory* property of sequences in order to explicitly model the sequences as a set of states in a Markov chain. This equivalence is not obvious at first sight. When the size of the history used for prediction is large, the number of states in the Markovian model increases to potentially  $|\Sigma|^k$ . Rule-based models can be used to provide a pruned and incomplete heuristic description of the transition behavior implied by the Markovian model. This provides a simpler framework for understanding the behavior of the sequences. Thus, rule-based models can be considered heuristic simplifications of (more formal) Markovian models.

### 10.2.1 Rule-Based Models

The primary goal in rule-based models is to estimate the value of  $P(a_i|a_{i-k} \dots a_{i-1})$  from the training database of sequences  $\mathcal{D}$ . When the value of  $P(a_i|a_{i-k} \dots a_{i-1})$  is large, the following rule can be expressed:

$$a_{i-k} \dots a_{i-1} \Rightarrow a_i$$

The predictive probability  $P(a_i|a_{i-k} \dots a_{i-1})$  is often referred to as the *confidence* of the rule. Rules may be generated either lazily for specific test sequences, or they may be generated using pre-processing on the training data. In the latter case, a relevant subset of rules (to the specific test sequence) is identified and used for prediction.

What are the challenges in robust estimation of the rules from the training data? The main problem is the large number of possibilities for the antecedent even at small window sizes (value of  $k$ ). In fact, a particular sequence  $a_{i-k} \dots a_{i-1}$  may not even occur in a modestly sized training data set, and therefore rules containing such antecedents may not occur. When the number of occurrences is small, the corresponding rule confidence cannot be estimated accurately. Therefore, for a particular antecedent subsequence in the test sequence, it may not be possible to estimate the probability of the next forecasted symbol in a robust way. In order to improve robustness, a number of heuristic relaxations and variations of the basic probability-estimation approach have been developed:

- **Support criterion:** The rules should not be generated using only the confidence criterion. Both support *and* confidence criteria should be used. The *support* of a rule is defined by the number of times that the antecedent of the rule occurs in the training data. Only rules with large support may be relevant. This condition ensures that one does not use noisy rules that are caused by chance.
- **Variable antecedent size:** Not all relevant rules may correspond to a fixed window size. In many cases, the lengths of the antecedents of the rules may vary considerably.

- **“Don’t care” positions:** Allowing “don’t care” positions in the antecedents of the rules allows for the possibility of “noise” symbols that are not relevant to prediction. For example, a rule may be specified as follows:

$$a_1 a_2 * * a_5 \Rightarrow a_6$$

Here, the asterisk represents a “don’t care” position, which may be noisy and may not have predictive power for the symbol  $a_6$ .

Although the support-confidence criterion of frequent and sequential pattern mining [33] provides a natural framework for rule-generation, many other off-the-shelf methods are available. For example, classification rules can be generated by extracting windows from the training sequence and treating the last symbol as a class label. Many off-the-shelf classification-rule generators such as *RIPPER* are available for this purpose [148]. An example of such a rule-based model is provided in [349]. For a given test subsequence, the first *fired* rule is examined. A rule is said to be fired by a test subsequence if it matches the antecedent of the rule. If the right-hand side of the fired rule corresponds to a *different* symbol as the value in the test sequence and the rule has high confidence, then this position is flagged as an anomaly. The confidence of the rule may be used as the outlier score. Numerous off-the-shelf rule generators are available for sequence data in the literature. Refer to the bibliographic notes for a discussion of these models.

## 10.2.2 Markovian Models

These models represent the sequence-generation process with the use of transitions in a Markov chain. This is essentially a special kind of *Finite State Automaton*, in which the states are defined by a short (immediately preceding) history of the sequences generated. Such models correspond to a set of states  $A$ , which represent the different types of memory about history of the sequence. For example, in *first-order* Markov models, each state represents the symbol from the alphabet  $\Sigma$ , which is generated as the final element of the sequence being modeled. Thus, the word “first-order” refers to the fact that the memory of these models is only 1. In  $k$ th order Markov models, each state corresponds to the subsequence of the final  $k$  symbols  $a_{n-k} \dots a_{n-1}$  in the sequence being modeled. Each transition in this model corresponds to an event  $a_n$ , representing the addition of the element  $a_n$  at the end of the sequence. As a result of the addition of this element, the Markovian model transitions from state  $a_{n-k} \dots a_{n-1}$  to the state  $a_{n-k+1} \dots a_n$ . The probability of this transition is denoted by  $P(a_n | a_{n-k} \dots a_{n-1})$ . It is noteworthy that this transition probability is the same as the confidence of a rule discussed in the previous section:

$$a_{n-k} \dots a_{n-1} \Rightarrow a_n$$

This connection shows why Markovian models are essentially the same as rule-based models with a few minor differences in terms of formalization.

A Markovian model can be expressed as a set of nodes representing the states, and a set of edges representing the events, which cause movement from one state to another. Each state corresponds to the immediately preceding sequence of  $k$  states in a  $k$ th order Markov model. The probability of an edge provides the conditional probability of the corresponding event. Clearly, the order of the model encodes the memory that the model retains for the generation process. First-order models contain the least amount of retained memory. From the perspective of the (equivalent) rule-generation model, first-order models simply encode

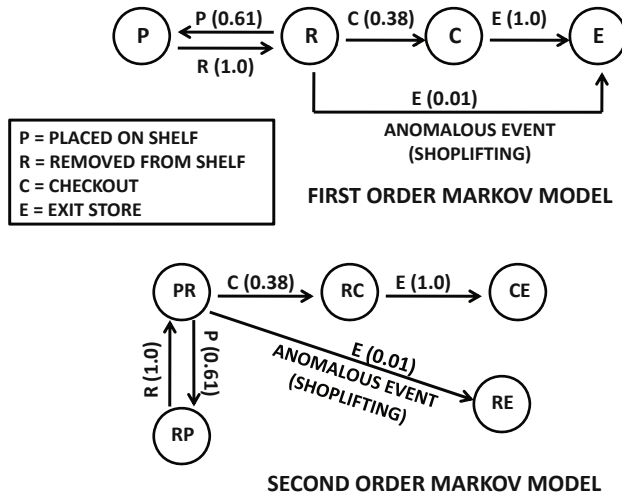


Figure 10.1: Markov model for the RFID-based shoplifting anomaly

the case, where the antecedent of the rule  $a_{n-1} \Rightarrow a_n$  is of unit length. The  $k$ th order models correspond to rules, whose antecedents are of length  $k$ . For a given test sequence, windows of size  $(k + 1)$  are extracted from the sequence. The first  $k$  symbols in it are used to determine the relevant state in the model and determine the probability of a particular event at the  $(k + 1)$ th position in the test sequence. Note that this corresponds to the firing of a rule in the equivalent rule-based models. In the event that the transition implied by the test subsequence is not defined, its probability is estimated to be 0. A simple first-order model is proposed in [588] and a  $k$ th order model is proposed in [396].

To understand how Markov models work, the previous example of tracking items with RFID tags will be revisited. An example of the different states of an item along with transitions is illustrated in Figure 10.1. Both a first-order and a second-order model have been illustrated in the figure. The edge transition probabilities are represented along with the edges, and are typically estimated from the training data. The transitions corresponding to the shoplifting anomaly are marked in both models, and they have a low probability of occurrence. This probability can also be viewed as an outlier score. This is a particularly simple example, in which a memory of one event is sufficient to completely represent the state of an item. In this case, identical results may be obtained from the first-order and second-order models in terms of the transition probabilities. This is not the case in general. For example, consider the case of a Web log in which the Markov models correspond to sequences of Web pages visited by users. In such a case, the probability distribution of the next Web page visited depends not just on the last page visited, but also on the other preceding visits by the user [169].

An observation from Figure 10.1 is that the number of states in the second-order model is larger than that in the first-order model. This is not a coincidence. In general, as many as  $|\Sigma|^k$  states may exist in an order- $k$  model. Of course, many of these subsequences may either not occur in the training data, or they may simply be invalid possibilities in a particular domain-specific context. For example, a PP state would be invalid in the example of Figure 10.1, since the same item cannot be sequentially placed twice on the shelf without removing it at least once. Higher-order models represent complex systems more accurately

if sufficient data is available. With limited data, higher-order models result in overfitting, which degrades accuracy. Furthermore, higher-order models tend to be inefficient. It is noteworthy that the transition probabilities need to be estimated from the training data, and each transition now represents a conditional on a sequence of length  $k > 1$ . Such sequences may be few when the training data is limited, as a result of which the *estimation* process may be very inaccurate. This type of over-fitting significantly impacts the overall accuracy of the model in a negative way. Furthermore, the larger number of states will also make the estimation process much slower. Therefore, a number of relaxations and variations of order- $k$  models can be constructed for greater robustness. These are analogous to the variations of rule-based methods.

- **Variable-order models:** In these methods, a state in the model might correspond to different orders, depending on its frequency in the data. Higher-order states with very low frequency can be pruned from the model, and replaced with lower-order generalizations. A method to achieve this goal with the use of *Probabilistic Suffix Trees (PST)* is presented in [524].
- **“Don’t care” subsequences:** Each state of the Markov model represents a subsequence of length  $k$ . Allowing a “don’t care” as a valid symbol in the subsequence significantly generalizes the state. This reduces the number of states, and also increases the number of training subsequences matching a state. This increases the robustness and efficiency of the approach. Such models are referred to as *Sparse Markov Transducers (SMT)* [189].

The aforementioned variations are computationally challenging, since the number of states is likely to be larger in a variable order model. Therefore, efficient data structures are required for representation and processing.

### 10.2.3 Efficiency Issues: Probabilistic Suffix Trees

It is evident from the discussion in the previous sections that the Markovian and rule-based models are equivalent, with the latter being a simpler and easy-to-understand heuristic approximation of the former. Nevertheless, in both cases, the challenge is that the number of possible antecedents of length  $k$  can be as large as  $|\Sigma|^k$ . This can make the methods rather slow, when a lookup for a test subsequence  $a_{i-k} \dots a_{i-1}$  is required in order to determine the probability of  $P(a_i | a_{i-k} \dots a_{i-1})$ . If these probability values are not organized properly, it can significantly slow down the approach, when retrieving the probability values for a particular subsequence. In many cases, the conditional probability needs to be estimated for each position in the sequence, which can be extremely slow.

Suffix trees [239] are a classical data structure, which store all subsequences in a given database. Probabilistic suffix trees (PST) represent a generalization of this structure, which also stores the conditional probabilities of generation of the next symbol for a given sequence database. For the case of order- $k$  Markov models, a suffix tree of depth at most  $k$  will store all the required conditional probability values for the  $k$ th order Markovian models, including the conditionals for all lower-order Markov models. Therefore, such a structure encodes all the required information for variable-order Markov models as well. Of course, a key challenge is that the number of nodes in such a suffix tree can be as large as  $\sum_{i=0}^k |\Sigma|^i$ . This issue is usually addressed with selective pruning.

A probabilistic suffix tree is a hierarchical data structure representing the different suffixes of a sequence. A node in the tree with depth  $k$  represents a suffix of length  $k$ , and

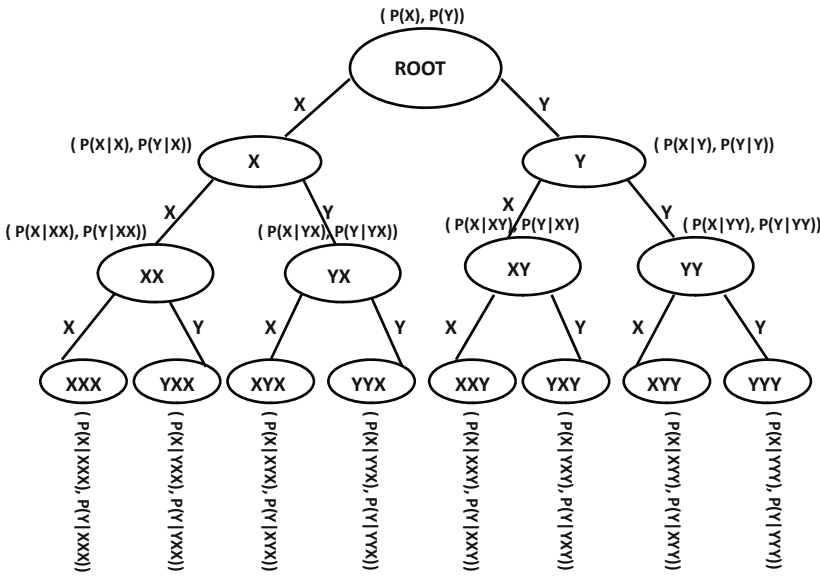


Figure 10.2: Probabilistic Suffix Tree

is therefore labeled with a sequence of length  $k$ . The parent of a node  $a_{i-k} \dots a_i$  corresponds to the sequence  $a_{i-k+1} \dots a_i$ , which is obtained by removing the *first* symbol from the sequence. Each edge is labeled with the symbol that needs to be removed in order to derive the sequence at the parent node. Thus, a path in the tree corresponds to *suffixes* of the same sequence. Each node also maintains a vector of  $|\Sigma|$  probabilities, whose elements correspond to the conditional probabilities of the generation of each symbol from  $\Sigma = \{\sigma_1 \dots \sigma_{|\Sigma|}\}$  after the sequence relevant to that node. Therefore, for a node corresponding to the sequence  $a_{i-k} \dots a_i$ , and for each  $j \in \{1 \dots |\Sigma|\}$ , the values of  $P(\sigma_j | a_{i-k} \dots a_i)$  are maintained. As discussed earlier, this corresponds to the conditional probability that  $\sigma_j$  appears immediately *after*  $a_{i-k} \dots a_i$ , after the latter sequence has already been observed. This also provides the generative probability for quantification of position outlier scores. An example of a suffix tree with symbol set  $\Sigma = \{X, Y\}$  is illustrated in Figure 10.2. The two possible symbol-generation probabilities at each node (for either  $X$  or  $Y$ ) are shown. It is also evident that a probabilistic suffix tree of depth  $k$  encodes *all* the transition probabilities for Markovian models up to order  $k$ . Therefore, such an approach can also be used for variable-order Markovian models.

The probabilistic suffix tree is pruned to improve its compactness. For example, suffixes that correspond to very low counts in the original data can be pruned. Furthermore, nodes with low generative probabilities can be pruned. The generative probability of a node, corresponding to sequence  $a_1 \dots a_n$  is approximated as follows:

$$P(a_1 \dots a_n) = P(a_1) \cdot P(a_2 | a_1) \dots P(a_n | a_1 \dots a_{n-1}) \tag{10.1}$$

For Markovian models of order  $k < n$ , the value of  $P(a_r | a_1 \dots a_{r-1})$  in the aforementioned equation is approximated by  $P(a_r | a_{r-k} \dots a_{r-1})$  for any value of  $k$  less than  $r$ . In order to create Markovian models of order  $k$  or less, it is not necessary to keep portions of the tree with depth greater than  $k$ . These observations have been used to create an efficiently pruned suffix tree for outlier analysis in [524].



Consider the sequence  $a_1 \dots a_i \dots a_n$ , in which it is desired to test whether position  $a_i$  is a position outlier. Then, it is desired to determine  $P(a_i | a_1 \dots a_{i-1})$ . It is possible that the suffix  $a_1 \dots a_{i-1}$  may not be present in the suffix tree, because it may have been pruned from consideration. In such cases, the *short memory* property is used to determine the longest suffix (denoted by  $a_j \dots a_{i-1}$ ) present in the suffix tree. The corresponding probability is estimated by  $P(a_i | a_j \dots a_{i-1})$ . Thus, the probabilistic suffix tree provides an efficient way to store and retrieve the relevant probabilities. The length of the longest path that exists in the suffix tree containing a non-zero probability estimate of  $P(a_i | a_j \dots a_{i-1})$  also provides an idea of the level of rarity of this particular sequence of events. Positions that contain only short paths preceding them in the suffix tree are more likely to be outliers. Thus, outlier scores may be defined from the suffix tree in multiple ways:

- If only short path lengths exist in the (pruned) suffix tree corresponding to a particular position in the sequence, then such a position is more likely to be an outlier.
- For the paths of lengths  $1 \dots r$ , which do exist in the suffix tree for position  $a_i$ , a combination score may be used based on the models of different orders. In some cases, only lower-order scores are combined. In general, the use of lower-order scores is preferable to ensure robustness.

One property of the suffix tree is that it can also be used to approximate the generative probability of local subsequences of the data with the use of Equation 10.1. This can be used to determine short *segments* of sequences, which are anomalies, by determining the segments with low generative probabilities. These are referred to as *combination outliers*, which will be discussed in the next section.

Position outliers can also be used for *correcting* noise in sequences by replacing particular positions with symbols that have higher generative probabilities. Such positions and symbols can be extracted from the suffix tree, and such an approach has been discussed in [465]. These methods can also potentially be used to correct grammatical errors in text sentences, where each sentence is treated as a sequence of words, and the training data contains a large number of grammatically correct sentences.

### 10.3 Combination Outliers

---

In combination outliers, the goal is to determine unusual combinations of symbols in a given sequence with respect to other sequences. There are several versions of the problem corresponding to the size of the compared sequences, and whether the base data contains multiple sequences or whether it consists of a single long sequence. Interestingly, all these different versions of the problem can be reduced to the same primitive family of sequence anomaly-detection problems, except that different *wrappers* need to be used for different scenarios. Many of these formulations are treated very differently in the literature, and are often not connected with one another, because of the diverse problem domains in which they arise. A number of selective reductions between some of these formulations are discussed in [126]. The treatment in this book is more unified because the *specific model used* (e.g., frequency-based method or distance-based method or hidden Markov model) has been cleanly separated from the methodology.

One challenge is that sequence-based outlier detection arises in rather diverse scenarios. Sequence outlier analysis can be formulated very differently, because the data may be present in many different formats, such as a single sequence, multiple sequences, long or

short sequences. However, these scenarios often turn out to be similar with the appropriate data preprocessing. The most common formulation scenarios are discussed below:

**Unsupervised versus semi-supervised:** In the unsupervised version of the problem, all anomalous sequences are determined within a database of sequences. In the semi-supervised version of the problem, an anomaly score is determined for a *test* sequence, with respect to a *training* database of *normal* sequences. At the formulation level, there is no difference between these problems except for the fact that the semi-supervised case makes a distinction between training and test data. The implicit assumption is that the training data does not contain anomalies and therefore a one-class model should be constructed on it. For discussion purposes, the semi-supervised version will be used, since the model of comparing a test sequence with a set of training sequences is more fundamental and can be generalized to the other version as a repetitively used primitive subroutine.

**Long versus short test sequence:** When the test sequence is relatively short, the sequence can be directly compared to windows of the training sequences in order to determine its *rarity*. In cases in which both the test sequences and training sequences are short, and of comparable length, multidimensional methods for anomaly detection such as the *k*-nearest neighbor method can be adapted to this case. These are referred to as *point-based* techniques.

When the test and training sequences are long, it is not meaningful to compare whole sequences. In such cases, the curse of dimensionality prevents an informative computation of distances over long sequences. Similar to subspace methods, windows of the test sequence are extracted as small subsequences. Such small subsequences will henceforth be referred to as *comparison units*. Then, the *relative behavior* of the comparison units is compared in both the training data and test sequence. If this comparison unit (or a small variation of it) is much less prevalent in the training sequence, as compared to the test sequence, it is considered an anomaly. The final anomaly score for a given test sequence can be derived as a *combination score* from all the subsequences extracted from it.

**Single long training sequence versus many training sequences:** The sequence database may contain a single long training sequence, or it may contain many training sequences. At a logical level, there is little difference between these cases because all distance-based, frequency-based, and Markov models can be derived equivalently either from a single long training sequence or multiple training sequences. This is because the models describe the behavior of only a very small window of the data corresponding to the *comparison unit*. For example, when a comparison unit is compared to training sequences in a *k*-nearest neighbor approach, the anomaly score is the *k*th nearest neighbor distance among all (similar size) subsequences in the training data, whether they are derived from the same series or multiple series. For greater generality, it will henceforth be assumed that the training data contains multiple sequences.

**Single long sequence that is undifferentiated between training and test data:** It is also possible to create difficult combinations of the aforementioned cases. A particularly common case is that of Web logs, in which a single long sequence can be extracted, and it might be desirable to determine unusual *portions* of this sequence. The additional step required for this case is to extract portions of the undifferentiated sequence as test sequences. A multi-granularity approach can be used in order to extract test sequences of different lengths, possibly in geometrically increasing sizes. Then, a *second level* of smaller subsequences is extracted from the test sequences. These subsequences correspond to the

*comparison units*. These are used to model the *relative difference* between the derived test sequences and the training sequences in terms of the smaller window-based comparison units.

**Presence or absence of domain knowledge about relevant comparison units for anomaly detection:** In all the aforementioned cases, it is assumed that the comparison unit subsequences are derived as contiguous windows from the test sequences. In many applications, specific domain knowledge may be available about important comparison units. For example, in a security application designed to detect unusual login attempts, a sequence such as *Login Password Login Password Login Password* may be very relevant for detecting interesting anomalies. In such cases, the models can be easily evaluated and combined in terms of these *domain-dependent comparison units*, rather than the units extracted from the test sequences. This is a form of supervision with domain-knowledge. In general, domain knowledge can significantly improve the quality of the overall results.

All the aforementioned variations are also relevant to time-series data, although discrete sequence data seems to exhibit a wider variation among these different possibilities in real application domains.

### 10.3.1 A Primitive Model for Combination Outlier Detection

From the aforementioned discussion, it is clear that in the most general case, a model needs to be constructed on the basis of relative comparisons between three different types of sequences: (i) the training sequences, (ii) the test sequence, and (iii) the comparison units. One of these three different types of sequences (the comparison unit) is usually extracted from the test sequence, and may not directly be a part of the input on a stand-alone basis, unless specific domain knowledge is available. However, it will still be included in the primitive formulation, since it provides a very general way to define a primitive formulation for combination outlier detection. This formulation and its minor variants are repeatedly used in various forms of sequence anomaly detection. Some notations and definitions will be used in order to distinguish between the training database, test sequence, and the comparison units.

- The training database is denoted by  $\mathcal{D}$ , and contains sequences denoted by  $T_1 \dots T_N$ .
- The test sequence is denoted by  $V$ .
- The comparison units are denoted by  $U_1 \dots U_r$ . Typically, each  $U_i$  is derived from small contiguous windows of  $V$ . In domain-dependent cases,  $U_1 \dots U_r$  may be provided by the user.

In the case of small test sequences, a single comparison unit  $U_1 = V$  may be used. Furthermore, in cases in which the test sequence is of similar (short) length as the training sequences, this reduces to the easier special case of point-anomaly detection.

This sets the stage for defining the primitive sequence anomaly detection problem.

**Definition 10.3.1 (Primitive Sequence Anomaly (Relative))** *Given a training database of discrete sequences  $\mathcal{D}$ , a test sequence  $V$ , and a comparison unit  $U_i$ , determine the anomaly score of  $U_i$  in terms of relative rarity in  $\mathcal{D}$  with respect to  $V$  with the use of model  $\mathcal{M}$ .*

The model  $\mathcal{M}$  may be a distance-based, frequency-based or hidden Markov model. Each of these will be discussed in subsequent sections. The comparison units may either be specified

by the user, or they may be extracted from small sliding windows of the test sequence. In cases, where the comparison unit  $U_i$  is extracted directly from  $V$ , the *absolute* rarity (rather than *relative* rarity) of  $U_i$  is computed with respect to the training sequences, since the comparison unit is already known to belong to the test sequence. For example, for a distance-based model, the comparison unit will have zero distance from at least one window in the test sequence  $V$ , and absolute comparisons to the training sequence will provide the same results as a relative comparison between the training and test data. In such cases, the absolute distances to windows of the training sequence are used as the anomaly score. Even in the case of frequency-based and Markov models, relative comparisons are possible, although they are not generally performed in domain-independent scenarios. Thus, the issue of *relative* surprise is more relevant in scenarios where the comparison units are provided as part of the input. For the case where the comparison units are extracted from the sequences, the primitive sequence anomaly detection problem may be restated as follows:

**Definition 10.3.2 (Primitive Sequence Anomaly (Absolute))** *Given a training database of discrete sequences  $\mathcal{D}$ , a test sequence  $V$ , and a comparison unit  $U_i$  extracted from  $V$ , determine the absolute anomaly score of  $U_i$  with respect to  $\mathcal{D}$  with the use of model  $\mathcal{M}$ .*

Multiple comparison units are extracted from  $V$  using a sliding window method, where each  $U_i$  corresponds to a contiguous window of  $V$ .

### 10.3.1.1 Model-Specific Combination Issues

For each of the two formulations introduced in the previous section, the anomaly score of the model is computed with respect to a comparison unit. However, multiple comparison units are extracted from a test sequence. As a post-processing step, it is needed to compute the overall anomaly score of the test sequence by combining the results from different comparison units. The precise method for combining the different scores will be discussed along with the model in the following sections. These methods are similar in spirit and principle, although differences may exist at the detailed level because of the differences in how the anomaly score for each comparison unit is quantified by the different models.

### 10.3.1.2 Easier Special Cases

An important special case is one in which the test sequence and training sequences are short, and of comparable size. In such cases, extraction of comparison units becomes irrelevant. Furthermore, it is much easier to compute similarity measures between pairs of *short* sequences in a more robust way, because the curse of dimensionality becomes less relevant. Such cases can be reduced to point-anomaly detection. Some of the models discussed below can also be applied to these simplified cases. Since this is a common special case, a separate formulation is defined for this problem.

**Definition 10.3.3 (Whole Sequence Anomaly (Semi-supervised))** *Given a database  $\mathcal{D}$  of short training sequences  $\{T_1 \dots T_N\}$ , and a test sequence  $V$  of comparable size, determine the anomaly score of  $V$  with respect to  $\mathcal{D}$  with the use of model  $\mathcal{M}$ .*

The unsupervised variation of the above problem is very similar, and can be stated as follows:

**Definition 10.3.4 (Whole Sequence Anomaly (Unsupervised))** *Given a database  $\mathcal{D}$  of short training sequences  $\{T_1 \dots T_N\}$ , determine all anomalous sequences with the use of model  $\mathcal{M}$ .*

As discussed earlier in this chapter, the unsupervised and semi-supervised versions are almost identical, and therefore do not need separate discussion. Therefore, both of these cases will be discussed in a unified way.

Note that both formulations above are missing the comparison unit. Short sequences do not require the extraction of smaller comparison units for robust distance computations in anomaly detection. In the following sections, the problem of whole sequence anomaly detection will also be discussed. Although certain models such as distance-based models and hidden Markov models can be used effectively in this case, frequency-based models cannot be used in such scenarios. This is because frequency-based models are dependent on the repetitive frequencies of short comparison units within long sequences. This is not possible in scenarios where all sequences are short.

### 10.3.1.3 Relationship between Position and Combination Outliers

Although the most popular and robust methods for detecting combination outliers use window-based comparison units, it is sometimes possible to determine the outlier score for smaller test sequences by repeated use of combination outliers. Specifically, the anomaly score of a sequence  $a_1 \dots a_n$  can be estimated as the product of the following conditionals:

$$P(a_1 \dots a_n) = P(a_1) \cdot P(a_2|a_1) \dots P(a_n|a_1 \dots a_{n-1}) \quad (10.2)$$

Note that each of these probability values is available in a probabilistic suffix tree of order (depth)  $n$ . For cases in which only a depth of  $k$  is maintained, the short-memory property of sequences can be used to approximate  $P(a_i|a_1 \dots a_{i-1})$  with  $P(a_i|a_{i-k} \dots a_{i-1})$ . Such an approach has been used in [524] for efficient determination of sequence outliers.

## 10.3.2 Distance-Based Models

In distance-based models, the absolute distance of the comparison unit is computed to equivalent windows of the training sequence. The distance of the  $k$ -th nearest neighbor window in the training sequence is used to determine the anomaly score. In the context of sequence data, some of the proximity-functions are *similarity* functions, whereas others are distance functions. An example of the former is the use of the *longest common subsequence* (*LCSS*), whereas an example of the latter is the *edit distance*. In the following, we provide a brief discussion of these proximity function, although more details may be found in [33, 23]:

- **Simple matching coefficient:** This is the simplest possible function and determines the number of matching positions between two sequences of equal length. This is also equivalent to the Hamming distance between a pair of sequences.
- **Longest common subsequence (and its normalized variants):** Let  $T_1$  and  $T_2$  be two sequences. Any sequence obtained by dropping some elements of a sequence is referred to as its *subsequence*. Therefore, the longest common subsequence of  $T_1$  and  $T_2$  is the longest sequence that is a subsequence of both  $T_1$  and  $T_2$ . Let the length of the longest common subsequence be denoted by  $L(T_1, T_2)$ . Then, the value  $NL(T_1, T_2)$  of the normalized longest common subsequence is computed by normalizing  $L(T_1, T_2)$  with the underlying sequence lengths in a similar way to the cosine computation between unordered sets:

$$NL(T_1, T_2) = \frac{L(T_1, T_2)}{\sqrt{|T_1|} \cdot \sqrt{|T_2|}} \quad (10.3)$$

The advantage of this approach is that it can match two sequences of unequal lengths. However, as with many sequence-based proximity function, a dynamic programming approach is required to compute the value of  $L(T_1, T_2)$ , which is rather slow. Details of this dynamic programming algorithm are provided in [33].

- **Edit distance:** The edit distance is one of the most common similarity functions used for sequence matching [239]. This function measures the distance between two sequences by the minimum number of edits required to transform one sequence to the other. The computation of the edit distance can be computationally very expensive.
- **Compression-based dissimilarity:** This measure is based on principles from information theory. Let  $W$  be a window of the training data, and  $W \oplus U_i$  be the string representing the concatenation of  $W$  and  $U_i$ . Let  $DL(S) < |S|$  be the description length of any string  $S$  after applying a standard compression algorithm to it. Then, the compression-based dissimilarity  $CD(W, U_i)$  is defined [312] as follows:

$$CD(W, U_i) = \frac{DL(W \oplus U_i)}{DL(W) + DL(U_i)} \quad (10.4)$$

This measure always lies in the range  $(0, 1)$  and lower values indicate greater similarity. The intuition behind this approach is that when the two sequences are very similar, the description length of the combined sequence will be much smaller than that of sum of the description lengths. On the other hand, when the sequences are very different, the description length of the combined string will be almost the same as the sum of the description lengths.

- **Other methods:** A variety of other methods may be used to compute similarity. These methods vary in terms of how two sequences may be aligned or the importance given to different lengths of the alignment. Some examples of such methods include counting mismatches among lookahead pairs [198], and a length-sensitive recursive computation of subsequence similarity [338, 339]. The latter method is of particular interest and is discussed below.

An important observation in the context of many sequence applications such as intrusion detection is that *contiguous mismatches are more important than non-contiguous mismatches*. This is because anomalous events usually cause *bursts* of anomalous symbols, which can be distinguished from the occasional noise. Let  $U_i$  be a comparison unit, and  $W$  be a window of the training sequence of the same length as  $U_i$ . For each position  $l$  in  $U_i$ , the length  $p$  of the longest contiguous set of positions to the left of position  $l$  is determined, so that the position indices  $\{l - p + 1, \dots, l\}$  in both sequences match exactly. This length is aggregated over the different values of  $l$  from 1 to  $|U_i|$ . The intuition here is that long contiguous matches are rewarded more than distributed and scattered matches.

In order to compute the anomaly score of a comparison unit  $U_i$  with respect to the training sequences in  $T_1 \dots T_N$ , the first step is to extract equivalent windows from  $T_1 \dots T_N$  as the size of the comparison unit. The  $k$ -th nearest neighbor distance is used as the anomaly score *for that comparison unit*. It now remains to explain how the results for the different comparison units  $U_1 \dots U_r$  extracted from  $V$  are combined to provide a unified anomaly score for the test sequence  $V$ .

### 10.3.2.1 Combining Anomaly Scores from Comparison Units

The anomaly scores from different comparison units extracted from a test sequence can be combined together in order to create a global anomaly score for the test sequence  $V$ . For ease in discussion, the default assumption is that higher scores correspond to greater outlierness. Some combination methods are as follows:

- **Number of anomalous units:** For each comparison unit, its anomaly score is compared to a specific threshold. The number of anomalous units among  $U_1 \dots U_r$  is reported as the final anomaly score. Note that this method is not specific to the distance-based model for anomaly detection, and can be used in the context of any of the models. For example, such an approach has been used in the context of intrusion detection in [197, 272]. However, it does lose some information because it does not use the values of the scores of the various units.
- **Aggregate anomaly score:** This aggregates the anomaly score over all comparison units. Although such an approach accounts for varying levels of anomaly scores, it can be impacted from the noisy scores of windows that are not anomalous.
- **Selective aggregate anomaly score:** This approach combines the virtues of the two aforementioned methods. In the first step, windows with anomaly score greater than a particular threshold are identified. The anomaly scores over these windows are aggregated to yield a final anomaly score.
- **Clustered anomaly scores:** It is generally recognized that *contiguous anomalous windows are generally more significant in application-specific scenarios than anomalous windows that are arbitrarily distributed over the sequence*. This is because in many motivating applications such as intrusion detection, anomalies are rare, but they often occur over locally clustered windows when they do occur.
  - **Locality frame count:** A method known as *Locality Frame Count (LFC)* proposed in [197] examines each possible set of  $\eta$  contiguous comparison units, as a *super-unit*. If number of anomalous units within this super-unit is larger than a given threshold, then the entire super-unit is deemed anomalous. The total number of anomalous super-units is reported as the outlier score. Note that this method is not specific to distance-based methods. In fact, a similar method has been used for HMM-based models, which will be discussed later in this chapter [211].
  - **Leaky bucket:** A method based on a similar principle has been proposed in [217]. In this method, the comparison units are scanned in temporal order, and a running count is maintained of the difference between the number of anomalous comparison units and the number of normal units. However, the running count is never allowed to fall below 0. The highest value of the running count over the entire sequence is reported as the outlier score. Such an approach also favors highly clustered anomalous comparison units. For example, interleaved normal and abnormal units will have a (combined) anomaly score no larger than 1. Note that the use of the leaky bucket is not specific to the use of distance-based methods, but can be used with virtually any model.

Different extreme-value analysis techniques can be used in order to identify thresholds at which scores should be considered anomalous. Such methods are discussed in Chapter 2.

### 10.3.2.2 Some Observations on Distance-Based Methods

Distance-based methods are more suitable for cases in which the comparison units are directly extracted from the test sequence. On the other hand, in many scenarios, comparison units may be provided by a domain expert and may correspond to significant semantic events (e.g., known intrusion patterns, shop-lifting patterns, hacking attack patterns, and so on). These correspond to the formulation in Definition 10.3.1. In such cases, other classes of models such as frequency-based or model-based methods (HMM) should be used. The latter classes of methods are suitable *both* for relative comparisons based on domain-dependent subsequences, and for absolute comparisons to the training data based on comparison units extracted directly from test sequences.

### 10.3.2.3 Easier Special Case: Short Sequences

The use of extracted comparison units or domain-specific comparison units is necessitated by the curse of dimensionality in distance computations over longer sequences. However, when the data contains a set of relatively short sequences of comparable size, straightforward generalizations of multidimensional methods can be used. This scenario corresponds to Definitions 10.3.3 and 10.3.4. Any standard  $k$ -nearest neighbor technique or clustering method discussed in Chapter 4 may be used. The major distinction is that the distances need to be computed directly on the sequences. For that purpose, any of the aforementioned distance (or similarity) functions in this chapter can be used. Specific examples of algorithms of the two types are as follows:

- A  $k$ -nearest neighbor approach, which uses the inverse of the similarity value as the anomaly score, is proposed in [127]. Such an approach has been shown to be quite effective in spite of its simplicity.
- The work in [103, 104] uses a  $k$ -medoid based clustering approach in order to determine relevant outliers. A method called *CLUSEQ* for clustering with the use of probabilistic suffix trees is proposed in [581]. The core idea is that a probabilistic suffix tree provides an efficient representation of a cluster, and this can be used for efficient similarity computations in the clustering process. The probabilistic suffix tree is pruned of infrequent nodes in order to ensure a more compact tree for efficiency. The approach is designed for clustering as the primary goal, but is also able to determine outliers as a side product.

## 10.3.3 Frequency-Based Models

Frequency-based models are typically used with domain-specific comparison units specified by the user. In this case, the relative frequency of the comparison unit needs to be measured in the training sequences and the test sequences, and the level of surprise is correspondingly determined. Such methods are primarily designed for user-specified comparison units; however, they can easily be extended to the setting of extracted comparison units.

### 10.3.3.1 Frequency-Based Model with User-Specified Comparison Unit

When the comparison units are specified by the user [310], a natural way of testing the anomaly score is to test the frequency of the comparison unit  $U_j$  in the training and test patterns. For example, when a sequence contains a hacking attempt, such as a sequence of *Login* and *Password* events, this sequence will have much higher frequency in the test



sequence, as compared to that in the training sequences. The specification of such relevant comparison units by a user provides very useful domain knowledge to an outlier analysis application.

Let  $f(T, U_j)$  represent the number of times that the comparison unit  $U_j$  occurs in the sequence  $T$ . Since the frequency  $f(T, U_j)$  depends on the length of  $T$ , the normalized frequency  $\hat{f}(T, U_j)$  may be obtained by dividing the frequency by the length of the sequence:

$$\hat{f}(T, U_j) = \frac{f(T, U_j)}{|T|} \quad (10.5)$$

Then, the anomaly score of the training sequence  $T_i$  with respect to the test sequence  $V$  is defined by subtracting the relative frequency of the training sequence from the test sequence. Therefore, the anomaly score  $A(T_i, V, U_j)$  is defined as follows:

$$A(T_i, V, U_j) = \hat{f}(V, U_j) - \hat{f}(T_i, U_j) \quad (10.6)$$

The absolute value of the average of these scores is computed over all the sequences in the database  $\mathcal{D} = T_1 \dots T_N$ . This represents the final anomaly score.

A useful output of this approach is the specific subset of comparison units specified by the user that are the most anomalous. This provides intensional knowledge and feedback to the analyst about *why* a particular test sequence should be considered anomalous. A method called *TARZAN* [310] uses suffix tree representations to efficiently determine all the anomalous subsequences in a comparative sense between a test sequence and a training sequence.

One issue with this model is that since the comparison units are specified by the user, they may not always be of the compact size required for effective frequency computations (unlike the case in which comparison units are extracted from test sequences with compactness in mind). When the comparison units themselves are long, the frequencies of these units in the training sequences may be small or zero. As a result, the anomaly scores may no longer remain significant. A number of methods have been proposed in the literature in order to address these issues, most of which have to do with examining the behavior of smaller portions of the comparison units. In particular, a method proposed in [310] uses subsequences of the comparison units, and computes an aggregate quantity as a function of the frequencies of the underlying subsequences. A different method in [243] allows relaxation of the definition of “subsequence” in counting frequencies by allowing permutations of the comparison unit. The assumption is that the exact ordering of the symbols within a short window is not as important in certain application-specific scenarios. In fact, in such cases, relaxations can add to the robustness of the similarity computations.

In order to improve the *efficiency* of frequency computation, the work in [244] suggests that the training sequences should be decomposed into smaller windows for subsequence computations. The number of windows in which the comparison unit occurs is used as a proxy for the frequency.

### 10.3.3.2 Frequency-Based Model with Extracted Comparison Units

The frequency-based model is mostly designed for the case of user-specified comparison units. However, it is also possible to generalize the model to the case where extracted comparison units are used. In this model, the comparison unit is extracted directly from the test sequence. For each comparison unit, the same model may be used, as in the case of user-specified comparison units. Although such methods are feasible, they have generally

not been used very frequently in the literature. Distance-based methods are more popular, when comparison units are extracted directly from the test sequences.

### 10.3.3.3 Combining Anomaly Scores from Comparison Units

Although frequency-based methods are generally used for surprise detection with user-specific comparison units, the results from multiple comparison units can also be combined into a single anomaly score. When many comparison units are specified by a user, this can be used to create a unified comparison score. The methodology for achieving this goal is the same as discussed for distance-based methods in section 10.3.2.1. This approach is also applicable when the comparison units are extracted directly from the test sequences. The former results in a domain-dependent anomaly score, whereas the latter results in an unsupervised anomaly score.

### 10.3.4 Hidden Markov Models

Hidden Markov models (HMMs) are probabilistic models that generate sequences through a sequence of transitions between states in a Markov chain. So how are hidden Markov models different from the Markovian techniques introduced earlier in this chapter? Each state in the Markovian techniques introduced earlier in this chapter is well defined by a particular position in the sequence. The state is uniquely defined by the previous  $k$  positions in the sequence from that position. This state is also directly visible to the user in terms of the precise order of transitions for a particular training or test sequence. Thus, the generative behavior of the Markovian model is always *fully visible* for a given sequence.

In a hidden Markov model, the states of the system are *hidden* by definition, which implies that they are not directly visible to the user. Only a sequence of (typically) discrete observations is visible to the user, which are generated by symbol emissions from the states after each transition. These observations correspond to the application-specific sequence data. In many cases, the states may be defined (during the modeling process) on the basis of an *understanding* of how the underlying system behaves, although the precise sequence of transitions may not be known to the analyst. This is why such models are referred to as “hidden.”

Each state is associated with a *set of emission probabilities* over the symbol  $\Sigma$ . In other words, a visit to the state  $j$  leads to an emission of one of the symbols  $\sigma_i \in \Sigma$  with probability  $\theta^j(\sigma_i)$ . Correspondingly, a sequence of transitions ( $\sigma_i$ ) in a hidden Markov model corresponds to an *observed data sequence*. Hidden Markov models may be considered special cases of the mixture models discussed in Chapter 2. The main caveat is that the different components of the mixture are not independent of one another, but are related through sequential transitions. Thus, each state is analogous to a component in the multidimensional mixture model of Chapter 2. Each symbol generated by this model is analogous to a data point generated by the multidimensional mixture model. Furthermore, the successive generation of data items (sequence symbols) is also dependent on the previous history. This is a natural consequence of the fact that the successive states emitting the data items are not independent of one another, because they continually transition to one another. Unlike traditional mixture models, hidden Markov models are specifically designed to capture the temporal correlations in sequential data.

In order to better explain hidden Markov models, an illustrative example will be used. Consider the scenario where a set of students register for a course, and generate a sequence corresponding to the grades received in each of their weekly assignments. This grade is drawn

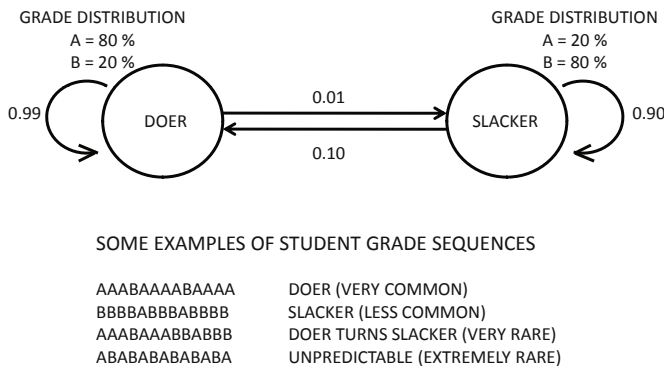


Figure 10.3: Generating grade sequences from a hidden Markov model

from the symbol set  $\Sigma = \{A, B\}$ . The model created by the analyst is that the class contains students who, at any given time, are either *doers* or *slackers* with different grade-generation probabilities. A student in a *doer* state may sometimes transition to a *slacker* state and vice versa. These represent the two states in the system. Weekly home assignments, which are given to the students, are graded with one of the symbols from  $\Sigma$ . This results in a *sequence* of grades for each student, and it represents the only *observable* output for the analyst. The (hidden) state of a student represents only a *model* created by the analyst to *explain* the grade sequences and is, therefore, not observable in of itself. It is important to understand that if this model is a poor reflection of the true generative process, then the accuracy of the learner will be poor.

Assume that a student in a *doer* state is likely to get an *A* grade in a home assignment with an 80% probability and a *B* with a 20% probability. For *slackers*, these probability values are reversed. Although we have explicitly specified these probabilities here for illustrative purposes, they are usually not known a priori and need to be *learned* or *estimated* from the observed grade sequences. The precise state of a student is not known to the analyst at any given time. These grade sequences are, in fact, the only *observable* outputs for the analyst. Therefore, from the perspective of the analyst, this is a *hidden* Markov model, which generates the sequences of grades from an *unknown* sequence of state transitions. This unknown sequence can only be probabilistically *estimated* for a particular observed sequence.

The two-state hidden Markov model for the aforementioned example is illustrated in Figure 10.3. This model contains two states corresponding to *doer* and *slacker*, which represent the state of a student in a particular week. It is possible for a student to transition from one state to another, although the likelihood of this is low. It is assumed that the set of initial state probabilities governs the a priori distribution of *doers* and *slackers*. These probabilities represent the a priori understanding about the students when they join the course. Some examples of *possible sequences* and their corresponding rarity levels are illustrated in Figure 10.3. For example, the sequence **AAABAAAABAAA** is most likely generated by a student who is consistently in a *doer* state, and the sequence **BBBBABBABBBB** is most likely generated by a student who is consistently in *slacker* state. The second sequence is typically rarer than the first because the population predominantly contains<sup>1</sup> *doers*. The sequence

<sup>1</sup>The assumption is that the initial set of state probabilities are approximately consistent with the steady state behavior of the model for the particular set of transition probabilities shown in Figure 10.3.

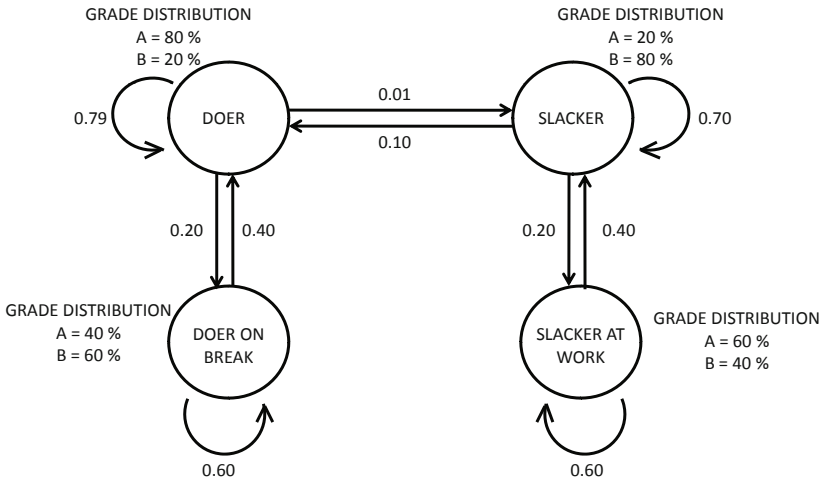


Figure 10.4: Extending the model in Figure 10.3 with two more states provides greater expressive power for modeling sequences

AAABAAABBABBB corresponds to a *doer* who eventually transitions into a *slacker*. This case is even rarer, because it requires a transition from the *doer* state to a *slacker* state, which has very low probability. The sequence ABABABABABABA is *extremely anomalous*, because it does not represent temporally consistent *doer* or *slacker* behavior that is implied by the model. Correspondingly, such a sequence has very low probability of fitting the model.

A larger number of states in the Markov model can be used to encode more complex scenarios. It is possible to encode domain knowledge to improve the basic design of the state layouts in the model. For example, *doers* may sometimes slack off for short periods and then return to their usual state. Similarly, *slackers* may sometimes become temporarily inspired to be *doers*, but may eventually return to what they are best at. Such episodes will result in local portions of the sequence that are distinctive from the remaining sequence. These scenarios can be captured with the 4-state Markov Model illustrated in Figure 10.4. A larger number of states enable the modeling of more complex scenarios. In general, more training data is required to learn the (larger number of) parameters of such a model without overfitting. For smaller data sets, the transition probabilities and symbol-generation probabilities are not estimated accurately. As discussed in Chapter 2, this is a common problem of generative models. Therefore, a number of important issues arise about the initial design choices in a hidden Markov model. These will be discussed in the next subsection.

10.3.4.1 Design Choices in a Hidden Markov Model

In the previous example, an *understanding* of the generating process was used to design the basic architecture of the hidden Markov model. Such an understanding may or may not be available in a given application. A good initial design (i.e., state layout) of the Markov model is crucial, because a poor design could result in the model either overfitting the data or not fitting the training sequences at all. Two primary design choices are key to this process:

- In its most general form, a hidden Markov model with  $n$  states will have  $n^2$  possible

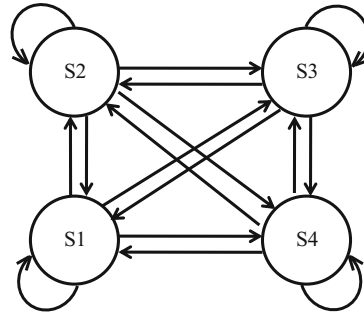


Figure 10.5: A black-box hidden Markov model with four states and no encoded domain knowledge: A larger number of transition probabilities need to be learned.

transitions between different states (including self-transitions), and  $n \cdot |\Sigma|$  symbol generation probabilities, where one distribution is associated with each state. This is a black-box model, in which no prior knowledge is encoded about the generating process. In such cases, the initial Markov model is a clique with  $n$  states. In practice, the states may be designed with an intuitive understanding about the underlying generating process, and not all pairs of states may have transitions among them. This encodes significant *domain knowledge* about the sequence generation process. For example, in the case of Figure 10.4, the states are intuitively associated with characteristics of the generating process (students). Furthermore, transitions do not occur between all pairs of states. These states represent a priori domain knowledge of the analyst about the generative process for the sequences.

The equivalent four-state black-box Markov Model is illustrated in Figure 10.5. Note that the states are not labeled in this case, because they do not reflect an analyst's understanding of the underlying generating process. Clearly, different trade-offs exist between these choices. By selecting a particular topology of the model based on domain understanding, the number of parameters is greatly reduced. For example, fewer transition probabilities need to be learned in Figure 10.4 as compared to Figure 10.5. In the former case, many transition probabilities have already been set to zero before the training process, and this reflects the domain-specific understanding. This reduces the likelihood of overfitting in the case of Figure 10.4. On the other hand, if the specific topology of the domain-specific architecture of Figure 10.4 does not truly reflect the student behavior, then the resulting model will also poorly reflect the training data. In this sense, the model of Figure 10.5 has the advantage that it can be used as black-box for the sequences from any domain.

- The number of states represents the level of complexity in the Markov model. This is analogous to the number of components in the mixture model of Chapter 2. A larger number of states can encode greater complexity about variations in sequence patterns, but may also overfit a training data set, when it is small. Therefore, when the black-box approach is used, it needs to be decided up front, how many states should be used in the model. When the data sets available are small, it is advisable to use a smaller number of states.

The first of the aforementioned design choices is important, because hidden Markov models are particularly effective when they are used to encode domain-specific understanding. The

ability to encode a set of sequences with a Markov model of a particular architecture is dependent on the understanding of the analyst about the generating process. As discussed throughout the book, the ability to encode domain knowledge in an indirect way during the model construction process is crucial for effective outlier analysis.

### 10.3.4.2 Training and Prediction with HMMs

In this section, hidden Markov models will be defined formally, and the associated training methods will be introduced. It is assumed that a hidden Markov model contains  $n$  states denoted by  $\{s_1 \dots s_n\}$ . The symbol set from which the observations are generated is denoted by  $\Sigma = \{\sigma_1 \dots \sigma_{|\Sigma|}\}$ . The symbols are generated from the model by a sequence of transitions from one state to another. Each visit to a state (including self-transitions) generates a symbol drawn from a categorical probability distribution on  $\Sigma$ . The symbol emission distribution is specific to each state. The probability  $P(\sigma_i | s_j)$  that the symbol  $\sigma_i$  is generated from state  $s_j$  is denoted by  $\theta^j(\sigma_i)$ . The probability of a transition from state  $s_i$  to  $s_j$  is denoted by  $p_{ij}$ . The initial state probabilities are denoted by  $\pi_1 \dots \pi_n$  for the  $n$  different states. The topology of the model can be expressed as a network  $G = (M, A)$ , in which  $M$  is the set of states  $\{s_1 \dots s_n\}$ . The set  $A$  represents the possible transitions between the states. If the architecture of the HMM is constructed with a domain-specific understanding, the set  $A$  is not a complete network, and, therefore, many transition probabilities are implicitly set to 0. *The goal of training the HMM model is to learn the initial state probabilities, transition probabilities, and the symbol emission probabilities from the training database  $\{T_1 \dots T_N\}$ .*

Three methodologies are commonly leveraged in creating and using a hidden Markov model:

- **Training:** Given a set of training sequences  $T_1 \dots T_N$ , estimate the model parameters, such as the initial probabilities, transition probabilities, and symbol-emission probabilities with an expectation-maximization algorithm. The Baum-Welch algorithm is used for this purpose.
- **Evaluation:** Given a test sequence  $V$  (or comparison unit  $U_i$ ), determine the probability that it fits the HMM. This process is used to determine the anomaly scores. A recursive forward algorithm is used to compute this.
- **Explanation:** Given a test sequence  $V$ , determine the most likely sequence of states that generated this test sequence. This is helpful for providing an understanding of why a sequence should be considered an anomaly, when the states correspond to an intuitive understanding of the underlying system. In the example of Figure 10.3, it would be useful to know that an observed sequence is an anomaly, because of the unpredictable switching of a student between *doer* and *slacker* states. This can provide the *intensional knowledge* for understanding the state of a system. The most likely sequence of states is computed with the Viterbi algorithm.

Since the description of the training procedure relies on technical ideas developed for the evaluation method, we will deviate from the natural order of presentation and present the training algorithms at the very end. The evaluation and explanation techniques will assume that the model parameters, such as the transition probabilities, are already available from the training phase.

### 10.3.4.3 Evaluation: Computing the Fit Probability for Observed Sequences

One approach for determining the fit probability of test sequence  $V = a_1 \dots a_m$  would be to compute all the  $n^m$  possible sequences of states (paths) in the HMM, and compute the probability of each. This probability can be computed based on the observed sequence, symbol-generation probabilities, and transition probabilities. The sum of these values can be reported as the fit probability. Obviously, such an approach is not practical because it requires the enumeration of an exponential number of possibilities.

The computation can be greatly reduced by recognizing that the fit probability of the first  $r$  symbols (and a fixed value of the  $r$ th state) can be recursively computed in terms of the corresponding fit probability of the first  $(r - 1)$  observable symbols (and a fixed  $(r - 1)$ th state). Specifically, let  $\alpha_r(V, s_j)$  be the probability that the first  $r$  symbols in  $V$  are generated by the model, and the final state in the sequence is  $s_j$ . Then, the recursive computation is as follows:

$$\alpha_r(V, s_j) = \sum_{i=1}^n \alpha_{r-1}(V, s_i) \cdot p_{ij} \cdot \theta^j(a_r) \quad (10.7)$$

This approach recursively sums up the probabilities of all the  $n$  different paths for different penultimate nodes. The aforementioned relationship is iteratively applied for  $r = 1 \dots m$ . The probability of the first symbol is computed as  $\alpha_1(V, s_j) = \pi_j \cdot \theta^j(a_1)$  for initializing the recursion. This approach requires  $O(n^2 \cdot m)$  time. Then, the overall probability is computed by summing up the values of  $\alpha_m(V, s_j)$  over all possible states  $s_j$ . Therefore, the final fit  $F(V)$  is computed as follows:

$$F(V) = \sum_{j=1}^n \alpha_m(V, s_j) \quad (10.8)$$

This algorithm is also known as the *Forward Algorithm*.

### 10.3.4.4 Explanation: Determining the Most Likely State Sequence for Observed Sequence

One of the goals of outlier analysis is to provide an explanation for why a data point or sequence should be considered an outlier. Since the sequence of (hidden) generating states often provides an intuitive explanation for the observed sequence, it is sometimes desirable to determine the *most likely sequence of states* for the observed states. This is achieved with the *Viterbi algorithm*.

One approach for determining the most likely state path of the test sequence  $V = a_1 \dots a_m$  is to compute all the  $n^m$  possible sequences of states (paths) in the HMM, and compute the probability of each of them, based on the observed sequence, symbol-generation probabilities, and transition probabilities. The maximum of these values can be reported as the most likely path. This algorithm would have exponential complexity. Note that this is a similar problem to the fit probability except that it is needed to determine the *maximum* rather than the *sum*, over all possible paths. Correspondingly, it is also possible to use a similar recursive approach as the previous case to determine the most likely state sequence.

Any sub-path of an optimal state path must also be optimal for generating the corresponding subsequence of symbols. This property, in the context of an optimization problem of sequence selection, normally enables dynamic programming methods. The best possible state path for generating the first  $r$  symbols (with the  $r$ th state fixed to  $j$ ) can be recursively computed in terms of the corresponding best paths for the first  $(r - 1)$  observable

symbols and different penultimate states. Specifically, let  $\delta_r(V, s_j)$  be the probability of the best state sequence for generating the first  $r$  symbols in  $V$  and ending at state  $s_j$ . Then, the recursive computation is as follows:

$$\delta_r(V, s_j) = \text{MAX}_{i=1}^n \delta_{r-1}(V, s_i) \cdot p_{ij} \cdot \theta^j(a_r) \quad (10.9)$$

This approach recursively computes the maximum of the probabilities of all the  $n$  different paths for different penultimate nodes. The approach is iteratively applied for  $r = 1 \dots m$ . The first probability is determined as  $\delta_1(V, s_j) = \pi_j \cdot \theta^j(a_1)$  for initializing the recursion. This approach requires  $O(n^2 \cdot m)$  time. Then, the best path is finally computed by using the maximum value of  $\delta_m(V, s_j)$  over all possible states  $s_j$ . This approach is, essentially, a dynamic programming algorithm.

### 10.3.4.5 Training: Baum-Welch Algorithm

The problem of determining the optimal parameters in the case of HMM is very difficult, and no known algorithm is guaranteed to determine the global optimum. However, a number of options are available to determine a reasonably effective solution in most scenarios. The Baum-Welch algorithm is one such method. It is also known as the *Forward-backward* algorithm, and it is an application of the expectation-maximization approach to the generative hidden Markov model. First, a description of training with the use of a single sequence  $T = a_1 \dots a_m$  will be provided. Then, a straightforward generalization to  $N$  sequences  $T_1 \dots T_N$  will be discussed.

Let  $\alpha_r(T, s_j)$  be the *forward* probability that the first  $r$  symbols in a sequence  $T$  of length  $m$  are generated by the model, and the last symbol in the sequence is  $s_j$ . Let  $\beta_r(T, s_j)$  be the *backward* probability that the portion of the sequence after *and not including the  $r$ th position* is generated by the model, *conditional on the fact that* the state for the  $r$ th position is  $s_j$ . Thus, the forward and backward probability definitions are not symmetric. The forward and backward probabilities can be computed from model probabilities in a similar way as the evaluation procedure discussed above in section 10.3.4.3. The major difference for the backward probabilities is that the computations start from the end of the sequence in the backward direction. Furthermore, the probability value  $\beta_{|T|}(T, s_j)$  is initialized to 1 at the bottom of the recursion to account for the difference in the two definitions. Two additional probabilistic quantities need to be defined in order to describe the EM algorithm:

- $\psi_r(T, s_i, s_j)$ : Probability that the  $r$ th position in sequence  $T$  corresponds to state  $s_i$ , the  $(r + 1)$ th position corresponds to  $s_j$ .
- $\gamma_r(T, s_i)$ : Probability that the  $r$ th position in sequence  $T$  corresponds to state  $s_i$ .

The EM procedure starts with a random initialization of the model parameters and then iteratively estimates  $(\alpha(\cdot), \beta(\cdot), \psi(\cdot), \gamma(\cdot))$  from the model parameters, and vice versa. Specifically, the iteratively executed steps of the EM procedure are as follows:

- **(E-Step):** Estimate  $(\alpha(\cdot), \beta(\cdot), \psi(\cdot), \gamma(\cdot))$  from currently estimated values of the model parameters  $(\pi(\cdot), \theta(\cdot), p..)$ .
- **(M-Step):** Estimate model parameters  $(\pi(\cdot), \theta(\cdot), p..)$  from currently estimated values of  $(\alpha(\cdot), \beta(\cdot), \psi(\cdot), \gamma(\cdot))$ .



It now remains to explain how each of the aforementioned estimations is performed. The values of  $\alpha(\cdot)$  and  $\beta(\cdot)$  can be estimated using the forward and backward procedures, respectively. The forward procedure is already described in the evaluation section, and the backward procedure is analogous to the forward procedure, except that it works backward from the end of the sequence. The value of  $\psi_r(T, s_i, s_j)$  is equal to  $\alpha_r(T, s_i) \cdot p_{ij} \cdot \theta^j(a_{r+1}) \cdot \beta_{r+1}(T, s_j)$ , since the sequence-generation procedure can be divided into three portions corresponding to the sequence portion up to position  $r$ , the generation of the  $(r+1)$ th symbol, and the portion after the  $(r+1)$ th symbol. The estimated values of  $\psi_r(T, s_i, s_j)$  are normalized to a probability vector by ensuring that the sum over different pairs  $[i, j]$  is 1. The value of  $\gamma_r(T, s_i)$  is estimated by summing up the values of  $\psi_r(T, s_i, s_j)$  over fixed  $i$  and varying  $j$ . This completes the estimations of the E-step.

The re-estimation formulas for the model parameters in the M-Step are relatively straightforward. Let  $I(a_r, \sigma_k)$  be a binary indicator function, which takes on the value of 1 when the two symbols are the same, and 0, otherwise. Then the estimations can be performed as follows:

$$\begin{aligned} \pi(j) &= \gamma_1(T, s_j), \quad p_{ij} = \frac{\sum_{r=1}^{m-1} \psi_r(T, s_i, s_j)}{\sum_{r=1}^{m-1} \gamma_r(T, s_i)} \\ \theta^i(\sigma_k) &= \frac{\sum_{r=1}^m I(a_r, \sigma_k) \cdot \gamma_r(T, s_i)}{\sum_{r=1}^m \gamma_r(T, s_i)} \end{aligned}$$

The precise derivations of these estimations on the basis of expectation-maximization techniques may be found in [454]. This completes the estimations for the M-step.

As in all EM methods, the procedure is applied iteratively to convergence. More details on robust termination criteria may be found in [454]. The approach can be generalized easily to  $N$  sequences by applying the steps to each of the sequences, and averaging the corresponding model parameters in each step.

### 10.3.4.6 Computing Anomaly Scores

In theory, it is possible to compute anomaly scores directly for the test sequence  $V$ , once the training model has been constructed from the sequence database  $\mathcal{D} = T_1 \dots T_N$ . However, as the length of the test sequence increases, the robustness of such a model diminishes because of the increasing noise resulting from the curse of dimensionality. Therefore, the comparison units (either extracted from the test sequence or specified by the domain expert) are used for computing the anomaly scores.

- When the comparison units are extracted from the test sequence (the absolute model of Definition 10.3.2), it is possible to compute the probability of fit of each comparison unit to the training data. The negative logarithm of the resulting probability provides an anomaly score in which large values are indicative of anomalous behavior for a particular comparison unit. At this point, the method of section 10.3.2.1 can be used to combine the final anomaly scores over different windows.
- When the comparison units are provided by a domain expert (the relative model of Definition 10.3.2), two separate Markov Models are computed for the training and test sequences, respectively. This approach will not work, if the test sequence is not sufficiently long to create a robust model without overfitting. The negative logarithm of the fit probability of the comparison unit to the two models is computed with the Viterbi algorithm. The difference in the two values is reported as the anomaly score.

The second of the two methods is used rarely, because of the tendency of a single test sequence to create a model which overfits the data.

A number of methods also use the Viterbi algorithm on the test sequence in order to mine the most likely state sequence. In some domains, it is easier to determine anomalies in terms of the state sequence rather than the observable sequence. Furthermore, low transition probabilities on portions of the state sequence provide anomalous localities of the observable sequence. The drawback is that the most likely state sequence may have a very low probability of matching the observed sequence. Therefore, the estimated anomalies may not reflect the true anomalies in the data when an *estimated* state sequence is used for anomaly detection. The real utility of the Viterbi algorithm is in providing an *explanation* of the anomalous behavior of sequences in terms of the intuitively understandable states rather than quantifying its anomaly score.

#### 10.3.4.7 Special Case: Short Sequence Anomaly Detection

A special case arises, when database  $\mathcal{D}$  of shorter sequences is available, and the anomalous sequences need to be determined, on the basis of their dissimilarity with other sequences. This corresponds to Definition 10.3.4. As discussed earlier, this problem is similar to point anomaly detection formulations, which are used in multidimensional data. Distance- and clustering-based methods for addressing this special case have been discussed in section 10.3.2.3. A question arises whether HMMs can also be used for probabilistic clustering and anomaly detection in sequences with the use of mixture models.

It turns out that it is possible to use hidden Markov models for clustering and anomaly detection. The idea is to represent the data as a *mixture of HMMs* [111, 500]. As discussed earlier, the HMM is itself a mixture model, in which each state can be considered a component of a mixture. This approach uses a *second* level of mixture modeling, where the observations correspond to individual sequences from  $\mathcal{D}$  rather than the individual symbols of a sequence. A single HMM model for a training data set makes the implicit assumption that every sequence  $T_i \in \mathcal{D}$  is generated from the same distribution (cluster). This is not true in practice, since the individual sequences might themselves belong to different clusters, each of which has its own generating HMM. This can be modeled using a second level of mixture modeling, in which the  $k$  independent HMMs are used, and each sequence  $T_i \in \mathcal{D}$  is associated with a generative probability from this mixture. Sequences that have low generative probabilities, or which have low probability of assignment to their best matching component may be considered anomalies.

### 10.3.5 Kernel-Based Methods

An interesting approach for anomaly detection in whole sequences is to use kernel methods. Consider a setting, where we have  $N$  sequences, we would like to determine outliers among them, without using window-based analysis. In other words, each sequence is treated individually as a comparison unit.

The basic idea in kernel-based methods is to use linear models in a transformed multidimensional space. In other words, each sequence is implicitly transformed into a multidimensional space with the use of a kernel similarity function. The basic idea here is that we can construct an  $N \times N$  matrix  $S$  of similarities between the various sequences. Typically, kernel functions are used to compute these similarities. Some examples of such kernels include the use of the *bag-of-words kernel*, *spectrum kernel*, and the *weighted degree kernel* [33]. All these kernel functions have the virtue of being positive-semidefinite which is essential for

extracting a valid embedding from the similarity matrix  $S$ . One can then extract a multidimensional ( $k$ -dimensional) embedding from the symmetric and positive semi-definite matrix  $S$  by extracting all the  $k$  strictly positive eigenvectors from the embedding as follows:

$$S \approx Q\Lambda^2Q^T \quad (10.10)$$

Here,  $Q$  is an  $N \times k$  matrix with orthonormal columns and  $\Lambda$  is a  $k \times k$  diagonal matrix. The matrix  $Q\Lambda$  is an  $N \times k$  matrix containing the  $k$ -dimensional embeddings of the data points. Furthermore, each of these  $k$  dimensions are uncorrelated with one another, which makes it particularly easy to use the Mahalanobis method. The first step is to normalize each column of  $Q\Lambda$  to unit norm, which yields the matrix  $Q$ . Subsequently, the  $k$ -dimensional mean of the rows of  $Q$  is computed. The Euclidean distance of each point to this mean provides the Mahalanobis outlier score in the embedded space.

It is noteworthy that this approach is implemented with the use of kernel similarity functions among strings, which may not always satisfy domain-specific criteria. Therefore, one might ask whether it is possible to use more traditional similarity functions like the use of the longest common subsequence or other heuristic methods. The main problem in using such methods is that the resulting matrix might not be positive semi-definite. In such cases, we might not be able to diagonalize the matrix with non-negative eigenvalues, which is required to extract the embedding  $Q\Lambda$ . Note that the matrix  $\Lambda^2$  in Equation 10.10 is required to contain nonnegative eigenvalues. In such cases, one can still *heuristically* extract an approximate embedding. For many natural similarity functions, the negative eigenvalues are small in magnitude, and they may be truncated to 0 as an approximation. It can be shown that such an approach closely approximates the original similarity matrix when the negative eigenvalues are small.

Kernel functions can be used in combination with other types of linear models. For example, one can adapt the one-class support-vector machine to outlier detection in sequences by using the kernel similarity function. This approach boils down to a straightforward application of the technique in section 3.4 of Chapter 3.

## 10.4 Complex Sequences and Scenarios

---

In many real applications, the available sequences may be more complex than the ones that have been introduced so far in this chapter. For example, in a system diagnosis application, multiple sequences may continuously be produced by different types of sensors. In other cases, *each element* of the sequence may be a *set* drawn from  $\Sigma$  rather than a solitary symbol. In some scenarios such as online applications, it may be desirable to declare a sequence an anomaly as early as possible by observing only the early part of it. These cases may sometimes require more complex techniques than those discussed so far.

### 10.4.1 Multivariate Sequences

Multivariate sequences are quite common in system diagnosis applications, in which multiple sensors may record sequences drawn from possibly different alphabets. Consider the case, where  $r$  different sensors record sequences drawn from the (respective) alphabets  $\Sigma_1 \dots \Sigma_r$ . Each generated symbol is associated with a time-stamp, which is particularly crucial in the multivariate case in order to determine the temporal correspondence between the sequences generated by the different sensors.

Much work has not been done in the literature on the multivariate case for *discrete* sequences, as compared to the continuous case [140]. However, a variety of solutions are possible for finding unusual *time-windows* by combining the results from univariate analysis of the different series. Let  $q_i(t_c-h, t_c)$  represent the generative probability of the subsequence generated in the time-interval  $(t_c-h, t_c)$ . This generative probability can be computed using Equation 10.1 on an individual sequence. The final generative probability for the anomaly score of the *time window*  $(t_c-h, t_c)$  over all the  $r$  different series is given by  $\prod_{i=1}^r q_i(t_c-h, t_c)$ .

The case in which the anomalies need to be discovered on the entire sequence is a bit simpler. In this setting, we have a database of  $N$  multivariate sequences and we wish to discover the anomalies among them. In that case, one can use an approach similar to that discussed in section 10.3.5. The only difference is that a *multivariate* similarity function needs to be used to construct an  $N \times N$  similarity matrix and extract the kernel embedding. All other steps are identical to those discussed in section 10.3.5.

### 10.4.2 Set-Based Sequences

Most of the algorithms in this chapter are designed for the case of sequences in which each symbol is drawn from the symbol set  $\Sigma$ . In practice, each unit element of the sequence may be a set  $S_i \subseteq \Sigma$ . Such cases arise commonly in domains such as market-basket analysis. The analysis of anomalies in such complex sequences are different from the case of simpler sequences, because two set-based elements may have varying levels of similarity, depending upon the number of common elements between the sets. Furthermore, temporal correlations between the set-elements in such sequences are typically much weaker than symbol-only sequences because of the larger number of possibilities for a set element in the sequence. For example, in the market basket scenario, it is generally not meaningful to predict conditional probabilities of set-based symbols based on the previous history.

The problem of finding anomalies in set-based sequences is much closer to the problem of finding unusual novelties or change analysis in multidimensional data. Here, each set is treated as discrete multidimensional binary vector of dimensionality  $|\Sigma|$ , corresponding to whether or not an element of  $\Sigma$  is present in the set. A number of methods available in the literature are applicable to these scenarios:

- The models discussed in Chapter 8 for first-story detection [622] in text can be applied to this scenario by treating each set-element analogous to a document, and the symbol set  $\Sigma$  analogous to a text lexicon. In fact, a few methods [29] treat the problem of novelty detection of text, categorical data, and market-basket sequences in a unified way, with the use of clustering. The creation of new clusters in a streaming cluster generation process, corresponds to the relevant anomalies. It has been shown in [29], that such a method can simultaneously determine evolving clusters, anomalies, and newly forming trends in the underlying data.
- Some methods have also been designed to find surprising patterns with the use of the concept of *temporal description length* [122]. The goal is to determine itemsets and data segments which reflect significant changes in correlations over time. This technique borrows ideas from information theory. The itemsets are temporally segmented. For each segment, the minimum number of bits required to encode each segment is determined. Segments that require a larger number of bits to encode are assumed to be less homogeneous and may contain surprising patterns or change points. Such segments may be deemed to be outliers.

In general, set-based sequences are much closer in spirit to sparse multidimensional data, and provide a much richer domain for analytical purposes. Numerous methods for aggregate change analysis discussed in Chapter 8 can also be easily generalized to this case.

### 10.4.3 Online Applications: Early Anomaly Detection

In many sequential data mining applications, such as those arising in Web click-streams, or systems diagnosis applications, it may be desirable to perform the analysis in online fashion, as more data is received. In such cases, most of the techniques discussed earlier can be adapted to the online case:

- **Position outliers:** For the case of position outliers, the prediction is dependent only on a short memory before the position being predicted. The current model can be used to make an efficient prediction. Furthermore, many of the commonly used training models such as probabilistic suffix trees can be efficiently updated as more sequences are added to the training database.
- **Combination outliers:** In these cases, the unit of prediction is typically the comparison unit. As long as the next comparison unit has been derived, it can be used in conjunction with the current model to make the prediction. The process of updating the training model may sometimes be more challenging. Some models such as  $k$ -nearest neighbors may slow down with increasing training data size. In such cases, the training data may need to be sampled in order to retain efficient prediction.

Numerous examples of online sequential anomaly detection methods exist in domains as diverse as systems diagnosis and intrusion detection [10, 22].

## 10.5 Supervised Outliers in Sequences

---

Because of the complexity of sequence data, it is often hard to obtain meaningful outliers with the use of purely unsupervised methods. Many patterns that are discovered as outliers may correspond to noise, and may not represent true anomalies. It has been shown in diverse application domains such as system call anomalies [338], financial fraud [374], and Web sequence robot detection [531], that the nature of the anomalous sequences are highly specific and governed by the underlying application. Therefore, the ability to distinguish noise from anomalies can only be obtained by incorporating additional knowledge about previously known (and interesting) examples into the outlier analysis process. Such learning methods can significantly improve the quality and relevance of the underlying outliers in real application scenarios.

The problem of supervised outlier detection in sequence data is equivalent to sequence classification. As in all supervised outlier analysis methods, the major difference is in terms of class imbalance and cost sensitivity, which needs to be accounted for in the classification process. This ensures that rare classes can be meaningfully discovered. In these models, labels may be available in different ways:

- Labels may be associated with each position in a sequence. Such situations occur commonly in natural language data, in which the different words may need to be classified in a sentence [51, 337]. In the rare-class versions of this problem, most of the labels may belong to the normal category, but an occasional label may correspond to an anomaly (e.g., a word which is “out of place” or grammatically incorrect).

- Rare labels may be associated with small subsequences of a single large sequence [16]. Other subsequences are assumed to be normal by default.
- Labels are associated with the full sequences in a database of multiple sequences. Most labels are normal, but a small number of them may be anomalous. This is the most common scenario in real applications such as intrusion detection.

The second (window-based) formulation above can be transformed to the third (full-sequence) formulation by extracting windows of subsequences from the full sequence, and classifying them with the use of a full-sequence classifier. Furthermore, in many cases of temporal sequences, it is desirable to learn the label of a sequence by examining as little of it as possible from the early portions of the sequence. This is known as *early classification*. Detailed surveys on sequence classification may be found in [11, 575].

Although most of the methods for classification of sequence data are not designed for the imbalanced case, the generalization to the imbalanced case is straightforward with the use of the meta-algorithms discussed in Chapter 7. Thus, the issue of class imbalance is orthogonal to the actual techniques that are used for sequence classification. While sequence classification is too vast an area to be covered in this chapter, a brief review of the main classes of techniques are discussed below.

### Feature Transformations

The first class of methods extracts symbolic sequences of size  $k$  from the base sequences. These are referred to as  $k$ -grams, and they form the “words” from the sequence vocabulary. These words are used as the features, and a variety of classifiers such as SVMs [354] can be used on this new representation. Another class of feature extraction methods includes *pattern-based methods*, in which relevant patterns with sufficient support and confidence are mined. Such an approach mines the *local* patterns from the underlying data, which are relevant to sequence classification. Other forms of feature extraction such as wavelet decomposition [13] can be used, which extract both the local and global properties of the underlying data. A rule-based classifier was constructed which relates the wavelet patterns to the underlying classes.

### Distance-based Methods

As discussed in Chapter 9 on time-series outlier analysis, distance-based methods are particularly popular in the context of continuous time-series. In the case of continuous data, distance functions such as the Euclidean metric can be used. However, for discrete data sets, popular methods for sequence alignment such as the edit distance [239] can be used. Distance-based methods can also be used *in combination* with feature transformation methods. For example, the  $k$ -gram representation can be used in combination with distance-based methods in order to construct sequence classifiers.

### Hidden Markov Models

These are extremely popular methods, which can be used in order to model either whole sequences or subsequences associated with class labels [16, 511]. In these methods, a hidden Markov model is used in order to create a generative model which is specific to each class. The major difference from the methodology discussed in this chapter is that the parameters of the HMM are learned separately for each class. This is done by using only the sequences

specific to a particular class in order to train the relevant model. For a given test sequence, the evaluation algorithm is used to determine the identity of the best fit class. In the rare class scenario, the cost-sensitive methods of Chapter 7 can be easily generalized to this case. Specifically, let  $q_1 \dots q_m$  be the probabilities generated by the forward algorithm, when applied to each of the  $k$  different HMMs, corresponding to the  $m$  different classes. Let the costs for the  $m$  different classes be  $c_1 \dots c_m$ . Then, the cost-weighted probability  $f_i$  for the  $i$ th class is defined as follows:

$$f_i = \frac{c_i \cdot q_i}{\sum_{j=1}^k c_j \cdot q_j} \quad (10.11)$$

The class with the highest value of  $f_i$  is reported as the relevant one.

### Kernel Support Vector Machines

Finally, one can use kernel support-vector machines by using string kernels to compute the similarities between pairs of strings. The kernel similarity matrix can be used to perform the classification of the sequences. The SVM needs to be modified slightly to account for the rare-class setting. These modifications are discussed in Chapter 7.

### Comparative Studies

A natural question arises as to how the aforementioned classifiers compare with one another. A detailed performance study, which compares different sequence classifiers is provided in [168]. The results suggest that SVM classifiers can work well when they are used in combination with the right feature transformation methods. Furthermore, such classifiers can also provide high-levels of interpretability, when the feature space corresponds to  $k$ -grams extracted from the data.

## 10.6 Conclusions and Summary

---

Discrete sequences are extremely common in virtually all application domains such as biological data, user-generated data, systems diagnosis data and Web logs. This provides a massive source of temporal data, which is very rich, and is also very complex from the perspective of outlier analysis. The complexity of sequence data allows for diverse definitions of sequence outliers. The broad classes of models are similar to continuous time-series data, although the details of the models are different because of the effect of the different data type. As in the case of time-series data, a single position can be identified as a contextual outlier, or a set of positions may be identified as a collective outlier. Contextual outlier-detection methods typically use Markovian techniques and are significantly less computationally complex. This chapter also discusses some of the more common models and techniques for identifying collective outliers in sequence data. These are distance-based methods, frequency-based methods, and hidden Markov models. Many of these methods can also be generalized to the supervised case, when labels are available.

## 10.7 Bibliographic Survey

---

The problem of sequence outlier detection is useful in the context of a wide variety of applications such as host-based intrusion detection, system fault detection, biological sequences,

and Web click-streams [338, 374, 484, 506, 531]. A general survey on sequence outlier detection may be found in [126]. Another recent survey [232] and a related book [231] provide an integrated treatment of time-series and discrete-sequence outlier detection.

Outliers are of two types corresponding to *position outliers* and *combination outliers*. Position outliers correspond to specific events (symbols) in the sequence (string), which may be considered an outlier. Combination outliers correspond to unusual subsequences of symbols in the sequence, which differ from the overall trends in the data. Position outliers are computed using either Markovian techniques [396, 588] or rule-based systems [148, 588], which are equivalent. The short-memory property [465] can be used in order to predict the value of a sequence from the use of the last  $k$  positions. These models can be considered the discrete equivalent of continuous regression-based predictive models discussed in Chapter 9. The use of rule-based methods for finding position outliers is discussed in [349, 148]. The use of a suffix-tree to represent all order- $k$  (or less) states for outlier prediction was proposed in [387]. Sparse Markov transducers, which are constructed using probabilistic suffix trees, are presented in [189]. The simplest Markov model of the first order is proposed in [588] and a  $k$ th order Markovian model is presented in [396]. A discussion of pruning techniques for improving the efficiency of predictive Markovian models is provided in [169]. Probabilistic suffix trees [524] can be used in order to greatly improve the speed of Markovian techniques. A method for determining outliers based in information theoretic measures was proposed in [312]. Given a sequence, the idea is to segment it into two parts, and determine the segment that has the larger MDL (Minimum Description Length). This segment is assumed to be more noisy and contain the outliers. This process is recursively repeated, until the appropriate outlier position in the sequence is determined. The actual computation of the description length is based on the concept of Kolmogorov complexity.

Combination outliers typically use windowing techniques in which comparison units are extracted from the sequence for the purpose of analysis [197, 272]. Proximity-based methods for outlier detection require the efficient computation of similarity functions in sequence data, such as the longest common subsequences [279] and the edit distance [239]. A number of different similarity functions are discussed in [198, 338, 339]. These methods are typically used in conjunction with window-based methods in order to compare short subsequences of the test sequence with the training sequence. Frequency methods provide a more effective way of computing the surprise of a user-specified comparison unit between a training and test sequence. The TARZAN algorithm [310] uses suffix trees in order to compute the surprise level for a user-specified comparison unit. Proximity methods are also used for point-anomaly detection with the use of  $k$ -nearest neighbor methods [127],  $k$ -medoid based clustering [103, 104], and probabilistic suffix tree-based clustering [581]. Markov Models for the detection of significant episodes and change points in sequences are proposed in [242, 501].

Hidden Markov models are a popular method for finding combination outliers. A tutorial on hidden Markov models may be found in [454]. While they support the direct determination of the outliers with the use of the underlying fit probabilities, methods are also designed to mine the state transition sequences with window-based methods for anomaly detection [197, 612]. Hidden Markov Models have also been used for point anomaly detection of small sequences [111, 500] by using a mixture of hidden Markov models.

Outlier detection is also a challenging problem in the context of complex sequences in which individual points are sets rather than symbols. Most methods for novelty detection in text, categorical, and market-basket streams can be generalized to this case [29, 122]. The techniques for first-story detection in text can also be applied to set-based sequential data [47, 48, 622].



Supervised methods are used frequently in a variety of applications such as intrusion detection, when labeled data is available [185, 216, 217, 561, 545]. Reviews of sequence classification methods can be found in [11, 575]. Position classification in sequences is useful for many natural language applications [51, 337], in which the specific properties of positions in sentences need to be learned. The most common method for sequence classification is use of feature extraction methods such as  $k$ -gram extraction and wavelets [13, 354]. Hidden Markov models are a natural technique for the problem of classification [16, 511]. An experimental evaluation of methods for sequence classification may be found in [168].

## 10.8 Exercises

---

1. Consider the discrete sequence denoted by ACGTACTACGTACGTACGT.
  - Construct an order-1 Markovian model in order to determine the position outliers. Which position is found as the outlier?
  - Now create an order-2 Markovian model in order to determine the position outliers. Which position is found as the outlier?
2. Determine all order-1 rules from the discrete sequence of Exercise 1. Which position is found as the outlier?
3. For the discrete sequence of Exercise 1, determine all subsequences of length 2. Use a frequency-based approach to assign outlier scores to objects. Which pairwise sequence of length 2 should be considered a combination outlier?
4. Repeat Exercise 3 with sequences of lengths 3, 4, and 5. What happens to the relative outlier scores with increasing subsequence length?
5. Repeat Exercises 3 and 4 with the use of a distance-based approach.
6. Construct a black box 2-state hidden Markov model in order to learn the state probabilities of the sequence in Exercise 1. Write a computer program to learn the state transition and symbol emission probabilities. Now apply the model to extracted subsequences of length 2. Which subsequences are predicted as the outliers by the model?

---

---

# Chapter 11

## Spatial Outlier Detection

---

---

“Time and space are modes by which we think and not conditions in which we live.” – Albert Einstein

### 11.1 Introduction

---

Spatial data is a contextual data type in which one can clearly distinguish between two different types of attributes, one of which is spatial. These two types of attributes are as follows:

- **Behavioral attributes:** This is the attribute of interest that is measured for each object. For example, this attribute could correspond to sea-surface temperatures, wind speeds, car speeds, disease outbreak numbers, the color of an image pixel, and so on. It is possible to have more than one behavioral attribute in a given application. Thus, in many applications, this attribute is non-spatial because it measures some quantity of interest at a given location. However, in some data types such as trajectories, it is possible for the behavioral attribute to be spatial.
- **Contextual attributes:** In many spatial data types, the contextual attribute is spatial, although it might not be spatial in some occasional cases (such as trajectories in which the context is temporal). Sea-surface temperatures, wind speeds, and car speeds are often measured in the context of specific spatial locations. Spatial context is often expressed in terms of coordinates, which typically correspond to two or three numerical values. In some cases, the contextual attributes may be expressed at the granularity of a *region of interest*, such as a county, ZIP code, and so on.

Spatial data shares a number of similarities with time-series data in being a contextual data type. In fact, it is often possible for the spatial and temporal attributes to occur in various combinations of behavioral and contextual attributes. Such data is also referred to as *spatiotemporal* data. For example, in some applications, such as hurricane tracking, the contextual attributes are both spatial and temporal. A particularly unusual case is that

of trajectory data in which the behavioral attributes are spatial, whereas the contextual attributes are temporal. Thus, it is clear that different types of semantics and applications can be captured depending on whether the spatial attributes are contextual or behavioral. In general, there are two key settings for spatial data:

1. **The spatial attribute is contextual:** In this setting, some quantity of interest is measured at various spatial locations. Often there might be other contextual attributes. For example, the time attribute might be temporal. In such cases, one might be interested in determining important spatiotemporal anomalies (or events) based on the underlying dynamics. For example, the dynamics of behavioral attributes such as humidity, wind speeds, sea-surface temperatures, and pressure can be used in order to identify and predict anomalous weather events.
2. **The spatial attribute is behavioral:** The most common example is that of trajectory data. In fact, trajectories can also be viewed as a special case of multivariate temporal data. For example, a 2-dimensional *real-time* trajectory-mining application can be modeled as a bivariate time series in which the  $X$ -coordinate and  $Y$ -coordinate is each a time series.

Trajectories are also analyzed in the *offline* setting. In the offline shape-analysis scenario, anomalies may correspond to unusual shapes irrespective of their temporal provenance. In the latter case, the temporal aspects of the problem are limited. Therefore, trajectory-based applications can be modeled in multiple ways, depending on the scenario at hand.

In the setting in which the spatial attributes are contextual, outliers are objects that have very different *behavioral* attribute values from those of their surrounding *spatial* neighbors. Thus, *spatial continuity* plays an important role in the identification of anomalies, just as temporal continuity is important in time-series outlier detection. The fundamental principle of spatial continuity is as follows [550]:

“ Everything is related to everything else, but nearby objects are more related than distant objects.”

In the setting in which the spatial attributes are behavioral, the contextual attribute is often temporal. This corresponds to trajectory data, which is a form of multivariate time-series data. Therefore, the techniques of Chapter 9 can be used.

Some examples of spatial applications are as follows:

- **Meteorological data:** Numerous weather parameters are measured at different geographical locations, which may be used in order to predict anomalous weather patterns in the underlying data [615].
- **Traffic data:** Moving objects may be associated with many parameters such as speed, direction, and so on. In many cases, such data is also spatiotemporal, since it has a temporal component. Finding anomalous behavior of moving objects [102] can provide numerous insights. For example, the discovery of anomalous taxi-trajectories can be used to discover greedy and dishonest taxi-drivers [118, 137].
- **Earth science data:** The land-cover types at different spatial locations may be the behavioral attributes. Anomalies in such patterns provide insights about anomalous trends in human activity such as de-forestation or other anomalous vegetation trends [341].

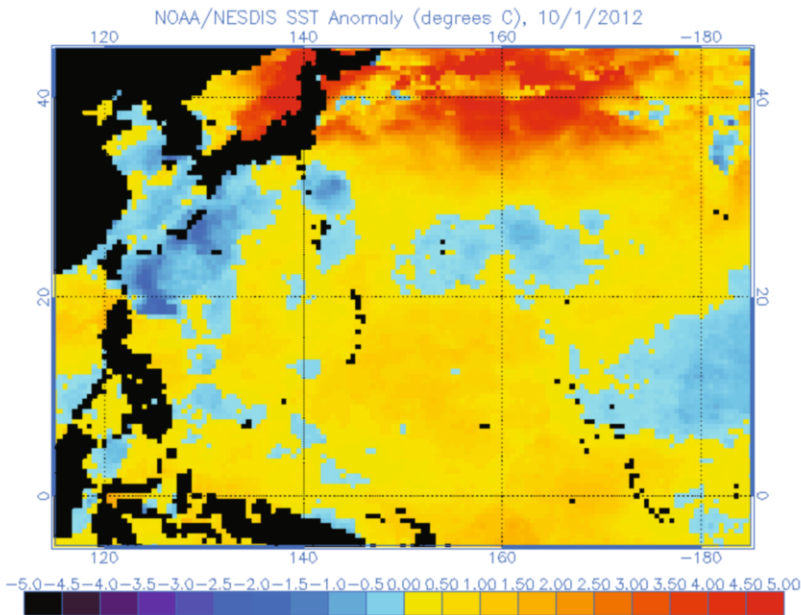


Figure 11.1: Sea surface temperature anomalies. Source: NOAA Satellite and Information Service

- **Disease outbreak data:** Data about disease outbreaks is often aggregated by spatial locations such as ZIP code and county. Anomalous trends in such data [568] can provide information about the causality of the outbreaks.
- **Medical diagnostics:** Magnetic resonance imaging (MRI) and positron emission tomography (PET) scans are spatial data in two or three dimensions. The detection of unusual localized regions in such data can help in detecting diseases such as brain tumors, the onset of Alzheimer disease, and multiple sclerosis lesions [251, 448, 505, 553].
- **Demographic data:** Demographic attributes such as age, gender, race, and salary can be used in order to identify demographic anomalies. Such information can be useful for target-marketing applications.

As in the case of temporal data, *abrupt changes in the behavioral attribute that violate spatial continuity* are used to identify contextual anomalies. For example, consider a meteorological application, in which sea-surface temperatures and pressure are measured. Unusually high sea-surface temperature in a very small localized region is a hot-spot which may be the result of volcanic activity under the surface. In this case, spatial continuity is violated by the attribute of interest. Such attributes are often tracked by meteorologists in real-time. In Figure 11.1, a color-coded map of the sea-surface temperatures on October 1, 2012 from the *NOAA Satellite and Information Service* is illustrated. Unusually high-temperature anomalies are illustrated in red, whereas unusually low-temperature anomalies are illustrated in blue.

In *spatiotemporal* data, both spatial and temporal continuity are used for modeling. For example, a sudden change in the velocity of a few cars in a small localized region may suggest

the occurrence of an accident or other anomalous event. Similarly, evolving events such as hurricanes and disease outbreaks are spatiotemporal in nature. Spatiotemporal methods for outlier detection [141, 142] are significantly more challenging because of the additional challenge of modeling the temporal and spatial components jointly.

There are two main characteristics of spatial data that are commonly leveraged in outlier detection algorithms:

- **Spatial autocorrelations:** This corresponds to the fact that behavioral attribute values in spatial neighborhoods are closely correlated with one another. However, unlike temporal data, where future values of the time-series are unknown, the values in all spatial directions of a data point can be used. Note that spatial autocorrelations are *exactly* analogous to the temporal autocorrelations that are leveraged in time-series autoregressive (AR) models. Refer to Chapter 9 for details of AR models.
- **Spatial heteroscedasticity:** This corresponds to the fact that the variances of the behavioral attribute depend on spatial location [523].

Whereas the first property is the primary criterion for outlier analysis and is used universally, the second is used more occasionally. Nevertheless, it has proven to be useful in many scenarios. This is because when certain regions are likely to have greater variance as a matter of expectation, then abrupt changes in those regions are less likely to be significant. Such insights have led to local methods [523], which are based on ideas derived from local density-based methods (LOF) [96].

Most of the work on spatial outliers is about finding *abrupt changes* which violate spatial auto-correlations. Such outliers are *contextual* outliers, for which numerous methods have been proposed in the literature. The primary ones among them use *variations of the behavioral attribute* within a neighborhood to define outliers. Such outliers use either multi-dimensional analysis methods or graph-based methods. In addition, many of the temporal autocorrelation methods discussed in the previous chapter can be generalized to the spatial domain. It is sometimes useful to intuitively visualize the key outlier points. The spatial nature of the data also lends itself to more intuitive visualization methodologies. Two examples of such methodologies are *variogram clouds* and *pocket plots* [247, 424]. The former will be described in this chapter.

As in the case of time-series databases, it is also useful to find unusual *shapes of behavioral-attribute patterns* in a database of multiple spatial distributions. For example, the color distribution in an image or MRI scan may correspond to an unusual shape, when compared to other images in the database. Such images may be identified as *collective* outliers in the context of spatial data.

Supervised methods are also very useful in the spatial domain, where it is desirable to determine unusual shapes from multiple spatial patterns. For example, while many conditions such as weather patterns of interest, or brain tumors in MRI scans may be rare on a *relative* basis, a significant amount of training data may be available on an *absolute* basis for modeling purposes. In medical applications, large numbers of pathological examples are sometimes available for modeling purposes. Similarly, many examples of pathological patterns of unusual shapes may be available in meteorological and earth science applications. In such cases, it is useful to utilize supervision for the purposes of outlier detection. Supervised methods are particularly useful in the context of outlier detection in such cases because of the unusually high complexity of a database containing multiple spatial patterns. Such methods are closely related to topics such as image classification.

A close relationship exists between temporal and spatial outlier detection because both methods use concepts of *behavioral attribute continuity with respect to one or more contextual attributes*. The main difference lies in the fact that spatial contextual attributes are often multidimensional, whereas time is a single attribute. Furthermore, time is unidirectional, in which only values in the past are known at a given point, whereas spatial attributes are usually known in the different directions of all axes. Nevertheless, in many applications, these differences are not significant enough to invalidate the applicability of similar methods in both settings. While recent work has adapted temporal techniques to some spatial applications such as anomalous image shape detection [565], many other temporal techniques have the potential for use in the spatial domain. This chapter will point out the different temporal techniques that are adaptable to the spatial domain. In some cases, these temporal methods are not adaptable, especially when the spatial context is not specified in a quantitative way (e.g., coordinates). For example, the spatial attribute may be specified as a categorical attribute such as a county or ZIP code.

This chapter is organized as follows. In the next section, the case in which spatial attributes are contextual will be discussed. Many algorithms such as multidimensional methods, graph-based methods, and autoregressive methods will be studied. Methods for discovering anomalous shapes will be discussed in this section. The setting in which both spatial and temporal attributes are contextual will be studied in section 11.3. In section 11.4, we will study the case in which the spatial attributes are behavioral. This setting corresponds to the case of trajectory data. We will also study the connections of this approach with multivariate time-series analysis. The conclusions and summary are presented in section 11.5.

## 11.2 Spatial Attributes are Contextual

---

In this section, we will discuss the case in which the spatial attributes are contextual. This setting corresponds to weather data, temperature data, disease outbreaks, and so on. Different types of models, such as neighborhood-based models, graph-based models, and autoregressive models will be discussed.

### 11.2.1 Neighborhood-Based Algorithms

Neighborhood-based algorithms can be very useful in the context of a wide variety of tasks. In these algorithms, abrupt changes in the spatial neighborhood of a data point are used in order to diagnose outliers. Such algorithms depend on the exact way in which the spatial neighborhood is defined, the function used to combine these neighborhood values into an expected value, and the computation of the deviations from the expected values. The neighborhood may be defined in many different ways [2, 324, 376, 487, 488, 489, 490], depending on the nature of the underlying data:

- **Multidimensional neighborhoods:** In this case, the neighborhoods are defined on the basis of multidimensional distances between data points.
- **Graph-based neighborhoods:** In this case, the neighborhoods are defined by linkage relationships between spatial objects. The linkage relationships might be defined by a spatial domain expert with an understanding of spatial neighborhoods. Such neighborhoods may be more useful in cases in which the location of the spatial objects may not correspond to exact coordinates (e.g., county or ZIP code), and graph-representations provide a more general modeling tool.

This section will study methods for neighborhood-based outlier detection with the use of multidimensional and graph-based methods.

### 11.2.1.1 Multidimensional Methods

While traditional multidimensional outlier detection methods (e.g., LOF) can also be used to detect outliers in spatial data, such methods do not distinguish between the contextual attributes and the behavioral attribute. Therefore, such methods are not optimized for outlier detection in spatial data, especially in cases in which the outliers are defined on the basis of the behavioral attribute within a specific contextual vicinity.

Many methods define the contextual vicinity (spatial neighborhood) with the use of multidimensional distances on the spatial attributes. Thus, the contextual attributes are used for determining the  $k$  nearest neighbors, and the deviations on the behavioral attribute values are used in order to predict outliers. A variety of distance functions can be used on the multidimensional spatial data for determination of proximity. The choice of the distance function is important, because it regulates the choice of the neighborhood for comparative purposes. For a given spatial object  $o$  with behavioral attribute value  $f(o)$ , let  $o_1 \dots o_k$  be its  $k$ -nearest neighbors. Then, the predicted value  $g(o)$  of the behavioral attribute of object  $o$  may be computed using the neighborhood mean:

$$g(o) = \sum_{i=1}^k f(o_i) / k$$

Alternatively, one can use the neighborhood median instead of the mean to reduce the impact of extreme values. Then, for each data object  $o$ , the value of  $f(o) - g(o)$  represents a deviation from predicted values. The extreme values (positive or negative) among these deviations may be computed using a variety of methods discussed in Chapter 2. These are reported as outliers.

### Local Outliers

An observation in [523] is that all local deviations are not equally important from the perspective of outlier analysis. For example, consider the case in which sea-surface temperatures are measured at different spatial locations. In some spatial regions, the changes in temperatures may naturally show larger variations than others. Therefore, the same variation cannot be treated with equal importance in all regions. Specifically, the outlier scores in high-variance regions need to be suppressed. In such cases, it may be useful to quantify the changes around a data point in a local way. For example, instead of using the value of  $f(o) - g(o)$  as discussed above, it is possible to use a normalized value of  $\frac{f(o) - g(o)}{L(o)}$ , where  $L(o)$  represents a *spatially local* quantification of the deviations around  $o$ . For example,  $L(o)$  could represent the standard deviations of the behavioral attribute values in the spatial neighbors of  $o$ .

In practice, a variety of different methods could be used in order to characterize the local deviations around the spatial object  $o$ . The work in [523] has also defined a deviation measure *SLOM* which is based on the LOF method for defining local spatial outliers. This approach is sensitive to the spatial heteroscedasticity of the data, in which the specific variance of the behavioral attribute in a particular spatial locality is carefully accounted for in constructing the outlier score.

### 11.2.1.2 Graph-Based Methods

In graph-based methods, spatial proximity is modeled with the use of links between nodes. Thus, nodes are associated with behavioral attributes, and strong variations in the behavioral attribute across neighboring nodes are recognized as outliers. Graph-based methods are particularly useful when the individual nodes are not associated with point-specific coordinates but may correspond to regions of arbitrary shape. In such cases, the links between nodes can be modeled on the basis of the neighborhood relationships between the different regions.

Graph-based methods define spatial relationships in a natural way, since semantic relationships can also be used to define neighborhoods. Typically, a spatial domain expert might construct the neighborhood graph. This allows for a more general view of neighborhoods. For example, two objects could be connected by an edge, if they are in the same *semantic* location such as a building, restaurant, or office. In many applications, the links may be weighted on the basis of the strength of the proximity relationship. For example, consider a disease-outbreak application in which the spatial objects correspond to county regions. In such a case, the weights of the links could correspond to the lengths of the boundaries between pairs of adjacent regions.

Let  $S$  be the set of neighbors of a given node  $o$ . Then, the concept of spatial continuity can be used to create a *predicted* value of the behavioral attribute based on those of its neighbors. The weight of the links between  $o$  and its neighbors can also be used in order to compute the predicted values as either the weighted mean or median on the behavioral attribute of the  $k$  nearest spatial neighbors. For a given spatial object  $o$ , with behavioral attribute value  $f(o)$ , let  $o_1 \dots o_k$  be its  $k$  linked neighbors based on the relationship graph. Let the weight of the link  $(o, o_i)$  be  $w(o, o_i)$ . Then, the linkage-based weighted mean may be used to compute the predicted value  $g(o)$  of the object  $o$ .

$$g(o) = \frac{\sum_{i=1}^k w(o, o_i) \cdot f(o_i)}{\sum_{i=1}^k w(o, o_i)} \quad (11.1)$$

Alternatively, the weighted median of the neighbor values may be used to compute  $g(o)$ . As the true value,  $f(o)$ , of the behavioral attribute is known, it can be used to model the deviations of the behavioral attributes from their predicted values. Specifically, the value of  $f(o) - g(o)$  represents a deviation from the predicted values. Extreme value analysis can be used on these deviations in order to determine the spatial outliers as those objects for which the value of  $f(o) - g(o)$  is unusually positive or negative. This process is identical to what was discussed before for the multidimensional case. As in all outlier analysis algorithms, a variety of extreme-value analysis methods of Chapter 2 can be used on these deviations in order to determine the outliers. The nodes with high values of the normalized deviations may be reported as outliers.

### 11.2.1.3 The Case of Multiple Behavioral Attributes

In many cases, multiple behavioral attributes may be associated with the contextual attributes. For example, in a meteorological application, both temperature and pressure values may be available with the spatial attributes. In these cases, the deviations may be computed on each behavioral-attribute, and then these values need to be combined into a single deviation value, which provides the final outlier score. For this purpose, any of the multivariate extreme value analysis methods in section 2.3 of Chapter 2 may be used. In particular, the



work in [138] has proposed the use of the Mahalanobis distance-based method of Chapter 2 for extreme-value analysis.

## 11.2.2 Autoregressive Models

Spatial data shares a number of similarities with temporal data. Both types of data measure a behavioral attribute (e.g., temperature) with respect to a contextual attribute (e.g., space or time). In many scenarios, spatial data is available in the form of coordinates, and the values of the behavioral attribute may be available at *each possible spatial reference point in the grid*. Such data arises commonly in weather contour maps, images, MRI scans, and so on. In cases *in which the data is completely specified at most points in the grid*, it is possible to use autoregressive models in order to determine unusually large deviations in the data in an analogous way to the temporal scenario.

Let  $X_{t_1, t_2}$  be the value of the behavioral attribute at the spatial location  $(t_1, t_2)$ . In the temporal autoregressive model, the predicted value of the behavioral attribute is based on a 1-dimensional window of *past* history of length  $p$  (see section 9.2.1 of Chapter 9). In the 2-dimensional spatial scenario, this can be generalized to a square window of size  $(2 \cdot p + 1) \times (2 \cdot p + 1)$ , with  $p$  coordinates in either direction. More generally, in the case of 3-dimensional spatial data, one can use a cube of size  $(2 \cdot p + 1) \times (2 \cdot p + 1) \times (2 \cdot p + 1)$ . As in the case of the 1-dimensional autoregression for temporal data in section 9.2.1 of Chapter 9, a 2-dimensional model can be defined as follows.

$$X_{t_1, t_2} = \sum_{i=-p}^p \sum_{j=-p}^p a_{ij} \cdot X_{t_1-i, t_2-j} + c + \epsilon_{t_1, t_2}$$

The value of  $a_{00}$  is always set to 0, so that it is missing from the aforementioned summation. This is done in order to avoid the trivial solution in which a spatial value is used to predict itself. The values of  $a_{ij}$  need to be learned from the underlying training data. Thus, such an equation can be created for each value of  $(t_1, t_2)$ . When the number of available spatial-coordinates is much larger than  $(2 \cdot p + 1) \times (2 \cdot p + 1)$ , this is an over-determined system of equations in which the least-squares error is minimized to determine the optimal values of the coefficients  $a_{ij}$ . The optimal values of the coefficients are computed with least-squares regression according to the approach discussed in section 3.2.1 of Chapter 3. Thus, the process of computing the regression coefficients is very similar to that in temporal data.

In the aforementioned system of equations, the value of  $c$  is a constant, and the value of  $\epsilon_{t_1, t_2}$  represents the noise, or the *deviation* from the expected values. Large absolute values of this deviation represent the anomalies in the underlying data. These values are assumed to be independent and identically distributed random variables, which are drawn from a normal distribution. Thus, the extreme-value analysis methods of Chapter 2 can be used on these values to identify the abnormal locations.

The aforementioned discussion provides a generalization of Autoregressive (AR) models from temporal to spatial data for illustrative purposes. In practice, it is possible to generalize *all* the regression models (ARMA, ARIMA, PCA) to the spatial scenario, by using the appropriate slice of values from the spatial data. As in the temporal case, it is even possible to create multivariate spatial regression models, where multiple behavioral attributes are available. Typically such behavioral attributes may be correlated with one another (e.g., temperature and humidity), and it is desirable to determine unusually large local deviations with the help of multivariate correlations. Refer to section 9.2.2 of Chapter 9 for a discussion of multivariate time-series regression models. The spatial generalization to multiple behavioral attributes is similar to this case.

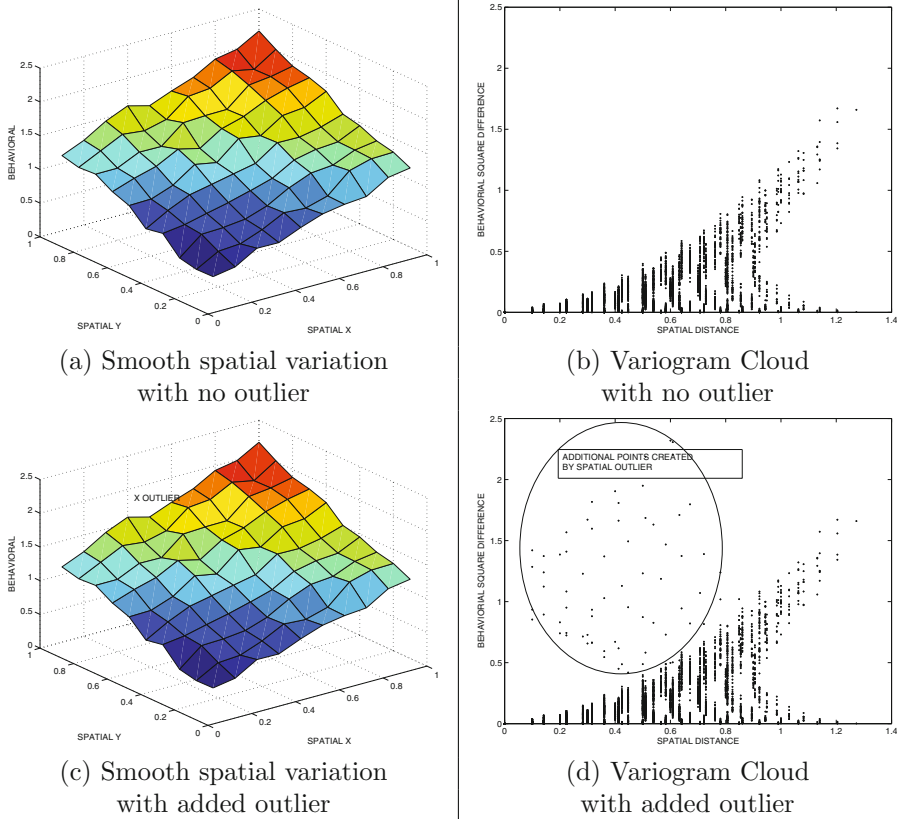


Figure 11.2: Effect of adding spatial outlier to variogram cloud

Although the autoregressive nature of spatial data is widely recognized, such models have rarely been used for anomaly detection in the research literature. This is partially a result of the high computational complexity of autoregressive models with an increasing number of coefficients. Such models also cannot easily handle spatial data which is incompletely specified by spatial location, region-based locations or semantic locations. Nevertheless, such models can be very useful in many scenarios such as image analysis or weather patterns in which large amounts of reasonably complete data are available for analysis. In such cases, the statistical robustness of these methods is likely to be higher than simpler neighborhood-based models.

### 11.2.3 Visualization with Variogram Clouds

A number of visualization techniques such as pocket plots and variogram clouds are used in order to visualize spatial outliers. The latter will be discussed detail here, because of their relative popularity. Since spatial outliers are based on *disagreement* in the continuity of the behavioral attribute *in relation* to the spatial attribute, a natural method to visualize this would be to create a scatter plot between the pairwise spatial distances and the pairwise behavioral attribute (square) deviation. The spatial distance is simply the Euclidean distance between a pair of points. The behavioral attribute deviation is defined as half the squared distance between the behavioral attribute values. A scatter plot is created between

the spatial distances on the  $X$ -axis, and the behavioral squared deviations on the  $Y$ -axis for every pair of points in the data set. The idea is that smaller spatial distances correspond to smaller behavioral attribute variances and vice versa. In particular, large variations of the behavioral attribute for smaller spatial distances should be considered deviants. Such points on the variogram cloud can be traced back to the original data to determine pairs of points that are spatially close but behaviorally different.

In order to illustrate the impact of outliers on variogram clouds, a synthetic example will be used. First, the data set for the variogram clouds of Figure 11.2 will be described. In this case, a grid of 100 points on the spatial plane are used with coordinates drawn from  $X, Y, = 0.1, 0.2 \dots 1.0$ . The value of the behavioral attribute  $Z$  is generated as follows:

$$Z = X + Y + \epsilon$$

Here,  $\epsilon$  is a small amount of noise, which was randomly generated from the uniform distribution in  $[0, 0.2]$ . This spatial variation of the attribute is quite smooth, since the noise is small relative to the global variation in values of the behavioral attribute. The spatial profile of the generated data is illustrated in Figure 11.2(a), and the corresponding variogram cloud is illustrated in Figure 11.2(b). It is evident that low values of the spatial distance always correspond to low deviations of the behavioral attribute. While it is possible for high spatial deviations to be related to low behavioral deviations, the converse is not true.

Subsequently, a single outlier is added to the data by distorting the behavioral attribute of one of the spatial values in the grid of Figure 11.2(a). The corresponding outlier is shown in Figure 11.2(c), and is marked explicitly. Note that the spatial data sets in Figures 11.2(a) and 11.2(c) are virtually identical, with the only difference between them being the outlier created by a distorted behavioral attribute value. The corresponding variogram cloud is illustrated in Figure 11.2(d). It is evident that in this case, a new set of points have been added to the variogram cloud in which significant behavioral deviations exist even at low spatial distances. Multiple such deviant points are created corresponding to the different data points in the immediate spatial locality of the added outlier. Such points can easily be isolated visually and linked back to the original points in the data. Thus, this approach provides an easy and intuitive way of visually identifying the spatial outliers.

One challenge in creating a variogram cloud is the high computational complexity. Note that a single point exists in the variogram cloud for each pair of points in the original data. Therefore, the number of points in the variogram cloud scales quadratically with the number of points in the spatial data. This can make the approach rather slow, when the number of data points is large. In practice, it is difficult to create a variogram cloud for situations in which the data contains a few hundred thousand spatial data points. This can be a significant problem, since spatial data sets are often quite large in practice. Nevertheless, one can construct variogram clouds at rougher levels of granularity.

One observation about the variogram cloud is that it is not always necessary to represent *every pair* of points on the plot. Pairs of data points that are spatially very far away from one another add little insight about the outlier behavior. Therefore, each spatial dimension can be discretized into ranges, and this creates a 2-dimensional grid in the data. The pairwise relationships between all spatial points *within* this grid can be used in order to create the variogram cloud. This significantly reduces the computational complexity of creating the variogram cloud. For example, consider the case in which the original data set contains  $N$  points. Assume that these  $N$  points are discretized into a  $t \times t$  grid with approximately<sup>1</sup>

<sup>1</sup>In practice, the different grid regions may contain a different number of data points because of spatial correlations. However, in many applications such as image data, pixels may be available for every spatial coordinate. Therefore, the division into grids will create a uniform division of the data points.



Figure 11.3: NASA satellite image of hurricane *Fran*: The anomalous shape is characteristic of a hurricane

$N/t^2$  data points in each. Then, the computational complexity of creating a variogram cloud for each grid is  $O(N^2/t^4)$ . Of course, since there are a total of  $t^2$  grids, the aggregate computational complexity is  $O(N^2/t^2) < O(N^2)$ . This provides a speedup factor of  $O(t^2)$ . It is noteworthy that in this case, an optimistic scenario was assumed in which the data points were uniformly distributed into the grid structure. It can be shown theoretically that a speedup factor of at least  $t$  can be obtained with this approach. This is because the speed up achieved with a grid partitioning into  $t \times t$  ranges will always be better than the discretization along only one dimension into  $t$  ranges with an equal number of data points. A significant speedup may be obtained even for modest values of  $t$ , without significant reduction in the quality of the visual discrimination between the outliers and the normal points.

#### 11.2.4 Finding Abnormal Shapes in Spatial Data

The problem of finding unusual shapes in spatial data finds numerous applications such as image analysis. For example, the detection of unusual shapes from brain PET scans or MRI scans can help detect conditions such as tumors, Alzheimer, and sclerosis [448, 553], or can help identify anomalous conditions such as hurricanes from weather maps. For example, consider the satellite image illustrated in Figure 11.3. The anomalous shape in the image corresponds to hurricane *Fran*, which was a large destructive hurricane. This hurricane hit Cape Fear in North Carolina on September 1996, and it can easily be identified by its characteristic shape in the satellite image. However, such a shape may not appear in other similar satellite images on normal days, and is therefore an unusual event. Another example from the medical domain is illustrated in Figure 11.4, in which the PET scans from a normal person and an Alzheimer patient are presented. The colored regions correspond to the uptake of the radioactive tracer administered in a PET scan (behavioral attribute). It is evident that this behavioral attribute shows very different spatial behavior for normal

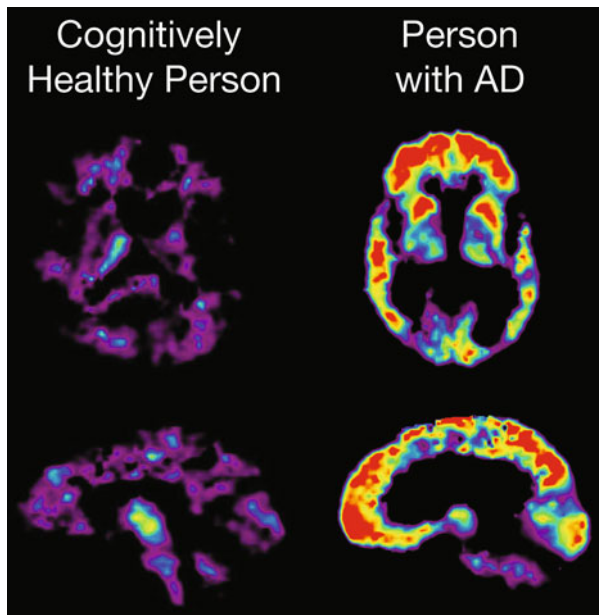


Figure 11.4: PET scans of brain for cognitively healthy person versus an Alzheimer patient: Image courtesy of the National Institute on Aging/National Institutes of Health

and diseased individuals.

Abnormal shapes in spatial data can be identified with both unsupervised and supervised methods. In general, supervised methods are always desirable because they allow the encoding the domain knowledge and training data in the prediction process. In the following, both unsupervised and supervised methods will be discussed. Many unsupervised and supervised methods use various feature transformation techniques such as contour extraction, multidimensional feature extraction, and multidimensional wavelet transformation. These different transformation techniques are relevant to different application domains, and should therefore be judiciously selected, depending on the problem at hand.

#### 11.2.4.1 Contour Extraction Methods

In their simplest form, shapes can be modeled by the contours (or boundaries) of regions with particular ranges of behavioral attribute values in the data. For example, in the case of Figure 11.3, the boundaries of such regions can be extracted by direct analysis of sensor and satellite readings such as pressure, cloud cover, temperature, wind speed, and humidity or from the (already processed) color histogram of the corresponding image.

A key simplification for shape analysis is that the contours of an object can be converted into a time-series by using a feature-engineering trick. One possible way to achieve this is to use the distance from the centroid of the object to the boundary of the object, and compute a sequence of real numbers derived in a clockwise sweep of the boundary [602]. This yields a time series of real numbers, and is referred to as the *centroid distance signature*. This transformation can be used to map the problem of mining shapes to that of mining time-series, a domain which is much more easier to address from an analytical perspective. For example, consider the elliptical shape illustrated in Figure 11.5(a) with centroid denoted

by  $X$ . Then, the time-series representing the distance from the centroid, by using 360 different equally spaced angular samples, is illustrated in Figure 11.5(b). In this case, the sample points are started at one of the major axes of the ellipse. Starting the sampling at a different place or rotating the shape (with the same angular starting point) causes a cyclic translation of the time-series. The resulting time-series may be normalized in different ways depending on the needs of the application at hand:

- If no normalization is performed, the outlier analysis approach is sensitive to the absolute size of the underlying object. This may be the case in many medical images such as MRI scans, in which all spatial objects are standardized in terms of the scale. Therefore, normalization is not necessary.
- If all time-series values are multiplicatively scaled down by the same factor to unit mean, then such an approach will allow the matching of shapes of different sizes, but will discriminate between different levels of relative variations in the shapes. For example, two ellipses with very different ratios of the major and minor axes will be discriminated well.
- If all time series are translated to zero mean and scaled to unit variance, such an approach will match shapes where *relative* local variations in the shape are similar, but the overall shape may be quite different. For example, such an approach will not discriminate very well between two ellipses with very different ratios of the major and minor axes, but will discriminate between two such shapes with different relative local deviations in the boundaries. Furthermore, noise effects in the contour will be differentially enhanced in shapes that are less elongated. For example, for two ellipses with similar noisy deviations at the boundaries, but different levels of elongation (major to minor axis ratio), the overall shape of the time-series will be similar, but the local noisy deviations in the extracted time series will be *differentially* suppressed in the elongated shape. This can sometimes provide a distorted picture from the perspective of shape analysis. A perfectly circular shape may show unstable and large noisy deviations in the extracted time-series because of image rasterization effects. The solution proposed in [565] is to treat circular shapes specially, although the unintended effects of such normalization may have unusually complex effects across a broader spectrum of shapes.

It is helpful to have a proper domain-specific understanding of the effect of a particular normalization, so that it may be selected in a domain-specific way.

The problem of shape analysis is further complicated by the effect that transformations such as rotations can have on the underlying data. For example, consider the images illustrated in Figure 11.6. All images correspond to the same object, but two of them are rotated with respect to the original shape, and the last is a mirror image of the original shape. It is clear that the rotation makes it more difficult to match the two images, if the time-series representation does not account for the rotation or the mirror image effects of the representation. Errors in matching the two shapes also lead to errors in outlier detection, especially when the outlier detection process uses a proximity-based method. It is important to note that all applications do not necessarily require the accounting of rotations. For example, in an MRI scan, where the correct orientation of the scan is known, such rotational transformations may not be needed.

An immediate observation is that *a rotation of the shape leads to a linear cyclic shifting of the time series generated by using the distances of the centroid of the shape to the contours*

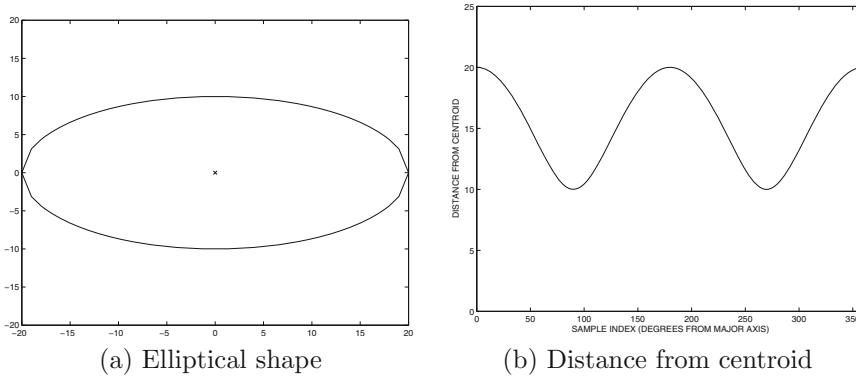


Figure 11.5: Conversion from shapes to time-series



Figure 11.6: Rotation and mirror-image effects on shape matching for outlier analysis

of the shape. For a time series of length  $n$  denoted by  $a_1 a_2 \dots a_n$ , a cyclic translation by  $i$  units leads to the time series  $a_{i+1} a_{i+2} \dots a_n a_1 a_2 \dots a_i$ . Then, the *rotation invariant Euclidean distance*  $RIDist(T_1, T_2)$  between two time series  $T_1 = a_1 \dots a_n$  and  $T_2 = b_1 \dots b_n$  is given by the minimum distance between  $T_1$  and all possible rotational translations of  $T_2$  (or vice versa). Therefore, the following is true:

$$RIDist(T_1, T_2) = \min_{i=1}^n \sum_{j=1}^n (a_j - b_{1+(j+i) \bmod n})^2 \tag{11.2}$$

Note that the reversal of a time-series corresponds to the mirror-image of the underlying shape. Therefore, mirror images can also be addressed by using this approach.

The shape discords can then be determined by computing the series whose  $k$ th nearest neighbor distance to its closest neighbor is as large as possible. The top- $r$  such shapes need to be found. As in all distance-based algorithms, a brute-force approach on a database with  $N$  shapes would require  $O(N^2)$  distance computations, unless pruning methods are used.

The major difference between this problem and the unusual time-series shape discovery problem discussed in section 9.3 of Chapter 9 is that the rotational invariant distances are used instead of the Euclidean distances. Furthermore, the distances are computed on whole time-series instead of on subsequences. While it may be possible in theory to use the method of Chapter 9, by making some modifications to address rotational invariance, longer lengths of whole sequences (compared to subsequences), may cause greater challenges in pruning. For example, rotational variations can be addressed by explicitly incorporating rotational variations of the time-series into the database, just as subsequences of a time-series are incorporated into the database for subsequence discord discovery in section 9.3 of Chapter 9. Care needs to be taken in avoiding self-similarity from the same shape during the distance computations, just as self-similarity is avoided in time series discord discovery. Therefore, the techniques in section 9.3 of Chapter 9 can be used in theory in order to find discords. Of course, the addition of multiple rotational variations of the shapes to the database is likely to slow down the discovery process. It also leads to some redundancy in the representation, because all rotational variations of the same object will have the same outlier score.

The work in [565] uses a different pruning method based on LSH-approximations [281] of the symbolic aggregate approximations of the time-series. The overall organization of the approach is similar to the algorithm discussed in section 9.3 of Chapter 9. Both methods first sort the objects by approximate outlier tendency in order to perform the outlier search in an ordered way, which optimizes the pruning behavior. For each object, pruning is performed with approximate nearest-neighbor distances. However, the specific technique used for pruning is different in the two scenarios.

A nested loop approach is used to implement the method. The algorithm examines the candidate shapes iteratively in an outer loop, and progressively improves the estimate of each candidate's  $k$ -nearest neighbor distance in an inner loop. The inner loop essentially computes the distances of the other shapes to the candidate. At the end of the execution of a candidate-specific inner loop, the approach then either includes the candidate in the current set of top- $n$  outlier score estimates, or discards the candidate at some point during the computation of its  $k$ -nearest neighbor in the inner loop. This is referred to as early inner loop termination. This inner loop can be terminated early, when the currently approximated  $k$ -nearest neighbor distance for that candidate shape is less than the score for the  $r$ th best outlier found so far. Clearly, such a shape cannot be an outlier. In order to obtain the best pruning results, the candidate shapes in the outer loop need to be heuristically ordered, so that the earliest shapes examined have the greatest tendency to be outliers. The pruning performance is also best, when the points are ordered in the inner loop, such that the  $k$ -nearest neighbors of the candidate shape are found early. It remains to explain how the heuristic orderings required for good pruning are achieved.

As in the case of time-series subsequences, each time series is mapped onto an LSH word with the use of Symbolic Aggregate Approximation. Assume that the resulting SAX words have length  $m$ . Locality sensitive hashing [281] randomly samples  $s < m$  distinct positions in the SAX representation. Therefore, two SAX words that are more similar are more likely to map to the same string. This is also referred to as the *locality sensitivity property* of the LSH-hashing approach, and the similarity can be robustly quantified by examining the mapping behavior over multiple hash functions. However, this does not account for the rotational invariance of the matching process. In order to account for the possible rotations, a rotational invariant LSH function is defined. This function first picks  $s < m$  position indices randomly, and then samples these  $s$  position indices from all possible  $m$  rotations of the SAX word. Clearly, similar shapes will lead to LSH-based collisions, even in the presence



of rotations. The LSH-hashing process is repeated with multiple hash functions in order to provide greater robustness to the collision-based counts.

For each SAX word, a count is maintained of its number of LSH-based collisions. This provides approximate information about its outlier score. Shapes with smaller counts need to be processed first as candidates in the outlier loop, since they have greater likelihood of being outliers. Furthermore, shapes which collide with one another frequently in LSH-based hashing are more likely to be nearest neighbors. Therefore, shapes that have the largest number of collisions with the current outer loop candidate are examined first in the inner loop for distance computations. This provides the heuristic order of processing in the inner loop. The reader is referred to [565] for a detailed description of the algorithm.

#### 11.2.4.2 Extracting Multidimensional Representations

Multidimensional methods are applicable in cases in which the behavioral attributes are specified along the entire grid of spatial coordinates. In this section, we assume that we are looking for abnormal grids of size  $p \times p$  from the database of one or more spatial images. The first step is to extract all the grids of size  $p \times p$ . Subsequently, the  $p^2$  behavioral values in the grid are converted into a  $p^2$ -dimensional record. Assume that  $N$  such points are extracted. This problem is therefore converted into that of discovering a point outlier in  $p^2$ -dimensional space. Any of the methods discussed in Chapters 1 through 6 can be used in this setting.

#### 11.2.4.3 Multidimensional Wavelet Transformation

A second way of extracting a multidimensional representation is to use a wavelet transformation from a grid of values. Even though wavelets are inherently designed for the case in which there is a single contextual attribute, they can also be generalized to cases in which there is more than one contextual attribute (such as spatial data). In these cases, a Haar wavelet representation can be constructed by alternately dividing the spatial regions along the different axes and computing the differences. A detailed discussion of the multidimensional wavelet transformation is beyond the scope of this book, although a detailed discussion may be found in [33] (cf. section 16.2.2 of Chapter 16).

#### 11.2.4.4 Supervised Shape Discovery

Spatial data is particularly common in many forms of image data such as weather maps, PET scans or MRI scans. For example, consider the case of MRI scans, where 3-dimensional images of the brain may be available for analysis. The anomalies in the data such as tumors and lesions may show up as characteristic regions in the data, which are rare but are nevertheless indicative of specific kinds of abnormalities. In such cases, previous examples of anomalous and normal scans may be available for the purposes of training. Although unsupervised anomaly detection can help outlier analysis up to a point, the use of supervision can increase the sophistication of the analysis by revealing specific *types* of abnormalities. In most applications, at least semi-supervision is used, where examples of normal spatial profiles are available for analysis. The collection of normal examples is typically not very difficult in most application-specific scenarios, since copious examples of normal instances are usually available.

For all forms of shape classification, the actual *representation* of the shape is the most important step. For example, the *centroid distance signature* discussed in this chapter [602] is one possible way of representing the shapes, but by no means the only one. A thorough

review of shape representation techniques may be found in [602]. The shape to time-series transformation discussed in section 11.2.4 of this chapter can be used in order to transform the shape classification problem to the time-series classification problem. Any of a number of methods (such as subsequence-based  $k$ -nearest neighbor methods) can be used for time-series classification in this case. Furthermore, one can also use the multidimensional and wavelet transformations in the previous section, in combination with an off-the-shelf classifier.

Numerous methods for time-series classification may be found in the literature [408, 590]. These methods typically try to determine discriminative shapes of the series (or shapelets) which distinguish the normal and abnormal series. In the context of spatial data, such abnormal series are typically derived from abnormal shapes from a spatial perspective. In the *semi-supervised* case, the distances of the test series to examples of normal profiles can be used in order to create outlier scores for the underlying series. The only distinction from the available methods for time-series analysis is that care must be taken in order to account for different rotational variants of the shape in particular application-specific scenarios.

The problem of supervised classification of unusual shapes is also closely related to the problem of detecting and recognizing specific shapes in images. This problem has been studied extensively in the field of computer vision and image analysis. The problem of supervised shape recognition is an important area of research in its own right, and is beyond the scope of this book. The reader is referred to [69, 115, 375, 602] for a detailed description of such methods for image classification, analysis and change detection in the image domain. The major modification to these methods is the incorporation of rare class detection and cost-sensitive methods into these algorithms, using the methods of Chapter 7. Since many of the algorithms discussed in Chapter 7 are meta-algorithms, they can be used in conjunction with any of the classification techniques in the literature.

#### 11.2.4.5 Anomalous Shape Change Detection

In spatial data such as weather data, PET scans, and MRI scans, unusual changes in the contours of the shapes may be used in order to predict anomalous events. For example, the formation of a hurricane or a tumor over multiple time stamps will show up as an unusual change in the shapes of the corresponding image representations of the weather data or the MRI scan. The determination of such changes is more complex than those of detecting unusual *point changes* in the data. However, the detection of unusual point changes can be a first step towards detecting regions of anomalous change in the data, by clustering the change points in the spatial data. Not all regions of change may necessarily correspond to anomalies. For example, increasing age may create certain characteristic change contours in an PET scan, which should be considered normal. In practice, this problem is not very different from finding unusual shapes in the original data, with the main difference being that the spatial contours of the behaviorally *changing* regions between two snapshots are extracted. The normally occurring changes in the data over time will usually be quite different from the anomalous changes. Therefore, the anomalous contours provide the unusually changing regions in the data.

This broader principle can also be used in conjunction with other outlier detection methods. In general, a differencing operation on two temporal snapshots of the data may be required as a pre-processing step before applying outlier analysis algorithms. Once the differencing step has been applied, the methods in the previous sections can be applied to the set of differenced features. A detailed description of many such change-analysis methods may be found in [115].

### 11.3 Spatiotemporal Outliers with Spatial and Temporal Context

---

Spatiotemporal data is very common in many real applications in which behavioral attribute values are continuously tracked at different spatial locations. For example, consider a chemical factory dumping chemicals in a river. In such cases, the concentrations of chemicals in the water cannot be described by using either only spatial or temporal contextual attributes. Thus, the contextual attributes need to contain *both* spatial and temporal components. Spatiotemporal data is extremely common in all forms of sensor data, in which behavioral attribute readings are continuously transmitted by sensors at different spatial locations. An example is provided in [569], where precipitation data from different spatial locations and times is aggregated. It is desirable to determine localized spatial regions that are also close together in time, whose precipitation values are significantly different from their “neighboring” values. So how should neighboring values be defined in the case of spatiotemporal data?

Virtually all the spatial methods discussed in earlier sections of this chapter can be generalized to spatiotemporal data, as long as the concept of neighborhood is properly defined in order to make it relevant for the spatiotemporal scenario:

- Spatial methods can be used on temporal snapshots of the data in order to determine the relevant outliers at different instants. However, such an approach is incomplete, because it fails to identify violations of temporal continuity.
- Some algorithms have been proposed in order to separately identify spatial outliers and temporal outliers, and then combining the results in order to provide the spatiotemporal outliers [89]. However, the decoupling of spatial and temporal aspects of the problem at an earlier stage is obviously a suboptimal solution.
- Spatiotemporal neighborhoods of data points may be used in order to determine predicted values. Thus, the only difference from purely spatial methods is that the expanded set of contextual attributes are used to define the neighborhoods for analysis and prediction. As in the previous case, deviations from the predicted values can be used to identify outliers. In some techniques such as neighborhood methods, the challenge is to combine the (contextual) distances along the spatial and temporal dimensions in a meaningful way. One simple way of achieving this would be to normalize the standard deviation across each of the contextual attributes to one unit before computation of distances. If desired, weights can be used to provide more importance to one or more of the contextual attributes.

The last of the aforementioned methods is the most general, because it can detect significant changes *both* across spatial and temporal attributes in an integrated and meaningful way. It is also important to note that spatial and temporal continuity may not be equally important, depending upon the underlying application. For example, in an application where precipitation level is the behavioral attribute [569], spatial continuity may be more important than temporal continuity. In such cases, appropriate scaling can be performed on the different dimensions to define neighborhoods in a way that provides greater importance to one or more contextual attributes.

## 11.4 Spatial Behavior with Temporal Context: Trajectories

---

A special case of spatiotemporal data is one in which the spatial attributes are behavioral and the temporal attribute is contextual. This data corresponds to trajectory data, which is encountered commonly in a variety of real-world settings. Such data can be treated as a form of bivariate temporal data by treating the  $X$ -coordinates and  $Y$ -coordinates of each object as the behavioral attributes and time as the only contextual attribute. This results in two related time series at the same instants. Thus, the methods for multivariate time-series data analysis can be applied effectively to such cases. Such analysis, when applied to single time-series, can identify sudden *changes* in trajectory directions and velocity. This can be very useful in detecting information about significant changes in cyclone or hurricane trajectories [117]. In other cases, a database of multiple trajectories may be available, and it is desirable to determine unusual shapes of trajectories. The temporal component is less important in this case, since the trajectories may have been created at different times. In such cases, it is possible to use subsequence analysis on these time-series in order to determine those trajectories which behave very differently from the remaining series by determining time-series of unusual shapes [360]. However, unlike the univariate scenario [360], spatial time-series are at least bivariate, and it is much harder to find unusual shapes in terms of the *combination behavior of the two time series*.

### 11.4.1 Real-Time Anomaly Detection

For the case of real-time change analysis, the prediction-based outlier detection methods discussed in section 9.2 of Chapter 9 can be applied separately on each of the  $X$ -coordinate and  $Y$ -coordinate time series. This results in a residual value along each of the two coordinates. If each of these residuals is modeled as a normal distribution, then the sum of the squares of the  $Z$ -values of these residuals is a  $\chi^2$  distribution with two degrees of freedom. This can provide an outlier score, along with a corresponding probability value, which can be derived from the  $\chi^2$ -distribution. Since a trajectory is a bivariate or trivariate time-series, it is also possible to use multiple time-series regression models from section 9.2.2 of Chapter 9 in order to identify outliers.

### 11.4.2 Unusual Trajectory Shapes

Although real-time *change* analysis of such scenarios can be performed more effectively by using temporal modeling, *unusual shape* detection of trajectories can be best performed by abstracting out the temporal component, and performing the spatial analysis directly on the trajectories. In such cases, each spatial object has a shape, and the difference of this shape to its nearest neighbor trajectories are used in order to determine outliers. Since such trajectories may contain a large number of time-stamps, it may often be difficult to determine outliers on the entire sets of trajectories. In such cases, unusual subsequences of the trajectories may be used in order to identify outliers.

#### 11.4.2.1 Segment-wise Partitioning Methods

Some specific methods such as TROAD have also been proposed in the literature [347] for unusual shape detection in trajectories. In particular, the partition-and-detect framework [347] first partitions the trajectories into a set of sub-trajectories. Note that this is

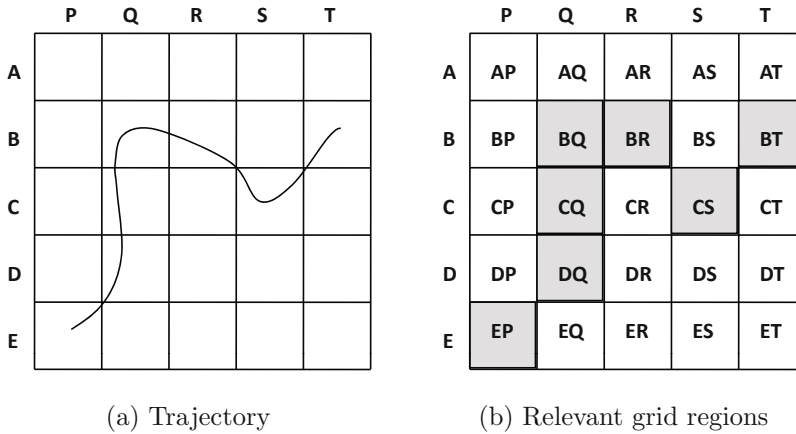


Figure 11.7: Grid-based discretization of trajectories

somewhat analogous to the concept of partitioning time series into subsequences (or finding outliers in subspaces of numerical data), since outliers cannot easily be determined on the full series (with high implicit dimensionality). The sub-trajectories are created with a two-level partitioning which is allowed to be coarse-grained at the higher levels, and fine-grained at the lower level. Subsequently, those sub-trajectories that are not similar to other ones in the data are reported as the outliers. The similarity is measured with the use of both distance-based and density-based methods. Note that the choice of the distance function is critical, and can regulate the nature of the outlier found. For example, a distance function which is sensitive to the *location* of the trajectory is likely to find an outlier based on location of the trajectories. On the other hand, a distance function which is sensitive to the angle between trajectory segments is likely to be sensitive to directions of movement. The precise definition of the distance function is application dependent, though a variety of such functions can be used in conjunction with the partitioned set of sub-trajectories.

The work in [347] defines a *t*-partition as a line segment from the trajectory. Intuitively, this can be considered analogous to comparison-unit schemes discussed in Chapter 10, which are used in the context of sequence data. A *t*-partition is said to be outlying using the variation<sup>2</sup> of the *k*-nearest neighbor distance definition, first proposed in [317]. Intuitively, a *t*-partition is considered an outlier, if a sufficient number of trajectories in the database are not close to it. The definition of closeness is based on measuring the portion of the trajectory, which is close to the *t*-partition. As in comparison-unit schemes for discrete sequences, the results from the different “units” (or partitions) are combined to declare a trajectory as an outlier, if a sufficient number of its partitions are also outlying. Furthermore, the locality sensitive density-based approach of [96] has also been generalized to this case, by creating a density-sensitive outlier score for the trajectories.

#### 11.4.2.2 Tile-Based Transformations

A natural approach that is used to transform trajectory outlier detection to the case of sequence outlier detection is that of tile-based transformation [33]. The basic idea is to discretize the grid-regions of the space into a set of tiles, which are labeled. An examples

<sup>2</sup>That variation fixes the nearest neighbor distance, and computes the required value of *k* rather than the other way around.

of a trajectory together with its tile-based discretization is shown in Figure 11.7. Note that each tile in the grid has an X-label and a Y-label, and the combination of the two provides a unique label to the tile. For example, the upper-left tile has label AP and the lower-right tile has label ET. In this case, the relevant trajectory in Figure 11.7(a) can be converted into the shaded sequence of tiles in Figure 11.7(b). Therefore, the trajectory can now be represented by the following sequence:

EP, DQ, CQ, BQ, BR, CS, BT

For each trajectory in the database, one such sequence can be extracted. Subsequently, the sequence outlier detection methods discussed in Chapter 10 can be leveraged to discover outliers in such cases.

### 11.4.2.3 Similarity-Based Transformations

In many settings, similarity-based transformations can be very useful. The basic idea here is to treat a trajectory as a bivariate time-series. As discussed in Chapter 16 of [33], one can use multivariate similarity functions to compute similarities between pairs of trajectories. For example, multivariate forms of dynamic time-warping similarity functions are available for computing similarities between trajectories. For a database containing  $N$  trajectories, one can create an  $N \times N$  similarity matrix. Note that distance functions can be converted into similarity functions with kernel transformations (like the Gaussian kernel in Chapter 3) where needed. Subsequently, the kernel trick can be applied as discussed in section 3.3.8 of Chapter 3. The basic idea is to extract a multidimensional embedding from the similarity matrix and then apply the Mahalanobis method (or any other distance-based method) in order to report outliers.

### 11.4.3 Supervised Outliers in Trajectories

In many cases, supervision may be available in the form of labels associated with trajectories. For example, consider a case where the trajectories of a large number of ships are available, and it is desirable to identify the suspicious ones based on their trajectory patterns. In some cases, previous examples of anomalous trajectories may be available. These can be used in order to detect significant anomalous patterns in the underlying data. This is a homogeneous attribute scenario, since the unusual shapes are based purely on the spatial and temporal attributes, rather than on a behavioral attribute.

The ROAM method [355] uses a discrete *symbolic* approximation of the trajectories, which converts the numerical coordinate sequence into a symbolic sequence based on the directions of movement and significant changes in this direction. For example, motifs could correspond to *right-turn*, *u-turn* or *loop*. Every movement pattern can be described as a sequence of these primitive movement patterns. The important motifs can be mined directly from the data by using a clustering approach. If desired, additional meta-attributes may be associated with the symbols corresponding to characteristics of the movement such as the speed. This is however different from the concept of behavioral attributes, since these attributes do not play the behavioral role in the learning process.

Once the discrete representation has been created, the sequences together with their labels can be fed to any sequence-based classifier, which identifies how different sequences are related to the class labels. While the ROAM method was applied in the context of supervised models, it is important to note that the feature transformation used in this work can also be used in the context of unsupervised scenarios.

Although such methods have not been explored extensively, it is possible to use the similarity-based and tile-based transformations of the previous sections. The similarity-based transformation can be used to transform the problem into the multidimensional domain, whereas the tile-based transformations can be used to transform the problem into the sequence mining domain. In both cases, one can leverage the large number of off-the-shelf classification tools in these settings. Furthermore, the similarity-based transformations need not be performed explicitly because one can directly use kernel SVMs in conjunction with the extracted similarity matrix.

## 11.5 Conclusions and Summary

---

The problem of spatial outlier detection arises in many domains such as demographic analysis, disease outbreaks, image analysis, and medical diagnostics. Spatial outlier detection shares significant resemblance with temporal outlier detection in terms of the effects of contextual attributes on the continuity of the behavioral attributes. Therefore, a number of methods in the temporal domain can be used for outlier detection in the spatial domain. Spatiotemporal outlier detection is even more complex and challenging, since it combines spatial and temporal characteristics effectively for outlier analysis. The spatial and temporal attributes can occur either as contextual or behavioral attributes. Depending on which attributes are contextual, the nature of the underlying data set is also quite different.

Spatial data can often be treated as an abstraction of image data, when the contextual attribute natural partitions the space into grid regions, and the values of the behavioral attribute(s) are known over all grid regions. In such cases, numerous methods for image analysis can be used for outlier detection. In fact, in many applications, such as MRI scans and weather maps, the data are *explicitly* represented as images. The analysis of such data involves the determination of unusual shapes from the distribution of the spatial attributes. Such analysis can be performed both in the unsupervised and supervised scenarios.

## 11.6 Bibliographic Survey

---

The problem of finding spatial outliers is different from that in multidimensional data because of the different types of attributes in spatial data. The simplest problem setting is that of discovering point outliers. In such cases, small changes in the behavioral attribute in a particular spatial proximity are used in order to identify outliers [2, 324, 376, 487, 488, 489, 490]. Spatial proximity can be defined either with the use of multidimensional distances or graph-based distances. Spatial distances are more relevant when the contextual attributes are expressed in terms of coordinates. On the other hand, when the contextual attributes correspond to spatial regions or semantic locations, graph-based methods are more relevant, because distances and proximity can be expressed as general functions across links. A random-walk approach to determine free-form spatial scan windows is discussed in [285]. The application of outlier detection to heterogeneous neighborhoods is discussed in [286]. The work in [573] introduces a spatial likelihood ratio test in order to determine local grid regions in which the variation of the behavioral attribute is different from the remaining data in a statistically significant way. Furthermore, such methods can also be used in the context of multiple behavioral attributes [138]. Spatial data also show local heterogeneity because of different levels of variance in different parts of the data. Therefore, a local method for spatial outlier detection was proposed in [523].

The standard autoregressive models for temporal data [467] can be extended to spatial data, in which the behavioral attribute values are completely specified over all the different reference values. This is often the case with many forms of image data. The problem of unusual shape detection in images is an important one from the perspective of outlier analysis. Some recent work [565] has been performed on finding unusual shapes in images in an efficient way. Supervised methods for shape detection and change analysis are also widely available in the literature [69, 115, 375, 602]. The work in [251] uses multivariate Gaussian Markov random fields in order to find unusual shapes in medical image data.

Spatial data is similar to temporal data in the context of the continuity shown by the behavioral attributes. Numerous methods for autoregressive modeling [467] can also be generalized to the case of spatial data. A significant amount of data in spatial domains also have a temporal component, when the attributes are tracked at multiple time-stamps. This requires methods for spatiotemporal outlier detection [141, 142]. An application of spatiotemporal outlier detection to precipitation data is discussed in [569]. A method for detecting flow anomalies in the context of sensors located upstream or downstream from one another is discussed in [304]. When the differences in the values of the sensors exceeds a given threshold, it is flagged as a spatiotemporal anomaly. A streaming method for explicitly quantifying the level of local change in a spatiotemporal data stream, which is discussed in section 9.4.2.1 of Chapter 9, was proposed in [19]. Methods for detecting anomalies in vegetation data with principal component analysis are discussed in [341].

The detection of outliers in trajectories can be modeled either spatially or temporally. Significant *changes* in trajectory directions are useful for many applications, such as hurricane tracking [117]. In such cases, the trajectory can be treated as bivariate temporal data, and change analysis can be applied to this representation. For this purpose, the prediction-based deviation detection techniques of the previous chapter can be helpful. The works in [102, 215] determine anomalies in moving object streams in real time, by examining patterns of evolution. On the other hand, the detection of anomalous trajectory *shapes* is a very different problem. The earliest methods for trajectory shape outlier detection were proposed in [319]. However, this method transforms the trajectories into point data by using a set of features describing meta-information about the trajectories. Unsupervised methods for trajectory outlier detection, which actually use the sequence information explicitly, were first investigated in [347, 409]. The work in [409] uses the discrete Fourier transform in order to represent the trajectories in terms of the leading coefficients, and find anomalies. In the second method [347], trajectories are divided into different line segments and anomalous patterns are identified in order to determine outliers. Supervised methods for anomaly detection in trajectory data may be found in [355]. These methods transform the data into discrete sequences, and a classifier is learned in order to relate the trajectories to the class labels. Another method proposed in [357] proposes methods for finding outliers in vehicle traffic data. However, these methods are not designed for determining outliers on individual objects, but are designed for finding anomalous traffic regions (or road segments) on the basis of aggregate spatial traffic characteristics.

## 11.7 Exercises

---

1. Construct the closed form solution to the AR regression model proposed in this chapter. Use the methods proposed in Chapter 3 for this purpose.
2. Construct PCA models for relating multiple behavioral attributes at the same spatial location and discovering points in the spatial grid that are anomalies. Use analogous



models to those discussed in Chapter 9 for this purpose.

3. Construct PCA models for relating multiple behavioral attribute values over spatially local slices of size  $p \times p$ . Use analogous spatial models to the time-series models proposed in Chapter 9 for this purpose.
4. What is the time complexity of the methods proposed in Exercises 2 and 3.
5. Create a generalization of the time-series shape detection algorithm discussed in section 9.3 of Chapter 9 [311] to the spatial shape detection scenario. Refer to the details in [311] for specific details of pruning based on Symbolic Aggregate Approximation.
6. Implement the algorithm developed in Exercise 5 using a C++ implementation. Test it over benchmark data sets discussed in [565].
7. Implement the algorithm discussed in this chapter for unusual shape detection. Refer to [565] for specific details of LSH-based pruning. Test it over benchmark data sets discussed in [565]. How does the speed compare to the algorithm developed in Exercise 6.

---

---

## Chapter 12

# Outlier Detection in Graphs and Networks

---

---

“In nature, we never see anything isolated, but everything in connection with something else which is before it, beside it, under it and over it.” – Johann Wolfgang von Goethe

### 12.1 Introduction

---

Graphs represent one of the most powerful and general forms of data representation. These structures are used to express diverse data, ranging from multidimensional entity-relation graphs, the Web, social networks, communication networks, and biological and chemical compounds. Broadly speaking, two types of graphs arise in real domains:

- The data may contain many small graphs, drawn over a small base domain of labeled nodes. Some examples of this scenario include chemical and biological compounds. The labels correspond to the chemical elements, which may be repeated within the same object or different objects. The repetition of node labels causes a serious computational challenge in such applications, which is referred to as *graph isomorphism*. Individual graph objects are defined as outliers based on the model of normal graph objects in the database. Therefore, outlier scores are associated with entire graphs as well.
- The data may be represented as a single large graph. Examples include the Web, social networks, and information networks. In most cases, such as the Web and social networks, the nodes correspond to distinct identifiers such as Uniform Resource Locators (URLs), actors, or IP addresses. In some cases, it is also possible for node labels to be repeated. In this setting, outlier scores are defined on the structural elements (e.g., nodes, edges, subgraphs) of the single large graph.

Many other natural variations of the aforementioned scenarios arise, such as the extraction of multiple small graphs from a larger network. An example of such a scenario is a bibliographic network, in which a publication may be represented as a small graph over a larger

bibliographic co-authorship network. In this setting, the smaller graphs may be defined as outliers based on their linkage relationships. As discussed in section 12.3.2.4, the methods for outlier analysis in a single large graph can be easily generalized to this case. These different settings require very different models. For example, in the case of small graphs, a single object may be represented as an outlier. However, in large graphs, the outliers are defined as portions of the network, which might be nodes, edges, or even subgraphs.

In temporal settings, outliers typically correspond to structural changes in large-scale networks like the Web, social or communication networks. Structural changes may be modeled in terms of communities, shortest paths, or other local structural properties. Temporal graphs represent one of the most challenging cases for outlier analysis, because of the many different ways in which outliers may be defined. The following phrase from Chapter 1 is restated here: “*The more complex the data is, the more the analyst has to make prior inferences of what is considered normal for modeling purposes.*” For example, in a temporal network, an outlier node could be a node with unusually high degree, unusual connectivity structure, unusually *changing* degree, unusually *changing* community structure, unusually *changing* distances to other nodes, or unusual relationships of node content to linkage structure. There is virtually an unlimited number of different ways in which outliers could be defined. Even within the context of a specific outlier type such as a node outlier, the appropriate model of regularity could be based on its degree, neighbor set, edge-weight distribution, and so on. Therefore, even the *definition* of an outlier in a very complex data type, such as a graph, might lend itself to a bewildering number of possibilities.

In such scenarios, it is important to define and model the outliers from an application-specific perspective, because no uniform definition exists and a specific application might provide better guidance. For example, in a spam detection application, the degree distributions of nodes can provide insights about outliers. In a network de-noising application, the linkage connectivity structure can be used to determine outlier links. This type of domain-specific knowledge can be considered a mild form of supervision, since it is often created based on prior data-centric experiences. This chapter will focus on a number of specific definitions of network outliers because of their ubiquity in many tasks. Nevertheless, the methods discussed are certainly not exhaustive and there might be numerous settings in which other definitions might be more appropriate.

This chapter will study the following aspects of outlier detection in graphs and networks:

- The problem of outlier detection in many small graphs will be introduced. Such graphs occur commonly in the domain of chemical data, and are therefore useful in determining unusual structural combinations.
- Different models for outlier analysis will be studied in the context of a single large graph. These cases arise commonly in the context of the Web, communication networks, and social networks. However, the complexity of a large network allows the definition of outliers in a variety of interesting ways. This setting is particularly important because of the increasing prevalence of very large graphs in real-world settings.
- The issue of temporal outlier detection in graphs will be studied. This area is intimately connected to the topic of *evolutionary network analysis* [14]. The goal is to study significant local and global structural changes in graphs that are abrupt and unexpected.

Different variations of the aforementioned scenarios will be addressed, such as the incorporation of node content or domain knowledge into the outlier analysis process.

This chapter is organized as follows. Section 12.2 addresses the problem of outlier detection in many small graphs. Section 12.3 discusses the problem of outlier detection on a single large graph. The case of many small graphs super-imposed on a single large graph will also be discussed within this section. Methods for incorporating node content in outlier analysis are discussed in section 12.4. The problem of change analysis and outlier detection in temporal graphs is discussed in section 12.5. The conclusions and summary are presented in section 12.6.

## 12.2 Outlier Detection in Many Small Graphs

---

An interesting case for outlier analysis arises when the data contains many small graphs. Some examples include chemical compounds, biological networks, XML graphs, Resource Description Framework (RDF) graphs, and entity-relation graphs. In some of these domains, the graphs could become quite large, although the graphs are of relatively modest size in most cases. A complicating factor in these domains is that the labels on nodes may typically be repeated, as a result of which the matching can be performed in a variety of different ways. This is referred to as *the challenge of isomorphism*. Therefore, similarity computations can sometimes be difficult. A discussion of these issues is provided in [558].

The problem of outlier detection in such cases is similar to multidimensional point anomaly detection, where each graph object is treated as an individual point. In this context, the easiest class of models to generalize is that of proximity-based models. This is because the problems of clustering and similarity search have been widely studied in domains such as chemical compound mining and XML graph analysis [15, 558]. In the case of clustering methods [15], the clusters are defined as sets of graphs that overlap with the frequent subgraph patterns in the underlying data set. Since outliers are defined in a complementary way to clusters, they are defined as graphs that do not overlap well with the frequent subgraph patterns in the data. Thus, a natural approach for outlier detection in such scenarios is to determine the frequent subgraphs in the underlying data, and determine those graphs which do not contain these frequent patterns. This approach is analogous to determining the outliers in transaction data with the use of pattern mining. Outliers can be defined as graphs which do not contain a significant number of frequent subgraph patterns. Therefore, an approach analogous to that discussed in section 8.5.1 of Chapter 8 for outlier detection in transaction data [254] can also be used for the case of graphs. As in the case of transaction data, support-based quantifications can be used in this case, except that the support is defined on subgraph frequent patterns, rather than transaction frequent patterns.

The use of a  $k$ -nearest neighbor outlier detection algorithm is also a viable alternative in this case, because of a wide variety of available algorithms for similarity search in graphs [558]. Numerous similarity functions are commonly used such as the graph edit distance, largest common subgraph, largest matching node set, to perform the similarity search. A detailed discussion of commonly used similarity and distance functions in the graph domain may be found in [33] (Chapter 17). While these methods correspond to generic extensions of multidimensional outlier detection algorithms, not much work has been specifically targeted towards outlier detection in this domain.

### 12.2.1 Leveraging Graph Kernels

The design of effective similarity functions is helpful not only for distance-based outlier detectors but also for using other linear models such as PCA or one-class support-vector

machines. For this purpose, we need a special type of similarity function, which is also referred to as a *graph kernel*. Two well-known kernels referred to as the random-walk kernel and the shortest-path kernel are discussed in [33]. For a database containing  $N$  graphs, one can construct an  $N \times N$  kernel similarity matrix  $S$ . This similarity matrix can be used in two different ways to detect outliers:

1. One can use the one-class SVM of section 3.4 in Chapter 3 to create a model of normal graphs. This model can be used to compute the outlier scores of the graphs.
2. One can use soft kernel PCA, which is discussed in section 3.3.8 of Chapter 3. The Mahalanobis method can be used in conjunction with the resulting embedding in order to score the data points. The extension of the Mahalanobis method to arbitrary data types is discussed in section 3.3.8.3.

Graph kernel methods can also be used for supervised outlier detection with the use of kernel support-vector machines.

## 12.3 Outlier Detection in a Single Large Graph

---

In large graphs, unusual structural characteristics can be used to define outliers in different regions of the network. Such outliers can be defined in a variety of ways, such as node outliers, linkage outliers, or subgraph outliers. Each of these different types of outliers will be discussed in detail. Throughout this section, it will be assumed that a single large network or graph is available for analysis. This graph is denoted by  $G = (N, A)$ . Here,  $N$  denotes the set of nodes, and  $A$  denotes the adjacency matrix of the set of edges. It is assumed that the number of vertices in  $N$  is  $n$ , and the number of edges in  $A$  is  $m$ . Therefore,  $A$  is an  $n \times n$  matrix, which contains  $m < n^2$  nonzero entries. In most settings, the matrix  $G$  is sparse, and therefore we have  $m \ll n^2$ . Unless otherwise mentioned, it will be assumed that the graph  $G$  is undirected; however, some of the following methods can also be generalized to directed graphs.

### 12.3.1 Node Outliers

Node outliers can be defined in a network in a wide variety of ways. The key is to extract features from the neighborhood of a node, and then define the outliers in terms of these features. The work in [41] defines a set of features that are extracted from the 1-step neighborhood (also known as *egonet*) of node  $i$ . The features are as follows:

- (Node Feature  $n_i$ ) The node feature is the number of nodes in the 1-step neighborhood of node  $i$  (which is the same as its degree).
- (Edge Feature  $e_i$ ) The edge feature is the number of edges between all nodes in the 1-step neighborhood of node  $i$ .
- (Weight Feature  $w_i$ ) For weighted graphs, the weight feature comprises the total weight of all edges in the 1-step neighborhood of node  $i$ .
- (Spectral Feature  $\lambda_i$ ) The spectral feature is the principal eigenvalue of the weighted subgraph in the 1-step neighborhood of node  $i$ .

These features can then be organized into carefully chosen pairs, which are shown to obey a number of interesting power laws. Data points that deviate from these power laws are flagged as outliers. Some examples of useful pairs of features for anomaly detection are as follows:

- **Node feature versus edge feature:** Extremely dense neighborhoods of a node correspond to near-cliques, whereas extremely sparse neighborhoods of a node are stars. By plotting the relationship between the node and edge features, it is possible to detect near cliques and stars. In general, the *expected relationship* between these features is  $e_i \propto n_i^\alpha$ , where  $\alpha \in (1, 2)$ . Therefore, for a given graph, one can learn the best fit for  $\alpha$ , and then report the deviations from this model as the outlier scores.
- **Weight feature versus edge feature:** If edges are received unusually frequently in the neighborhood of a node, this will result in a high value of the weight feature relative to the edge feature. This corresponds to the *heavy vicinity* of a node. The expected relationship between these features is  $w_i \propto e_i^\beta$ , where  $\beta \geq 1$ , but usually ranged in  $(1, 1.29)$  according to the experiments in [41].
- **Spectral Feature versus Weight Feature:** This approach detects a single dominating heavy edge in the neighborhood of a node. The expected relationship between these features is  $\lambda_i \propto w_i^\gamma$ , where  $\gamma \in (0.5, 1)$ .

It should be noted that many features can be mined from the neighborhood of a graph, each of which provides a different notion of an outlier. The aforementioned possibilities simply reflect three *instantiations* of features extracted from a graph neighborhood, so that deviations from power-law models can be used to define outliers. Power laws implicitly encode *domain knowledge* about the behavior of *typical* networks. As discussed frequently in the book, the use of domain knowledge is often very useful for meaningful outlier analysis.

On the other hand, not all features extracted may satisfy such power laws. In general, a wide variety of features can be extracted from the neighborhood of nodes. This can be used to define a multi-dimensional feature space, in which the features extracted from each node correspond to a multi-dimensional data point. For example, the shape of the degree distribution of the neighbors of a node provides a different characterization of the outlier behavior. A general meta-algorithm for node outlier analysis in networks is as follows:

1. Extract features  $f_1 \dots f_r$  from the  $k$ -hop neighborhood of a node. Create a new multi-dimensional data point  $(f_1 \dots f_r)$  specific to that node.
2. Apply any point anomaly detection algorithm to the extracted multi-dimensional data set, and report unusual data points as outliers.

Virtually an unlimited number of different features could be extracted from a node. The precise features extracted should depend on the application at hand. For example, in a social network, it could be a vector containing the number of messages exchanged with each neighboring node, and in a spam detection application, it could correspond to the degree of the node. Therefore, the key in effective node outlier detection is to extract these features in a way that is sensitive to the particular application at hand. In many cases, this process requires a deeper semantic understanding of the significance of the extracted features in that particular graph domain.

### 12.3.1.1 Leveraging the Mahalanobis Method

The feature extraction methods of the previous section are very effective in cases in which domain knowledge is available about *relevant* characteristics for outlier detection. However, when such domain knowledge is not available, one can use off-the-shelf methods like the Mahalanobis method of section 3.3.8.3. The basic idea is to use the following steps:

1. For a graph  $G$  with  $n$  nodes create an  $n \times n$  similarity matrix  $S$ . The adjacency matrix is itself a kind of similarity matrix. However, one can make the similarity matrix more robust by using a number of methods such as the SimRank between pairs of nodes, the Katz measure, the Jaccard coefficient between their neighbor sets, or the personalized PageRank values between pairs of nodes. Note that many of these techniques are used for the link prediction problem in graphs, and are discussed in [33] in detail.
2. Apply the Mahalanobis method of section 3.3.8.3 on similarity matrix  $S$  in order to discover anomalous nodes in the graph. Although the Mahalanobis method requires positive semi-definite similarity matrices, section 3.3.8.3 also discusses methods to adjust similarity matrices that are not positive semi-definite.

One can view this approach as a variant of matrix factorization methods, which will be discussed later in this chapter.

## 12.3.2 Linkage Outliers

Linkage outliers are defined as unexpected edges in particular regions in the network. In most cases, linkage outliers are edges that lie across dense partitions in the network. However, there are numerous models that one can use to define such unexpected edges. The main issue here is to ensure robustness and low computational complexity. The most promising class of algorithms in this category is that of matrix factorization, which is robust and can also be made to be computationally efficient in most settings. The matrix factorization methods can be used to identify both linkage outliers and node outliers. In fact, the Mahalanobis method of the previous section is a special case of this class of techniques in which the factorized basis vectors are mutually orthogonal.

### 12.3.2.1 Matrix Factorization Methods

Matrix factorization provides a natural way to determine anomalies in data which is represented in the form of a 2-dimensional matrix. Consider a graph  $G = (N, A)$  with node set  $N$  and an adjacency matrix  $A$ . Assume that the number of nodes in  $N$  is denoted by  $n$ . The adjacency matrix is therefore of size  $n \times n$ . It is possible for the entries of the adjacency matrix to either be binary or reflect weights on the edges. In most practical settings, the matrix  $A$  is very sparse and most of the entries take on zero values. Therefore, the problem of finding linkage anomalies in such graphs can be stated as the problem of determining anomalous entries in a matrix of nonnegative values.

**Problem 12.3.1** *Given an  $n \times n$  adjacency matrix  $A$ , determine anomalous entries in the matrix.*

Matrix factorization provides a natural technique for finding linkage anomalies in such networks. Section 3.5 of Chapter 3 discussed how one can find anomalous entries in a multidimensional data set with the use of matrix factorization. After all, a multidimensional data set can also be represented as a matrix. This general principle applies to *any type of matrix*,

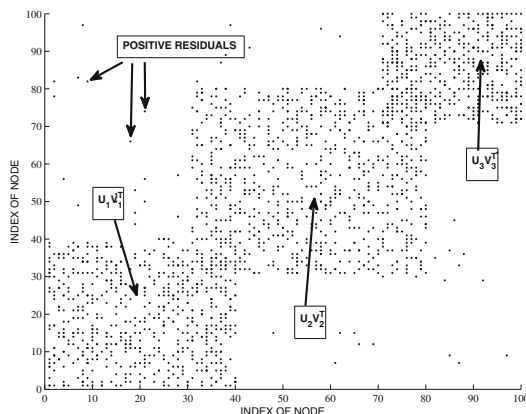


Figure 12.1: Sum-of-parts interpretability of NMF. The matrix has a natural clustering structure each of which is captured by a dominant rank-1 component  $U_i V_i^T$ . Outliers are not captured by any component and therefore remain as (positive) residuals. Missing entries within a block are negative residuals and are useful for link prediction.

such as an adjacency matrix, and not just matrices constructed from multidimensional data. In fact, in the case of adjacency matrices, the sparsity and nonnegativity can be leveraged to discover more interpretable decompositions in an efficient way. Just as Equation 3.28 of Chapter 3 factorizes a data matrix  $D$ , we can decompose the  $n \times n$  adjacency matrix  $A$  as follows:

$$A \approx UV^T \quad (12.1)$$

Here,  $U$  and  $V$  are  $n \times k$  matrices, where  $k$  is the rank of the factorization. The diagonal entries of  $A$  are set<sup>1</sup> to the node degrees. In the weighted case, the node degrees are defined as the sum of the entries in each row. One can construct an objective function that minimizes the Frobenius norm of  $A - UV^T$  as follows:

$$\begin{aligned} &\text{Minimize } \|A - UV^T\|^2 \\ &\text{subject to:} \\ &U \geq 0, \quad V \geq 0 \end{aligned}$$

Although the nonnegativity constraints are not essential, they have some interpretability advantages. Any off-the-shelf optimization solver can be used for this problem, although we will discuss a more efficient approach later in section 12.3.2.1.

This approach can be viewed as a low-rank factorization of  $A$ , in which the *residual matrix* ( $A - UV^T$ ) represents entries that do not conform to the low-rank assumption. Therefore, the low-rank assumption defines the model of normal data. At smaller values of  $k$ , the absolute values of the entries in the residual matrix will be larger. Therefore, the

<sup>1</sup>This ensures that the matrix is positive semi-definite and it helps in certain forms of *symmetric* factorization of the form  $A = UU^T$ , which requires positive semi-definite (PSD) matrices. This is because the adjacency matrix  $A$  can now be expressed  $A = \sqrt{W}\sqrt{W}^T$ , where  $W$  is the  $n \times m$  *node-edge* incidence matrix. Such a matrix is always PSD. Strictly speaking, the PSD property is not necessary for the type of asymmetric factorization discussed above.



outlier scores are defined by the following residual matrix  $R$ :

$$R = A - UV^T \quad (12.2)$$

Clearly, large *absolute* values of the residual correspond to linkage anomalies. Any form of extreme-value analysis may be used on the matrix  $R$  to determine the linkage anomalies. Positive values of the residuals correspond to existing edges that should not be present in the graph. Negative values of the residual correspond to edges that should be present, but are not present in the graph. Therefore, they represent the anomaly of *missing edges*. Such methods are, therefore, used for *link prediction* in networks [34], but the absence of such links can also be considered anomalous when the absolute value of the residual is large enough.

### Interpretability of NMF Decomposition

Although one can use any form of low-rank factorization, NMF methods have the advantage of being interpretable, and are intimately connected with clustering. Incidentally, clustering is one of the other techniques discussed later in this section for detecting linkage anomalies. Let  $U_i$  and  $V_i$  be the  $i$ th columns of the matrices  $U$  and  $V$ , respectively. Then, the aforementioned factorization can be expressed in the following additive form:

$$A \approx UV^T = \sum_{i=1}^k U_i V_i^T \quad (12.3)$$

Each  $U_i V_i^T$  is also nonnegative (because of nonnegativity of  $U$  and  $V$ ), and the positive edges in  $U_i V_i^T$  typically represent the edges in a particular community (together with other predicted edges in that community). Note that  $U_i V_i^T$  is a rank-1 matrix with a complete block structure. Therefore, the overall decomposition can be viewed as an additive sum of the contributions from  $k$  different communities (blocks) in the graph.

In order to understand this point, consider a graph with 100 nodes and three natural clusters, some of which are overlapping. Assume for convenience that the nodes are indexed in order of their clustering structure. In other words, nodes 1 through 40 form one community, the nodes 30 through 80 form another (overlapping) community, and the nodes 70 through 100 form the third community. The nonzero entries in such a  $100 \times 100$  matrix are shown in Figure 12.1, and they form a natural block structure. Note that some scattered entries in this matrix structure are the residuals, and these entries correspond to the outlier edges. Consider a setting in which we use rank-3 NMF on resulting matrix. Each  $U_i V_i^T$  will yield one of the blocks (clusters) in the aforementioned matrix. The scattered entries (outliers) will not be captured by any of the blocks, and will therefore show up as large (positive) residuals.

A different type of matrix factorization is one in which the residuals are constrained to be non-negative for entries in the matrix containing edges [551]. This type of factorization provides different types of insights into the outliers.

### Efficiency and Practical Issues for Matrix Factorization

There are several efficiency and practical issues for performing nonnegative matrix factorization. In order to avoid overfitting, it is useful to leverage regularization. Regularization methods are discussed in section 3.5.1 of Chapter 3, in which we add penalties  $\alpha(\|U\|^2 + \|V\|^2)$  to the objective function. A second issue is that the underlying matrix may be very large.

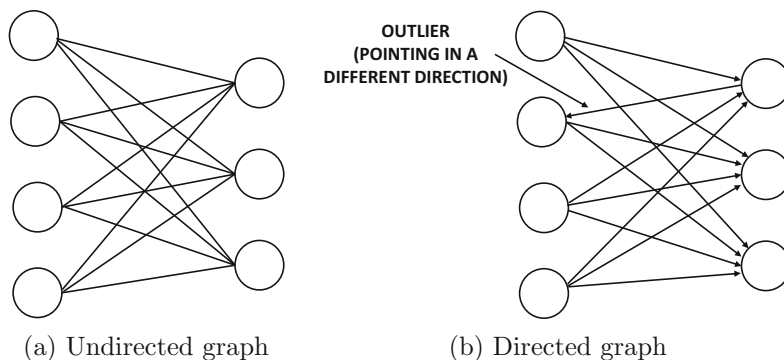


Figure 12.2: Finding outliers in directed graphs. The direction of the edge matters and not just the underlying community structure. The matrix factorization family is general enough to address directed graphs.

For example, for a very modest network containing  $10^5$  nodes, the number of entries in the matrix is  $10^{10}$ , which creates challenges. Fortunately, such matrices are sparse, and therefore, one can often perform the factorization more efficiently. One interesting approach is to use the sampling of the zero entries. The basic idea is to use all the nonzero entries and to sample the zero entries to perform the factorization using the incomplete matrix factorization approach of section 3.5.1. Some changes need to be made to the gradient-descent steps to accommodate the nonnegativity of factors. To achieve this goal, any negative entries of matrices  $U$  and  $V$  must always be set to 0 after each gradient-descent step. Furthermore, one must always initialize the entries in  $U$  and  $V$  to nonnegative values. The resulting factorization can be used to create an outlier score of each non-diagonal entry in the adjacency matrix. This process can be repeated multiple times, and the scores can be averaged to provide a final prediction.

In addition to estimating the outlier scores of links, one can also score nodes. The outlier score of node  $i$  is obtained by averaging the squared (edge) scores in both row  $i$  and column  $i$  in  $R$ . The scores of all non-diagonal entries in the  $i$ th row and column are used, whether or not they correspond to links that are actually present in the graph.

### Application to Directed Graphs

One of the nice aspects of the matrix factorization methodology is that it can be easily applied to directed graphs, and it is not restricted to undirected graphs. For example, the undirected graph in Figure 12.2(a) is a complete bipartite graph, and it has no outlier edge. However, the graph in Figure 12.2(b) is derived from Figure 12.2(a) by adding directions to the edges. In this case, there is a single outlier edge in Figure 12.2(b), which is caused primarily by its direction. Such outliers can also be identified by using the aforementioned matrix factorization methodology. The only difference is that an asymmetric adjacency matrix is factorized in this case, and the matrices  $U$  and  $V$  will be interpreted as *sender* and *receiver* factors. The steps for identifying outliers are otherwise identical to the undirected case.

### 12.3.2.2 Spectral Methods and Embeddings

The matrix factorization methodology discussed above can be (conceptually) replicated with the use of spectral methods, at least for the case of undirected graphs. In general, matrix factorization methods are known to be roughly equivalent to spectral methods. Note that we can factorize the adjacency matrix  $A$  directly into orthogonal factors using eigen-decomposition. This is (almost) equivalent to using (unnormalized) spectral methods. Note that  $A$  is positive semi-definite with nonnegative eigenvalues when the diagonal entries of  $A$  are set to be equal to the node degrees. Therefore, one can factorize  $A$  as follows:

$$A \approx P\Lambda P^T = (P\sqrt{\Lambda})(P\sqrt{\Lambda})^T \quad (12.4)$$

Here,  $P$  is an  $n \times k$  matrix, whose columns contain the orthonormal eigenvectors (with largest eigenvalues), and  $\Lambda$  is a diagonal  $k \times k$  matrix containing the nonnegative eigenvalues. One can also view this approach as a type of symmetric matrix factorization  $A \approx UV^T$ , in which both  $U$  and  $V$  are set to  $P\sqrt{\Lambda}$ . Furthermore, instead of nonnegativity, the columns of  $U$  and  $V$  satisfy the constraint that they are mutually orthogonal. Therefore, this spectral approach also falls squarely into the category of matrix factorization methods. As before, the residual matrix may be computed as follows:

$$R = A - UV^T = A - (P\sqrt{\Lambda})(P\sqrt{\Lambda})^T \quad (12.5)$$

Nondiagonal entries with large (positive) residual values may be declared anomalies. These methods are intuitively very similar to the matrix factorization methods discussed above, except that PCA is used for determining the residuals. Of course, the actual values of the residuals will be somewhat different, since the models are not mathematically identical. The interpretability of such methods is lower, since it is no longer possible to impose non-negativity constraints on the factors. The matrix  $(P\sqrt{\Lambda})$  is an embedding of the nodes in multidimensional space. However, this type of symmetric factorization has another advantage in terms of using the embedding directly in conjunction with multidimensional *node* outlier detection methods. In fact, the Mahalanobis method of section 12.3.1.1 is an example of such an approach.

### Leveraging the Embedding

Even though matrix factorization methods are designed for discovering edge anomalies, the resulting embeddings can also be leveraged for discovering node anomalies. Note that  $P\sqrt{\Lambda}$  provides a  $k$ -dimensional embedding of the nodes. In order to discover the outliers, we set  $k$  to the smallest value, such that all<sup>2</sup> nonzero eigenvectors are captured in  $P$ . The matrix  $P$  provides a normalized embedding, in which each of the  $k$  dimensions has unit variance. Subsequently, the Euclidean distance of each row to the mean of the rows of  $P$  provides the outlier score of that node. This approach is equivalent to the use of the kernelized PCA version of the Mahalanobis method (see Chapter 3). Furthermore, one can use many other multidimensional outlier detection techniques on this representation.

Note that one can use other kernel methods such as diffusion kernels, or use other types of node-node similarity matrices (such as the Katz, SimRank, or personalized PageRank measures), to perform the factorization and generate the embedding. These different types of embeddings will work well in different application domains. In many cases, it is useful to

---

<sup>2</sup>One must be careful not to be misled by nonzero eigenvalues caused by numerical errors. Such eigenvalues are usually obvious because of their extremely small values. As a practical matter, one can set a very small threshold, such as  $10^{-8}$ .

have some insight about a relevant and application-specific notion of similarity to generate an effective embedding.

### 12.3.2.3 Clustering Methods

Nonnegative matrix factorization is only one of the many ways in which one might cluster a network to determine outlier edges. In general, if the nodes in the graph are clustered, then the edges across these node clusters are defined as outliers. Unfortunately, there is no single way of defining the clusters in a network. In many cases, the clustering may be imperfect, and may merge two sets of nodes that should ideally be in different clusters. The source of this problem is that graphs often contain overlapping clusters, and a strict partitioning may be defined in many ways by a randomized algorithm. In such cases, different clusterings might result in different edges being reported as outliers. One approach for increasing robustness is to use randomized ensembles.

#### Randomized Ensembles

The method in [17] defines a clustering that is based on randomized sampling of the edges in the network. This is a special case of subsampling-based ensembles for outlier detection (cf. section 6.4.6.2 of Chapter 6), although the sampling is performed on the *entries* (edges) of the adjacency matrix rather than its *rows*. A sample of edges from a stream implicitly creates a set of clusters in terms of the connected components in this sample. Such connected components are much denser than randomly selected node sets in the graph, because of the inherent bias of edge sampling. Since the connected components in edge samples are known to create *weak clusters* of densely connected nodes, such an approach is an excellent strategy for link outlier estimation, when used within the context of an ensemble framework. In other words, this randomized sampling is repeated in order to create the clusters in different ways. For each clustering, a likelihood fit is defined for an edge. The median of these likelihood fits is defined as the final outlier score. Furthermore, the approach in [17] modifies the edge sampling in order to maintain specific properties of node partitions such as balancing the number of nodes in each cluster (see section 12.5.2).

Consider a partitioning of the nodes denoted by  $\mathcal{C} = C_1 \dots C_k$ . The number of node partitions in  $\mathcal{C}$  is denoted by  $k$ . Each set  $C_i$  represents a disjoint subset of the nodes in  $V$ . The likelihood fit is defined with respect to a *structural generation model* of edges with respect to node partitioning  $\mathcal{C}$ . This defines the probability of an edge between a pair of partitions and it also provides an outlier score for the edge. Lower values indicate that a data point is an outlier.

**Definition 12.3.1 (Edge Generation Model [17])** *The structural generation model of a node partitioning  $\mathcal{C} = \{C_1 \dots C_k\}$  is defined as a set of  $k^2$  probabilities  $p_{ij}(\mathcal{C})$ , such that  $p_{ij}(\mathcal{C})$  is the probability of a randomly chosen edge in the network to be incident on partitions  $i$  and  $j$ .*

The probabilities  $p_{ij}(\mathcal{C})$  are estimated from the fraction of edges between various pairs of clusters. The partition identifier for node  $i$  is denoted by  $I(i, \mathcal{C})$ , and it takes on a value between 1 and  $k$ .

**Definition 12.3.2 (Edge Likelihood Fit)** *Consider an edge  $(i, j)$ , a node partition  $\mathcal{C}$ , and edge generation probabilities  $p_{ij}(\mathcal{C})$  with respect to the partition. Then, the likelihood*

fit of the edge  $(i, j)$  with respect to the partition  $\mathcal{C}$  is denoted by  $\mathcal{F}(i, j, \mathcal{C})$  and is given by  $PI(i, \mathcal{C}), I(j, \mathcal{C})$ .

The likelihood fit of an edge can be defined with respect to any partition in the network. It remains to be explained how the likelihood fits from different partitions can be combined.

The  $r$  different ways of creating the partitions are denoted by  $\mathcal{C}_1 \dots \mathcal{C}_r$ . Specifically, the  $i$ th clustering  $\mathcal{C}_i$  contains  $k(\mathcal{C}_i)$  different clusters. The composite edge-likelihood fit from these  $r$  different ways of creating the partitions is defined as the *median* of the edge likelihood fits over the different partitions.

**Definition 12.3.3 (Edge Likelihood Fit (Composite))** *The composite edge likelihood fit over the different clusterings  $\mathcal{C}_1 \dots \mathcal{C}_r$  for the edge  $(i, j)$  is the median of the values of  $\mathcal{F}(i, j, \mathcal{C}_1) \dots \mathcal{F}(i, j, \mathcal{C}_r)$ . This value is denoted by  $\mathcal{MF}(i, j, \mathcal{C}_1 \dots \mathcal{C}_r)$ .*

A variety of randomized clustering methods can be used to create the different partitions. For example, any off-the-shelf randomized algorithm can be used for partitioning purposes. However, the edge sampling technique has the virtue of being efficient to implement, and it is also helpful in performing effective variance reduction. A method has also been proposed [17] to maintain the clusterings for a *stream of edges* with an online reservoir sampling approach. Thus, this approach can be used to determine edge anomalies in graph streams. The reservoir sampling approach will be discussed in detail in section 12.5.2.

### 12.3.2.4 Community Linkage Outliers

In some network applications, many small graphs may be superimposed on a single large graph. For example, in a bibliographic network, a small graph object may represent a publication. In such cases, it is desirable to identify objects, for which the linkage structure is anomalous compared to the linkage structure of other objects in the network. This case is often not very different from determining linkage outliers. The main difference is that the likelihood scores for the different edges in the object need to be combined into a single likelihood score [17].

The likelihood fit for a graph object  $G$  is the product of the likelihood fits of the edges in  $G$ . In order to fairly compare between graphs which contain different numbers of edges, we put the fraction  $1/|G|$  in the exponent, where  $|G|$  is the number of edges in the incoming graph stream object. In other words, we use the geometric mean of the likelihood fits of different edges in the incoming graph stream object. Therefore, we define the object likelihood fit as follows.

**Definition 12.3.4 (Graph Object Likelihood Fit)** *The likelihood fit  $\mathcal{GF}(G, \mathcal{C}_1 \dots \mathcal{C}_r)$  for a graph object  $G$  with respect to the partitions  $\mathcal{C}_1 \dots \mathcal{C}_r$  is the geometric mean of the (composite) likelihood fits of the edges in  $G$ .*

$$\mathcal{GF}(G, \mathcal{C}_1 \dots \mathcal{C}_r) = \left[ \prod_{(i,j) \in G} \mathcal{MF}(i, j, \mathcal{C}_1 \dots \mathcal{C}_r) \right]^{1/|G|} \quad (12.6)$$

Objects that have extremely low likelihood fits are reported as community-linkage outliers [17]. In many cases, any particular linkage in the set may not have a low fit, but the overall fit, when evaluated over the entire set, may be very low. Thus, such outliers are examples of *collective* outliers, whereas node and linkage outliers are *contextual* outliers.

Strictly speaking, this measure can be defined for any set of nodes in a single large network, rather than a specified set of nodes in the small graph superimposed over a larger network. In such cases, an additional challenge is to *discover* the appropriate sets of nodes that are connected together in an anomalous way. This is a much more difficult problem with the use of only network structure information, unless additional content-based supervision is available. A closely related notion of community outliers was proposed in [214], which finds such anomalous communities in a single large network. However, node content is used to provide additional supervision for the discovery of such anomalous sets of nodes.

### 12.3.3 Subgraph Outliers

A subgraph outlier is defined as a part of the graph which exhibits unusual behavior with respect to the normal patterns in the full graph. Subgraph outliers cannot be defined meaningfully in a large graph, unless there are repetitions in the labels on the nodes. For example, in a Web or social network graph, in which all node identifiers are distinct, it is much harder to define a notion of regularity. However, if the nodes are associated with labels drawn from a relatively small subset of possibilities, it is much easier to define subgraph outliers by finding subgraphs which relate these labels to one another in unusual ways. Alternatively, the nodes may not have labels associated with them, and the matching is based purely on structural similarity. Either of these scenarios can be addressed by the work in [416], which designs information-theoretic models for subgraph outlier detection.

The approach proposed in [416] is based on the Minimum Description Length (MDL) principle for outlier detection. The broader principles underlying the approach are based on the SUBDUE system for subgraph pattern mining. In the subdue system, a commonly occurring pattern is one which provides a small description length for the purposes of representation. For example, if  $S$  is a substructure which occurs repeatedly in a graph  $G$ , then by compressing that substructure into a single vertex, the graph is compressed (corresponding to a smaller description length), and the overall graph can be represented in terms of the description of the compressed graph and the smaller substructure. Therefore, the frequent nature of a substructure  $S$  in graph  $G$  may be leveraged to describe the compressed representation of the graph concisely in terms of the description length  $DL(G|S)$  of the compressed graph and that of the repeatedly occurring substructure  $S$ . The conciseness is achieved, because the repeatedly occurring substructure needs to be described only once. This description length  $F1(S, G)$  is defined as follows:

$$F1(S, G) = DL(G|S) + DL(S) \quad (12.7)$$

Subgraphs that are very common (and therefore not outliers) will have low values of  $F1(S, G)$ . One possible solution is to identify those subgraphs that have *high values* of  $F1(S, G)$  and flag them as outliers. Such an approach does not seem to work very well for anomaly detection because it does not discriminate between the varying sizes of subgraphs. For example, subgraphs with only one node will often have high values of the outlier score according to this criterion. Therefore, a different heuristic criterion was proposed in [416], which is based on the spirit of the original definition for pattern frequency, but normalizes for subgraph size as well. Specifically, this definition is as follows:

$$F2(S, G) = Size(S) * Instances(S, G) \quad (12.8)$$

Here,  $Size(S)$  represents the number of nodes in the subgraph  $S$ , and  $Instances(S, G)$  represents the number of instances of subgraph  $S$  in the graph  $G$ . Subgraphs with low value of  $F2(S, G)$  are reported as outliers.

A second approach is based more directly on the SUBDUE method [151]. The SUBDUE method is designed to determine the common substructures in the graph in an iterative way. Specifically, this method compresses the common substructures in the graphs, and replaces them with new nodes. Substructures that are more common are compressed in earlier iterations. The key insight in detecting anomalous subgraphs is in determining whether significant portions of the subgraph were compressed in early iterations. If this is *not* the case, it implies that the subgraph contains few common substructures. Such a subgraph should be considered anomalous. Therefore, the outlier score  $O$  of a subgraph  $S$  is defined in terms of the iteration index  $i$  of the SUBDUE method as follows:

$$O = 1 - \sum_{i=1}^n \frac{(n-i+1)}{n} \cdot \frac{DL_{i-1}(S) - DL_i(S)}{DL_0(S)} \quad (12.9)$$

The outlier score always lies in the range  $(0,1)$ . High values indicate that the object is anomalous. The description length  $DL_i(S)$  of a subgraph typically reduces with increasing iteration index  $i$ , because of the compression of nodes inside it. Note that the term  $\frac{DL_{i-1}(S) - DL_i(S)}{DL_0(S)}$  corresponds to the fraction of compression in the  $i$ th iteration, and the term  $\frac{(n-i+1)}{n}$  provides greater weight to a compression occurring in an earlier iteration. This is because more common graphs are compressed in earlier iterations. Therefore, the value of the outlier score will be impacted by high quickly the nodes of the subgraph  $S$  become compressed because of the membership of common patterns in them. This provides a heuristic definition of the outlier score, based on the MDL principle.

## 12.4 Node Content in Outlier Analysis

---

The incorporation of node content can have significant benefits in outlier analysis, particularly in terms of discovery of outlier links between nodes. In most application domains such as the Web, social networks, and information networks, linkages between nodes are highly correlated to content similarity between nodes. For example, Web pages with similar content are more likely to be linked. In social networks, network participants are more likely to be linked to other participants with similar interests. On the other hand, nodes with certain types of content are very unlikely to link with one another. For example, an official US Government Web site is unlikely to link to Web sites encouraging criminal endeavors for the most part. Discovering outlier linkages in such settings can have numerous applications:

- Noisy outlier linkages can degrade the accuracy of network mining algorithms. Therefore, it may be useful to detect and remove such linkages before performing the mining. This problem is referred to as *network denoising* [212, 452].
- Anomalous linkages can provide insights about unusual connections among nodes. This may have applications beyond noise removal, since the formation of anomalous connections is indicative of interesting facts about the underlying system.

These methods often use consistency modeling between the content and clustering structure to identify linkage anomalies.

### 12.4.1 Shared Matrix Factorization

The matrix factorization methods discussed in the previous section can also be extended to include content. The key idea is to use *shared* matrix factorization. Aside from the  $n \times n$

adjacency matrix  $A$ , we also have an  $n \times d$  content matrix  $C$ , where  $d$  is the size of the underlying lexicon. Then, the idea is to simultaneously factorize  $A$  and  $C$  as follows:

$$A \approx UV^T \quad (12.10)$$

$$C \approx UW^T \quad (12.11)$$

Here,  $U$  and  $V$  are  $n \times k$  matrices, whereas  $W$  is a  $d \times k$  matrix. Note that the matrix  $U$  is *shared* in both factorizations. Then, one can determine the appropriate factors by using the following optimization formulation:

$$\begin{aligned} & \text{Minimize } \|A - UV^T\|^2 + \beta\|C - UW^T\|^2 + \alpha(\|U\|^2 + \|V\|^2 + \|W\|^2) \\ & \text{subject to:} \\ & U, V, W \geq 0 \end{aligned}$$

Here,  $\beta > 0$  is the balancing parameter and  $\alpha > 0$  is the regularization parameter. One can initialize the matrices with nonnegative entries and use gradient-descent methods to solve this problem. Let  $E_A = (A - UV^T)$  and  $E_C = (C - UW^T)$  be the error matrices at any particular step of the descent. Then, the gradient descent steps are as follows (with step-size  $\eta > 0$ ):

$$\begin{aligned} U & \leftarrow U(1 - \eta\alpha) + \eta E_A V + \eta\beta E_C W \\ V & \leftarrow V(1 - \eta\alpha) + \eta E_A^T U \\ W & \leftarrow W(1 - \eta\alpha) + \eta\beta E_C^T U \end{aligned}$$

At each step, the entries in  $U$ ,  $V$ , and  $W$  that become negative are set to 0 in order to respect the nonnegativity constraint. The nonnegativity constraint provides better interpretability, but it is not essential to the basic formulation. Interested readers should refer<sup>3</sup> to [34] for more details on the derivation.

After the optimal solutions for  $U$ ,  $V$ , and  $W$  have been determined, the anomalous edges in  $A$  can be determined from the large positive residuals of  $(A - UV^T)$ . Similarly, the anomalous words in  $C$  can be determined from the large positive residuals in  $(C - UW^T)$ . Large negative residuals correspond to missing edges or missing words. Anomalous nodes can also be determined by summing up the squares of the residuals of the relevant residuals in  $A$  and  $C$ . For example, for the  $i$ th node, we can sum up the squares of the residuals of the  $i$ th rows in  $A$  and  $C$  to evaluate the outlier score of node  $i$ .

### 12.4.2 Relating Feature Similarity to Tie Strength

Another recent method [212] models the similarity in the feature vectors at the nodes to an intuitive notion of tie-strength across links. This tie-strength is modeled as a weight vector, and represents the consistency of links with their neighborhood feature vector. Each node  $i$  has a  $k$ -dimensional (column) feature vector  $\bar{f}_i$  associated with it. Let  $F_i$  be a  $k \times d_i$  matrix of content weights (for the neighborhood of node  $i$ ) with rows representing the  $k$  features and columns representing its  $d_i$  neighbors. Let  $\bar{w}_i$  be a non-negative column vector of length  $d_i$  containing the tie-strength of node  $i$  to each of its  $d_i$  neighbors. Then,  $F_i \bar{w}_i$  represents the set of predicted features at node  $i$  based on the structure and content of the

<sup>3</sup>A solution to a very similar optimization formulation is provided in section 11.3.8 of Chapter 11 of [34]. However, this solution is provided in the completely different context of recommender systems with trust matrices.



neighborhood of node  $i$ . The error of this prediction is given by  $\|F_i \bar{w}_i - \bar{f}_i\|_2^2$ . In addition, an  $L_1$ -regularization penalty is imposed on the weight vector to encourage sparsity. The following objective function is minimized:

$$\text{Minimize}_{\bar{w}_i \geq 0} \sum_i (\|F_i \bar{w}_i - \bar{f}_i\|_2^2 + \lambda \cdot \|\bar{w}_i\|_1) \quad (12.12)$$

Here,  $\lambda > 0$  is a regularization parameter that regulates the tradeoff between the sparsity of the weight vector and the learned accuracy. Links that correspond to very small values of the weights (or zero weights) are assumed to be outliers. Larger values of  $\lambda$  will result in sparser weight vectors, and, therefore, fewer links will be included. This is a convex non-linear programming problem for which many optimization solvers are available [112].

### 12.4.3 Heterogeneous Markov Random Fields

A recent method [452] uses a heterogeneous Markov random field framework (HMRF) to evaluate the consistency of links with the clustering structure of the network. Therefore, this model uses a binary variable  $n(i, j)$  to quantify whether or not a link should be considered anomalous. Similarly, a node-specific variable indicates its affinity to each cluster. Then, the two variables are *simultaneously* learned to identify the clusters and outlier links in the network. The criteria for learning these variables leverage the consistency between network clustering structure and node content. The key idea is to create clusters and infer the anomalousness of links simultaneously in a *mutually consistent way*. At the same time, the linkages that are inconsistent with the learned variables are deemed to be anomalous. A detailed introduction of HMRF models is beyond the scope of this book. Interested readers are referred to [452] for details.

Finally, the *community-outlier detection* method [214] identifies outlier communities by using a combination of content-based and structural analysis. Linkage outliers try to find a *pair* of unusually connected nodes in the network. Community-outliers can be considered unusual subgraphs in the network and are therefore a generalization of linkage outliers. The technique in [214] proposes a method for efficient identification of community outliers with a heterogeneous Markov random field (HMRF) model. Interested readers are referred to [214] for details.

## 12.5 Change-Based Outliers in Temporal Graphs

---

Many entities such as the Web, social, and information networks are temporal in nature. This has led to increasing interest in the field of *evolutionary network analysis* [14]. In such cases, unusual and abrupt changes in the structure of the network should be detected and reported as anomalies. As discussed in earlier chapters, outlier detection and change analysis are tightly integrated problems [193, 579], especially when the changes are abrupt and reflect unusual events. In the context of graphs and networks, one can characterize these changes in various ways:

- **Node anomalies:** Sudden changes in the structural patterns in the vicinity of a node can lead to node anomalies. Such anomalies are also referred to as *hotspots* [597]. For example, if a researcher suddenly starts publishing with a new set of co-authors in a bibliographic network, it will lead to an anomaly.

- **Linkage anomalies:** New edges that are inconsistent with the current graph structure may be considered anomalous. For example, the arrival of a link that bridges two sparsely connected regions of the network may be considered anomalous. This is similar to the static case, except that only the past history of the stream is relevant for modeling.
- **Community evolution:** In this case, significant and unusual changes in the community structure are tracked [20, 233, 519] and reported as anomalies.
- **Distance evolution:** In this case, pairs of nodes with unusually large changes in the distances are detected [234] and reported as anomalies.
- **Latent embedding methods:** These are very general methods that discover temporal embeddings from the sequence of graph snapshots. A temporal smoothness constraint is incorporated in the embedding. Violations of this constraint represent change points.

In certain types of networks such as communication networks, the edges may be received as a fast stream. The streaming scenario poses special challenges because it is important to be able to build the model of normal data and report outliers in an online setting.

### 12.5.1 Discovering Node Hotspots in Graph Streams

Unusual activity in a network might cause unusual changes in the neighboring network structure. These unusual changes are also referred to as *hotspots*. In order to discover such hotspots, the work in [597] performs *localized* eigenvector analysis. The idea is to create a matrix characterizing the immediate locality of a particular node. This matrix is created as a decay-based version of the adjacency matrix of the immediate locality of the node. Specifically, if  $A(i)$  is the adjacency matrix directly adjacent to the node  $i$ , then the matrix  $M(i) = A(i)^T A(i)$  is used to characterize the neighborhood of node  $i$ . This matrix is maintained in conjunction with a decay-function in online fashion. It has been shown in [597] how such a matrix can be maintained efficiently with the use of lazy updates.

In order to determine anomalies, the eigenvectors of  $M(i)$  are determined. Note that the eigenvectors of  $M(i)$  are the same as the right singular vectors of  $A(i)$ . Therefore, this approach is essentially a matrix factorization method. The magnitude and direction of the dominant eigenvector of this matrix will change over time as the local structure near the node changes over time. Furthermore, abrupt changes in these eigenvectors can also be helpful in discovering anomalous changes. There are two primary types of anomalous changes:

1. If the magnitude of the eigenvector changes abruptly, it means that there is a significant change in the level of activity near that node. For example, in a bibliographic network, a sudden increase in publication volume of an individual will increase the magnitude of the eigenvector.
2. If the angle of the eigenvector changes abruptly, it means that the pattern of linkages in the locality of the node has changed abruptly. In a bibliographic network, a sudden change in the pattern of co-authors will lead to a change in the direction of the eigenvector.

The level of change in both cases can be reported continuously as an outlier score. One can use a threshold on this outlier score in order to trigger alarms about significant events.

It is noteworthy that PCA is a form of matrix factorization. It is possible to use this approach in conjunction with other forms of matrix factorization. Other forms of factorization such as NMF might provide better interpretability in many settings.

## 12.5.2 Streaming Detection of Linkage Anomalies

The linkage anomaly detection method in section 12.3.2 can be extended to graph streams [17]. The only difference is that the likelihood estimation for a linkage anomaly needs to be computed with respect to the previous history of the graph stream. In this case, cluster-based partitions need to be maintained dynamically from the edge stream<sup>4</sup> to perform the linkage anomaly detection. It is well-known that the use of edge sampling can be used to create dense partitions. The main problem is that specific *structural properties* of the  $k$ -way cut partitions need to be maintained. For example, one might want to ensure a minimum number of points in each cluster, or to constrain the total number of clusters. Clearly, a random edge sample may not satisfy such constraints. This goal is achieved with the help of a *monotonic set function* of the underlying edges in the reservoir. A monotonic set function is defined on the sample as follows.

**Definition 12.5.1 (Monotonic Set Function)** *A monotonically non-decreasing (non-increasing) set function is a function  $f(\cdot)$  whose argument is a set, and value is a real number which always satisfies the following property:*

- If  $S_1$  is a superset (subset) of  $S_2$ , then  $f(S_1) \geq f(S_2)$

Some examples of monotonic set functions are as follows:

- The function value is the number of connected components in the edge set  $S$  (monotonically non-increasing).
- The function value is the number of nodes in the largest connected component in edge set  $S$  (monotonically non-decreasing).

In order to maintain the desired structural properties on the induced clustering (e.g., cluster balance), it is possible to use *thresholds* on the above properties, which are also referred to as *stochastic stopping criteria*. Let  $\mathcal{D}$  be a set of edges. Consider a restricted bunch of subsets of  $\mathcal{D}$  in which the edges are sorted and can be added to  $S$  only in *sort order priority*; in other words, an edge cannot be included in a subset  $S$ , if all elements occurring before it in the sort order are also included. Clearly, the number of such subsets of  $\mathcal{D}$  is linear in the size of  $\mathcal{D}$ .

**Definition 12.5.2 (Sort Sample with Stopping Criterion)** *Let  $\mathcal{D}$  be a set of edges. Let  $f(\cdot)$  be a monotonically non-decreasing (non-increasing) set function defined on the edges. A sort sample  $S$  from  $\mathcal{D}$  with stopping threshold  $\alpha$  is defined as follows:*

- All edges in  $\mathcal{D}$  are sorted in random order.
- The smallest subset  $S$  from  $\mathcal{D}$  among all subsets satisfying the sort-order priority is identified, such that  $f(S)$  is at least (at most)  $\alpha$ .

---

<sup>4</sup>The work in [17] uses a stream of composite objects, although this presentation simplifies it to edge streams. Either scenario can be handled by the approach with small modifications.

This means that if the last added element is removed, then that set (and all previous subsets) will not satisfy the stopping criterion. As a practical matter, the set obtained by removing the last added element is the most useful for processing purposes. For example, if  $f(S)$  is the size of the largest connected component, the stopping criterion determines the smallest sample  $S$  in which the size of the largest connected component is at least a user-defined threshold  $\alpha$ . By dropping the last edge  $(v, w)$  that was added to  $S$ , it is guaranteed that the size of the largest connected component in  $S - \{(v, w)\}$  is less than  $\alpha$ . Correspondingly, a *penultimate set* for a sort sample is defined as follows.

**Definition 12.5.3 (Penultimate Set)** *The penultimate set for a sort sample  $S$  is obtained by removing the last element in the sort order of sample  $S$ .*

For the case of a *fixed data set*, it is fairly easy to create a sort sample with a structural stopping criterion. This is achieved by sorting the edges in random order and adding them sequentially, until the stopping criterion can no longer be satisfied. However, in the case of a data stream, a random sample or reservoir needs to be maintained *dynamically*. Once edges have been dropped from the sample, it becomes a challenge to compare their sort order to incoming edges.

The key idea is use a *fixed random hash function*, which is computed as a function of the node labels on the edge, and remains fixed over the entire stream. This hash function is used to create a sort order among the different edges. This hash function serves to provide *landmarks* for incoming edges when they are compared to the previously received edges from the data stream. Furthermore, the use of a hash function *fixes the sort order among the edges throughout the stream computation*. The fixing of the sort order is critical in being able to design an effective structural sampling algorithm. Therefore, for an edge  $(i, j)$  the hash function  $h(i \oplus j)$  is computed, where  $i \oplus j$  is the concatenation of the node labels  $i$  and  $j$ . The use of a sort on a random hash function value induces a random sort order on the stream elements. Furthermore, a stopping criterion on the sorted data set *translates* to a threshold on the hash function value. This provides an effective way to control the sampling process. In other words, it is desirable to determine the smallest threshold  $q$ , such that the set  $S$  of edges that have hash function value at most  $q$  satisfy the condition that  $f(S)$  is at least (at most)  $\alpha$ . The key observation here is that the value of the threshold  $q$  varies over the life of the data set, as more edges arrive, and a smaller *fraction* of the edges in the stream need to be included in the reservoir to satisfy the structural constraint. Intuitively, this “smaller” fraction translates to lower hash thresholds. In fact, the following result has been explicitly shown in [17]:

**Theorem 12.5.1 (Hash Threshold Monotonicity)** *The stopping hash threshold is monotonically non-increasing over the life of the data stream.*

This result implies that edges that have not been included in the current sample will *never* be relevant for sampling over the future life of the data stream. Thus, there is no risk that any stream edge which was discarded will become useful later. The current sample is the only set needed for any future decisions about reservoir sample maintenance. The key here is to find ways to dynamically update the hash threshold and the stream sample continuously so as to maintain the structural constraints.

A simple algorithm is used to dynamically maintain the reservoir. The *current hash threshold* is maintained dynamically to make decisions on whether or not incoming elements are included in the reservoir. For each incoming edge, the hash function is applied, and it is added to the reservoir, if the hash function value is less than the current threshold value. The addition of an edge will always result in the stopping criterion being met because of

set function monotonicity. However, the set may no longer be the *smallest sort-ordered set* to do so. Therefore, edges may need to be removed in order to make it the smallest sort-ordered set to satisfy the stopping criterion. In order to achieve this goal, the edges in the reservoir are processed *in decreasing order of the hash function value* and removed, until the resulting reservoir is the smallest possible set which satisfies the stopping constraint. The corresponding hash threshold is then reduced to the largest hash function value of the *remaining edges in the reservoir* after removal. Clearly, the removal of edges may result in a reduction of the hash threshold in each iteration. However, it will never result in an increase in the threshold, because all the added edges had a hash function value lower than the threshold in the previous iteration. The penultimate set derived from the sort sample is always used for the purposes of maintaining the cluster partition as the set of underlying connected components.

The above description explains the maintenance of a single reservoir (and corresponding partition). For robustness, a set of  $r$  different reservoirs is maintained, which corresponds to  $r$  different partitionings of the data. Therefore,  $r$  different hash functions are used to create the different reservoir samples. These are used to define the  $r$  different node partitionings required by the outlier modeling algorithm. The *penultimate sets* of the  $r$  different reservoirs are denoted by  $S_1 \dots S_r$ . These will be used for the purpose of inducing the  $r$  different partitionings denoted by  $\mathcal{C}_1 \dots \mathcal{C}_r$ . Correspondingly, the outlier score of an edge can be computed from the likelihood fits, as discussed in section 12.3.2 of this chapter.

### 12.5.3 Outliers Based on Community Evolution

Evolutionary changes in a dynamic network often cause significant changes in the structure of the underlying communities. The analysis of community evolution reveals important insights about sudden changes in membership of nodes, the formation or disappearance of clusters, the erratic membership of nodes in clusters, and a general change in the overall clustering quality. Many of these algorithms integrate clustering maintenance with evolution analysis, but this is not always the case [14].

#### 12.5.3.1 Integrating Clustering Maintenance with Evolution Analysis

While evolutionary clustering [119, 143] is widely studied in the community detection literature, a recent method [233] also integrates the clustering process with evolution analysis. The integration of an evolution analysis procedure into the clustering process is critical in determining relevant change points in the data. The work in [233] uses a probabilistic clustering model to learn the clusters from the underlying data.

The *ENetClus* method [233] generalizes the probabilistic *NetClus* [525] model to the temporal scenario. This is a soft clustering model, which assigns probabilities of membership of each node to different clusters. The idea in this model is to perform the clustering on temporal snapshots of the data. On each snapshot, a probabilistic assignment is learned with the use of the *NetClus* algorithm. The final probabilistic assignment in a given snapshot is used as an initialization point (prior) for the next iteration. This ensures that continuity is maintained among the clusters, and the clusters found in the next snapshot can be directly compared to their counterpart in the current snapshot. A number of evolution metrics are then proposed in order to measure significant clustering properties in the form of a time-series stream. Significant deviations in these values (by using the methods discussed in Chapter 9) can be reported as anomalous changes in the network. A subset of the more important temporal change quantifications introduced in [233] are presented below:

- **Cluster membership consistency:** The vector of probabilities of membership of a node to different clusters is available in soft clustering algorithms like *ENetClus*. The cosine similarity between the vectors in successive snapshots can be reported as the change value.
- **Cluster snapshot quality:** The ratio of intra-cluster similarity to inter-cluster similarity is used as a quantification of the quality of the clusters. Significant changes in the clustering quality between successive snapshots is indicative of the fact that the inherent clustering tendency of the network has changed significantly over time. For example, if a network receives a significant number of spam links in a short period of time (an anomalous event), this could abruptly disrupt the clustering tendency of the data set. This will show up as a sudden change in the time series representing a quantification of the clustering quality.
- **Cluster novelties, merges, splits, and disappearances:** The formation of new clusters represents a novelty in the data. Similarly, significant structural changes may occur, corresponding to cluster merge, split or disappearance. Each of these different structural anomalies is quantified in [233].
- **Temporal object stability:** Objects that move across different clusters over time are inherently unstable and should therefore be considered outliers. The fraction of time-stamps at which the membership of the cluster remains the same between successive snapshots is a definition of temporal object stability. The inverse of this quantity is the outlier score for the object.
- **Temporal object sociability:** Objects that inherently belong to many different clusters are considered sociable. In the context of a soft clustering algorithm, this means that the membership probability vector for the object is distributed across different clusters. This definition of sociability is different from the degree of a node, since it is performed in the context of the community behavior, which provides more intuitive insights into aggregate sociability trends. This can be quantified by using the Gini index or the entropy for the object in terms of cluster membership probabilities. Let  $p_1 \dots p_k$  be the membership probabilities for an object in the  $k$  different clusters. The Gini index is defined by the expression  $1 - \sum_{i=1}^k p_i^2$ , and the entropy is defined by the expression  $-\sum_{i=1}^k p_i \cdot \log(p_i)/k$ .

Larger values of each of these quantifications indicate greater sociability. For example, in a bibliographic network, this corresponds to authors who collaborate across many different research areas. Sudden changes in the sociability can provide an understanding of the significant changes in the collaboration patterns of a particular author.

In general, the use of any of these metrics is dependent on the specific type of application at hand.

One can also use other clustering methods, such as spectral methods, for the aforementioned analysis. However, this results in a hard clustering and the aforementioned quantifications would need to be designed differently. A method for incorporating temporal smoothness in spectral clustering algorithms is discussed in [143]. While this method is not designed explicitly for change detection, it can be used for change detection, since an approximate mapping can be found between clusters at different time snapshots. This is because of the incorporation of the temporal smoothness criterion, which allows a clear mapping

between clusters at different snapshots. A specific application of this kind of approach to the monitoring of evolution in blog communities is discussed in [415].

### 12.5.3.2 Online Analysis of Community Evolution in Graph Streams

The work in [20] discovers the use of highly expanding or highly contracting communities with the use of the concept of a *differential graph*. The differential graph over interval  $(t_1, t_2)$  is defined as the fractional rate at which the level of interaction along an edge has changed in that period. Let  $w_{ij}(t)$  be the weight of edge  $(i, j)$  at time  $t$  based on its frequency of arrival. In order to generate the differential graph, the *normalized graphs* at times  $t_1$  and  $t_2$  are constructed. The normalized graph  $G(t) = (N(t), A(t))$  at time  $t$  is denoted by  $\overline{G(t)}$ , and contains exactly the same node and edge set, but with different weights. Let  $W(t) = \sum_{(i,j) \in A} w_{ij}(t)$  be the sum of the weights over all edges in the graph  $G(t)$ . Then, the normalized weight  $\overline{w_{ij}(t)}$  is defined as  $w_{ij}(t)/W(t)$ . The normalized graph therefore contains the fraction of interactions of each edge. The differential graph is constructed from the normalized graph by using a subtractive process on the normalized graphs at snapshots  $t_1$  and  $t_2$ . Therefore, the differential graph  $\Delta G(t_1, t_2)$  contains the same nodes and edges as  $\overline{G(t_2)}$ , except that the differential weight  $\Delta w_{ij}(t_1, t_2)$  on the edge  $(i, j)$  is defined as  $\Delta w_{ij}(t_1, t_2) = \overline{w_{ij}(t_2)} - \overline{w_{ij}(t_1)}$ .

In the event that an edge  $(i, j)$  does not exist in the graph  $\overline{G(t_1)}$ , the value of  $\overline{w_{ij}(t_1)}$  is assumed to be 0. Because of the normalization process, the differential weights on many of the edges may be negative. These correspond to edges over which the interaction has reduced significantly during the evolution process. For instance, in a bibliographic network, when the *rate of authoring new publications* between a pair of authors reduces over time, the corresponding weights in the differential graph are also negative.

After the differential graph has been constructed, it is desired to determine highly evolving node subgraphs. Edges with highly positive or negative weights correspond to evolving entities. Therefore, in order to identify expanding and contracting communities, it is desired to identify subgraphs in which most interactions are all either highly positive or highly negative. This is a much more difficult problem than that of finding clusters within the subgraph  $\Delta G(t_1, t_2)$ . Methods for finding such subgraphs are proposed in [20].

### 12.5.3.3 GraphScope

*GraphScope* [519] is a method to detect significant changes in bipartite (directed) graphs. Such graphs are useful for modeling a wide variety of directional interactions, such as users with Web sites, clients with hosts, and so on. Furthermore, since these interactions are dynamic, the patterns in the interactions will change over time.

The work in [519] uses a combination of dynamic community detection and information-theoretic methods to identify the key change points. In particular, the Minimum Description Length (MDL) principle is leveraged to identify change points. Intuitively, a change point significantly increases the encoding cost (MDL cost) to represent the stream. Therefore, by computing the change in the MDL cost, one can detect key change points.

Given a set of temporal snapshots  $G_1 \dots G_t$ , the idea is to group them into *contiguous segments*, which are separated by change points (denoted by ‘\*’ below):

$$\underbrace{\{G_1, G_2, G_3\}}_{\text{Segment 1}} * \underbrace{\{G_4, G_5\}}_{\text{Segment 2}} * \underbrace{\{G_6, G_7, G_8, G_9\}}_{\text{Segment 3}} * \dots$$

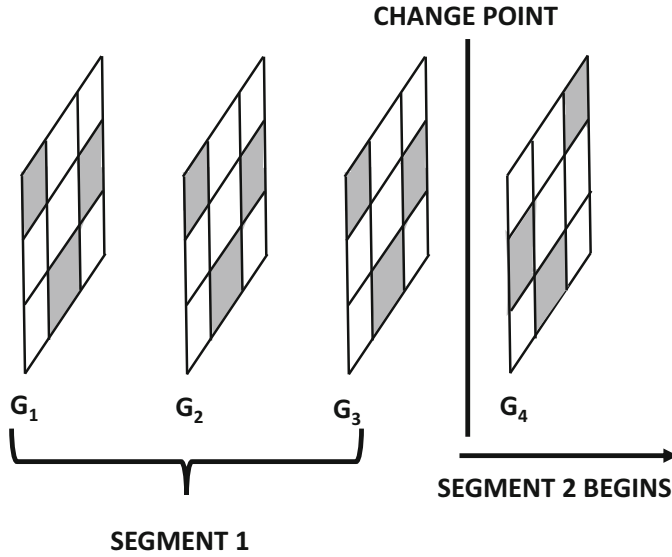


Figure 12.3: Illustrating change points in *GraphScope*

Each segment is compactly stored (i.e., encoded) in terms of the best fitting community structure of that segment, as well as the offsets (i.e., errors) of the individual graphs in that segment with respect to this best fit. The idea here is that graphs  $G_1 \dots G_3$  are very similar in terms of community structure (and can therefore be summarized in a small number of bits), but adding  $G_4$  to this segment will greatly increase the encoding cost. This situation is illustrated in Figure 12.3. The main problem in *GraphScope* is to identify the best segmentation of the time-series of graphs, and also determine the best community structure within each segment. Note that one can find a common community structure in each segment only if the communities have not changed very significantly in that segment.

The approach groups similar sources together into source groups, and similar destinations together into destination groups. One can view this as a block-wise co-clustering, shared by all the snapshots of a segment, which is similar to the block-wise clustering in Figure 12.1. The main difference from Figure 12.1 is that there are no node overlaps among clusters. If the underlying communities do not change much over time, then the snapshot of the evolving graphs will have similar descriptions and can also be grouped together into a time segment, to achieve better compression. A binary Huffman encoding of the community structure within a segment is used in order to represent it. In principle, one could use any other type of matrix compression, such as a shared matrix factorization of all graphs in the segment, as long as the size and error of the compressed description can be properly encoded into a cost. Whenever a new graph snapshot cannot fit well into the old segment in terms of this description, *GraphScope* introduces a change point and starts a new segment. These change points are identified by comparing the cost of including the snapshot in the current segment versus starting a new segment. The lower cost option is always selected. It has been shown in [519], that such change points correspond to drastic discontinuities in the network, which can be regarded as temporal outliers. Readers are referred to [519] for details.



### 12.5.4 Outliers Based on Shortest Path Distance Changes

Most real-world graphs such as the Web, social networks and information networks experience significant changes in terms of the pairwise distances between nodes in the network. For example, it has been shown in [67] that most real graphs such as the Web and social networks have shrinking diameters over time. This is because edges are continuously added to such networks, which makes them more dense.

In this context, *sudden and abrupt* changes in pairwise distances between nodes are indicative of unusual events in a network. For example, in a bibliographic network such as *DBLP* [625], most pairs of nodes within a specific topical area can typically be connected by small paths of lengths 2 or 3. On the other hand, the sudden addition of an edge that connects a pair of nodes at a distance of 5 is an unusual event. Most likely, this event reflects the sudden collaboration between a pair of authors in different topical areas. Such changes can, therefore, provide useful insights.

A straightforward solution to this problem is to solve the all-pairs shortest-path problem [40] at two snapshots  $t_1$  and  $t_2$ . The pairs of nodes for which the distances have changed very significantly are reported as the anomalies. However, such an algorithm requires the computation and storage of all-pairs shortest paths, which can be impractical in large-scale settings, such as social, communication, information and Web networks. For example, in a network containing  $10^8$  nodes, the number of possible pairs is  $10^{16}$ . The computation and storage of such a large number of node pairs becomes cumbersome and impractical.

Therefore, it is important to design methods that can *efficiently* identify the top- $k$  distance changes in a heuristic way. A key observation [234] is that edges on shortest paths between many pairs of nodes in either snapshot are important edges. The addition or deletion of such edges can significantly change the shortest path distances. Therefore, a randomized algorithm is proposed in [234] to identify such edges. This is then leveraged in order to identify significant node pairs between which the greatest change has occurred. Although the approach is heuristic, high precision and recall are achieved.

### 12.5.5 Matrix Factorization and Latent Embedding Methods

Many methods in the literature discover latent embeddings over sequences of graph snapshots, while incorporating temporal smoothness constraints [143, 598]. Violations of temporal smoothness constraints represent change points. Unlike the hotspot approach with PCA (cf. section 12.5.1), this approach does require the snapshot-based adjacency matrices  $A_1 \dots A_T$  and is therefore not designed for a fully streaming scenario. The total number of snapshots is denoted by  $T$ . In the following, we discuss the general principle of this method without focusing on any particular technique. Therefore, the description below is more general and quite different from that discussed in [143, 598].

The basic idea is to assume that each adjacency matrix  $A_t$  at time stamp  $t$  can be factorized into two matrices  $U_t$  and  $V_t$ :

$$A_t \approx U_t V_t^T \quad \forall t \in \{1 \dots T\} \quad (12.13)$$

Furthermore, we incorporate the temporal smoothness constraints to ensure that the latent structure does not change suddenly over successive snapshots:

$$\begin{aligned} U_t &\approx U_{t+1} \quad \forall t \in \{1 \dots T-1\} \\ V_t &\approx V_{t+1} \quad \forall t \in \{1 \dots T-1\} \end{aligned}$$

All these constraints can be incorporated into a single objective function with balancing parameters  $\alpha$  and  $\beta$  as follows:

$$\text{Minimize } J = \underbrace{\sum_{t=1}^T \|A_t - U_t V_t^T\|^2}_{\text{Embedding Objective}} + \alpha \underbrace{\sum_{t=1}^{T-1} \|U_t - U_{t+1}\|^2}_{\text{Temporal Smoothness Regularization}} + \beta \sum_{t=1}^{T-1} \|V_t - V_{t+1}\|^2$$

Time-stamps at which the last two terms are large represent change points. Furthermore, by examining the contributions of individual nodes to the last two terms, one can also identify the locality of the changes. Furthermore, the residuals in  $A_t - U_t V_t^T$  provide temporally inconsistent edges at time  $t$ . Therefore, this framework is very general in providing insights about different types of change points. This approach also has the merit of being applicable to directed graphs.

One can change the aforementioned objective function in various ways. For example, instead of penalizing  $U_t$  for violating temporal smoothness, one might fix each  $U_t$  to the same value. However,  $V_t$  is allowed to vary. This results in a shared matrix factorization. Adding regularizers corresponding to the Frobenius norms of  $U_t$  and  $V_t$  reduces overfitting. In other words, the term  $\lambda \cdot \sum_{t=1}^T (\|U_t\|^2 + \|V_t\|^2)$  can be added to the aforementioned objective function to reduce overfitting. Different types of regularizers would result in different properties of the embedding. One can also incorporate nonnegativity constraints to make the solution more interpretable. Nonnegative matrix factorization methods yield latent representations that are similar to temporal variants of Probabilistic Latent Semantic Analysis (PLSA). The main difference is that the objective function of non-negative matrix factorization uses the Frobenius norm, whereas PLSA uses maximum likelihood maximization of a generative model. As discussed in Chapter 8, such factorizations have the merit of probabilistic interpretability. Adding orthogonality constraints on the columns of  $U_t$  and  $V_t$  leads to embeddings similar to PCA and SVD. Such a factorization often has better geometric interpretability and out-of-sample generalizability because of the ability to easily project a point into the different orthogonal directions. These different options provide useful insights in different scenarios. In general, one should choose a specific type of factorization after carefully examining the needs of the application at hand.

It is also possible to combine the approach with the content-centric shared matrix factorization method of section 12.4.1 in order to address cases in which attributes are associated with the nodes at various snapshots. In such cases, one can model co-evolving network structure and attributes. Many social and Web networks have rich attributes associated with the nodes in the network. In such cases, the attributes can greatly help in modeling the network evolution process. In general, latent embedding methods provide the widest choices in enabling different types of application-centric scenarios by changing the corresponding objective function.

## 12.6 Conclusions and Summary

---

The problem of network outlier detection is particularly important because of the ubiquity of different kinds of graphs and networks in a wide variety of problem domains. Graphs can either occur as multiple entities of small size, or as a single large graph. Furthermore, since the nature of graphs is complex, the outliers can be defined in a wide variety of ways, depending upon how regularity is defined. Outliers can be defined in terms of nodes, edges,

subgraphs, evolving edges, evolving subgraphs, or evolving distances. Therefore, it is critical to use application-specific properties to define network outliers in a meaningful way.

Most of the network outlier detection methods examine community changes in the underlying network in one form or the other. This can be done directly using community detection methods, or indirectly through edge sample and spectral methods. Many of these methods have also been generalized to the temporal context. Furthermore, it is relatively easy to generalize such methods to the content-centric scenario with the use of matrix factorization.

## 12.7 Bibliographic Survey

---

Several surveys have recently been written in the context of graph anomaly detection. A general survey on graph-based anomaly detection and description is provided in [43]. Surveys have also been written on anomaly detection in the temporal context. A recent survey on evolutionary network analysis is provided in [14]. Anomaly detection in dynamic networks is discussed in [457]. Detailed surveys on outlier detection in temporal data may be found in [231, 232].

Outlier detection can be defined in the context of many small graphs, or in the context of a single large graph. The former case occurs frequently in scenarios such as chemical and XML data, in which unusual objects need to be determined [15, 558]. Two common methods can be used. Distance-based methods can be easily generalized to this case [558], by defining appropriate similarity functions between the graphs. Alternatively, clustering or frequent pattern mining methods [15] can be used to identify significant anomalies.

Outliers in graphs can be defined in the form of node outliers, linkage outliers, or subgraph outliers. The wide variety of ways in which outliers can be defined in networks is a direct result of the complexity of the data. Even in the context of particular kinds of outliers such as node outliers [41], it has been shown that a wide variety of definitions are possible, depending upon how the features are defined within the locality of a node. A network embedding approach to finding anomalies is discussed in [277]. Linkage outliers are defined as edges that lie across dense clusters in networks [17]. Sketch-based methods for anomaly detection in streaming heterogeneous graphs and edge streams are discussed in [386, 458]. The basic idea is to create a vector representation from the local substructures of a graph for fast similarity computation. The local substructures are themselves represented as short strings.

A method for finding linkage outliers with the use of the Minimum Description Length (MDL) principle is proposed in [120]. The MDL principle is also useful for finding subgraph outliers [416]. Other work along a similar direction was presented in [179]. The use of eigenvector analysis to compare subgraphs with background behavior of the full graph and correspondingly declare them as anomalous was proposed in [397]. The use of such an approach for threat detection in social networks for applications such as counter-terrorism is discussed in [398].

In many practical scenarios, content is available at the nodes. Some examples of such scenarios include social networks, information networks, and the Web. Such content can be used to significantly enhance and sharpen the outlier analysis process. The work in [212] uses the similarity between the features at a node and its neighbors to determine linkage outliers. The work in [452] determines outlier links simultaneously with the underlying communities, with the use of a heterogeneous Markov random field (HMRF) approach. Finally, the work in [214] determines community outliers, which are sets of linked nodes

that are mutually inconsistent with the underlying content. The reverse problem of determining attribute outliers by examining the hierarchical structure of data such as XML has been addressed in [320]. This is because the hierarchical structure of XML data provides semantically meaningful neighborhoods in which outliers may be found. The *FocusCO* algorithm [433] discovers clusters and outliers simultaneously in attributed graphs with the help of user supervision. Scalable methods for anomaly detection in attributed networks are discussed in [434].

Significant research has also been focused on leveraging user supervision for detecting outliers in networks. User supervision is particularly important in the network domain because of the inherent complexity of network data. The aforementioned *FocusCO* is an algorithm that leverages user supervision [433]. Another approach for user supervision is to allow the use of *query templates*, in which the users can determine outlier subgraphs with specific properties [237, 238].

An important problem is that of temporal outlier detection in evolutionary graphs [14]. Outliers can be defined in an almost unlimited number of ways in temporal graphs, because of the different combinations of time and structure, which can be used in order to define regularity. Some of the earliest work focuses on measuring similarities between successive snapshots of graphs with the use of different similarity functions [438]. Another method that uses graph matching between successive snapshots for anomaly detection is discussed in [492]. This creates a time-series which can be analyzed with standard auto-regressive moving average (ARMA) methods for finding the outliers. The work in [280] uses spectral methods to determine anomalies in time-series of graphs. The principal component is chosen as the activity vector for that graph. This graph is then represented as a time series of activity vectors, which creates a data set of activity vector values. The principal left singular vector of this data set provides the significant direction of correlation. The activity vector for a new graph is computed, and the corresponding angle with the principal left singular vector provides the outlier score. A method to detect node anomalies in graph streams with eigenvector analysis is discussed in [597].

The work in [522] uses compact matrix decomposition to approximate the adjacency matrix of large sparse graphs. The primary idea underlying the work is that it is harder to approximate anomalous graphs than normal graphs. Therefore, the approximation error for each graph in a sequence of graphs is constructed. Anomaly detection is performed on this time-series of values. The use of tensor analysis to discover comet communities is proposed in [61]. Other dimensionality reduction methods that are used for anomaly detection in graphs include the use of non-negative matrix factorization [551].

The use of community evolution methods is very common, since communities capture the broad patterns in the network. Therefore, a change in the community structure is used to model significant evolution [20, 143, 233, 235, 236, 383, 519, 520, 521]. The work in [449] uses the history of the node's neighborhood to detect anomalies. Other common methods used are shortest-path distance change metrics [234], and latent embedding methods [143, 598]. Some methods [519, 521] are specifically applicable to bipartite graphs. The determination of significant evolution in graphs can be useful in the context of a wide variety of applications such as monitoring blog communities [415], or mining traffic flow data sets [400]. In the latter case, values are associated with edges, corresponding to traffic flows. Anomalous regions are found in the network, by using the values on these edges. Incremental anomaly detection with the use of *commute time* is discussed in [314]. The basic idea is that incoming nodes with large commute times to existing nodes in the network are reported as anomalies. The commute time is a measure of the distance between two nodes, and it is equal to the expected number of hops required by a random walk to travel from one node to another

and back. Therefore, the approach is a proximity-based outlier detection method, and it is related to incremental community maintenance methods. The main innovation in this work is the ability to compute the commute times in incremental fashion in an efficient way.

Finally, many forms of pattern changes in a network may be characterized in the form of evolution rules. In the framework presented in [81], nodes, and edges are associated with labels (i.e., specific properties of the network). Furthermore, the time-stamps of the first appearance of edges are maintained. Patterns are defined as subgraphs that have similar structure and labels on nodes at different time stamps, and the same relative offsets of the time-stamps. This defines significant temporal patterns or *graph evolution rules* in the underlying data. Evolution rules do not necessarily represent outliers, since they correspond to frequent temporal patterns in the data. On the other hand, the formation of a new evolution rule at a given time may be considered a temporal novelty and may be reported as an outlier.

In many evolutionary graphs, a significant amount of content may be available. Some examples of such data include social streams such as those created by *Twitter*. Therefore, anomaly detection in such scenarios may need to combine both linkage and content. A number of recent methods have also shown how to determine evolutionary outliers in social media streams [30, 435, 530]. Some methods focus on linkage [530], whereas others focus on content [435], and others on a combination of the two [30]. The detection of anomalous meetings between a group of people in a social network is addressed in [495] with the use of an expectation-maximization approach.

## 12.8 Exercises

---

1. Download the *DBLP* bibliographic network [625]. Construct the co-authorship network, where the weights on the edges correspond to the number of publications. Extract the features  $n_i$ ,  $e_i$ ,  $w_i$  and  $\lambda_i$ , as discussed in the node outlier section of this chapter. Create pairwise plots between:
  - $n_i$  and  $e_i$
  - $w_i$  and  $e_i$
  - $\lambda_i$  and  $w_i$

Determine the points which deviate significantly from the least squares fit for each pair. Which nodes are determined as the outliers?

2. Use an edge sampling approach in order to create  $k$  connected components in the network, where  $k = 100000$ . Repeat the process 100 times. Which edges have end points that repeatedly lie in different partitions?
3. Construct a *DBLP* co-authorship network for each year since 1990. Construct a normalized dot-product between the edge sets in successive years. Which are the significant change points along the different years?
4. Repeat Exercise 3 by performing PCA on the un-weighted augmented adjacency matrix graph snapshot obtained from each year of the *DBLP* data set. PCA is used to obtain each year's multidimensional representation on which the dot product is performed.

5. Associate a weight with each edge of the *DBLP* network, corresponding to the number of publications between that author pair. Repeat Exercise 4 using these weighted edges.

---

---

## Chapter 13

# Applications of Outlier Analysis

---

---

“The study and knowledge of the universe would somehow be lame and defective, were no practical results to follow.” – Marcus Tullius Cicero

### 13.1 Introduction

---

Outlier analysis has numerous applications in a wide variety of domains, such as the financial industry, quality control, fault diagnosis, intrusion detection, Web analytics, and medical diagnosis. The applications of outlier analysis are so diverse that it is impossible to exhaustively cover all possibilities in a single chapter. Therefore, the goal of this chapter is to cover many problem domains at a higher level and show how they map to the various techniques discussed in earlier chapters. The practical issues and challenges in the context of real data sets will also be discussed. This will provide a broader understanding of the issues involved in *problem domain to technique mapping*. The main application domains covered in this chapter are as follows:

- Quality control applications
- Financial applications
- Web log analytics
- Intrusion detection applications
- Medical applications
- Text and social media applications
- Earth science applications

In addition, a section will also be devoted to miscellaneous types of data (such as images and trajectories). Within each domain and problem formulation, the diversity of the problems and data types are significant. Therefore, a few “core” formulations will be studied for each

application domain, rather than the wide gamut of specific and detailed variations. The formulations will be mapped to broad classes of techniques covered in this book. The goal of this chapter is to teach practitioners how to *use* outlier analysis methods by defining an appropriate mapping from the problem formulation to the specific class of techniques. The specific challenges of different problem domains will also be discussed, with related research being integrated into the discussion of each application.

At this stage, a few practical insights about outlier analysis will be mentioned. These insights seem to be common across many problem domains, and therefore provide an understanding of the areas that would benefit the most from further research:

- *Dependency-oriented data is ubiquitous:* Although traditional multidimensional outlier detection is applicable in many domains, an increasing number of domains generate *dependency-oriented data* such as time series, sequences, spatial data, or network data. Such data is extremely complex and more challenging to analyze as compared to multidimensional data. This is because anomalies are usually defined in a contextual or collective sense in such data. The ability to distinguish between noise and anomalies is limited, because large amounts of data are required in order to obtain a sufficient level of statistical significance about the frequency properties of collective groups of data items.
- *Supervision is often critical in distinguishing between noise and application-specific anomalies:* A significant amount of research has been devoted towards unsupervised outlier analysis in the literature. However, when examining *what really works* in many applications, the presence of supervision is critical. This is because outliers found by unsupervised methods often correspond to noise, and may greatly outnumber the number of interesting anomalies. In such cases, the incorporation of supervision is critical. Even when labels are not available, *indirect* supervision can be incorporated into unsupervised methods by using domain knowledge during feature extraction and specific details of algorithm design. Furthermore, active learning methods can be used in order to efficiently *create* labels by combining unsupervised methods with human feedback.
- *Very simple algorithms work surprisingly well:* Even though a large number of complex algorithms have been defined for outlier detection, many simple algorithms (like the exact  $k$ -nearest neighbor method and the Mahalanobis method) seem to work better than many of these complex techniques. Greater complexity in algorithm design is often not adequately rewarded for outlier analysis. In many cases, the complex algorithms may work well for a small subset of benchmarked data sets, which gives a false sense of security about their effectiveness.
- *Ensembles should be used where possible:* Even though simple algorithms work very well, when used on a *standalone* basis, the main advantage of complex algorithms can be gained when they are used in conjunction with ensemble methods. For example, subspace outlier detection is *inherently* ensemble-centric [4, 31] and it makes little sense to use such methods on a standalone basis. In fact, almost all the accurate algorithms for subspace outlier detection, such as feature bagging, rotated bagging, subspace histograms, and isolation forests use ensembles in one form or the other. Furthermore, ensemble methods like variable subsampling and rotated bagging can be combined with virtually any ensemble method because of their generic nature. Detailed experiments and analyses of different ensemble-centric algorithms are presented in [35].



This chapter is organized as follows. Quality control applications are discussed in section 13.2. A discussion on financial applications is provided in section 13.3. Methods for using outlier analysis in Web log analytics are discussed in section 13.4. The problem of outlier analysis in the context of security and intrusion detection applications is studied in section 13.5. Medical applications are studied in section 13.6. Text and social media applications are studied in section 13.7. Earth science applications are presented in section 13.8. A number of miscellaneous applications are studied in section 13.9. A broader discussion of the key guidelines for practitioners is provided in section 13.10. A description of the software resources available for the practitioner is provided in section 13.11. The conclusions are provided in section 13.12.

## 13.2 Quality Control and Fault Detection Applications

---

Quality control applications arise often in the context of manufacturing. Outliers can be detected in such applications either in terms of the characteristics of individual objects, or in terms of the aggregate characteristics of the manufacturing process. Some examples of applications in such domains are discussed below.

**Application 13.2.1 (Quality Control)** *A manufacturing process is designed to produce widgets, which are defective with probability  $p_i$ . A specific batch of widgets of size  $n$  contains  $q$  defective widgets. The goal is to estimate the probability that the manufacturing process is behaving in an anomalous way.*

**Discussion:** This is one of the standard formulations for quality-control analysis. In many variations of this problem, specific parameters of the widget (e.g., physical parameters) may be tracked and compared to an expected mean and standard deviation. In some cases, multiple parameters may be tracked simultaneously. Depending upon the specific parameters being tracked, a variety of extreme value analysis methods from Chapter 2 can be used.

- The Chernoff bounds discussed in section 2.2 of Chapter 2 can be used to provide tight bounds on the tail probabilities when specific fractions are being tracked (e.g., fraction of defective widgets).
- The Hoeffding inequality discussed in section 2.2 of Chapter 2 can be used to provide tight bounds when strict upper and lower limits exist on tracked values.
- The  $t$ -value and normal value distributions discussed in section 2.2 of Chapter 2 can be used in order to provide approximations for the level of significance of an aggregate value (e.g., fraction of defective widgets in a sample).
- In cases, where multiple parameters are tracked (eg. a combination of length, width, weight), the multivariate distance distribution methods discussed in section 2.3.4 of Chapter 2 may be used, either on the individual parameters (dimensions), random subsets of parameters (dimensions), or the entire set of parameters (dimensions). In many instances, the anomalous behavior may be reflected simultaneously in many dimensions, as a result of which the use of multivariate analysis may provide better insights.



While aggregate analysis is an important application in quality control, numerous scenarios require the examination of a specific object for faults. This is related to the problem of fault diagnosis.

**Application 13.2.2 (Fault Detection and Systems Diagnosis)** *A running engine or industrial system is continuously being monitored on a variety of parameters such as rotor speed, temperature, pressure, performance, and so on. It is desired to detect a fault in the engine system as soon as it occurs.*

**Discussion:** The data in this domain are usually in the form of sensor data, in which continuous sensor values are tracked over time. Thus, methods for time-series analysis may be used in this scenario. However, the specific methods being used may depend upon the specific application at hand.

- In many applications, extreme values of the sensor data may correspond to anomalies. For example, very high temperature or pressure may precede the bursting of a pipe. In such cases, even the simple extreme-value analysis methods of Chapter 2 may be used without accounting for the temporal aspect of the data. However, in order to perform early detection, sudden and unusual changes are more relevant. For example, a sudden and unusual *rise* in temperature may be relevant, even when the *absolute* value of the temperature is not high. In such cases, the abrupt change detection methods discussed in section 9.2 of Chapter 9 may be used.
- In many applications, thousands of time-series may be monitored, and unusual deviations in *specific* combinations of time-series provide information about *different types* of anomalous behavior. Section 9.2 of Chapter 9 discusses the case in which a sensor failure scenario can be distinguished from a pipe rupture scenario with the use of supervision. Such scenarios almost always require supervision in order to provide the learning necessary for specific kinds of anomalies.
- Unusual *shapes* in the time-series may often provide clues about anomalous behavior. For example, an unusual vibration in the engine system may cause oscillations in the pressure values, which are abnormal. In such cases, the unusual time-series shape detection methods of section 9.3 in Chapter 9 can be used. These methods can also be generalized to multivariate time-series by transformation to trajectory data, as discussed in section 9.3.1.3 of Chapter 9.
- Unusual shapes can also be related to *specific* diagnosis of system faults by using supervised methods. In such cases, training data relating similar faults with the corresponding time series may be available. Supervised methods for unusual shape detection in time series are discussed in section 9.3.5 of Chapter 9.

The work in [241] designs a method for novelty detection, such as the detection of shorted turns in the field-windings of operating synchronous turbine-generators. Signature patterns of signals are extracted from the running motor, and are compared with the normal signals in order to detect novelties. A probability density estimation method for detection of abnormal conditions in engineering is discussed in [167]. A kernel method is used for the density estimation process. Kernel density-estimation methods are discussed briefly in Chapter 4 of this book. The method in [428] uses principal component analysis to transform the data, and then identifies the anomalies as the points that lie far away from the primary hyperplanes of projection.

Many of the above methods can also be applied to problems such as structural damage detection, in which faults in mechanical units may be diagnosed with the use of different kinds of time-series data. PCA methods have also been used for anomaly detection in spacecraft components [206]. A method that combines unsupervised and supervised learning methods for fault detection in automobile data is discussed in [207]. A detailed discussion of the use of the wavelet transformation for machine-health monitoring is provided in [432]. The use of neural networks for motor fault detection is discussed in [146]. A supervised method for motor-bearing damage detection was proposed in [482]. In many cases, streaming and online detection methods are desirable [9].

■

The diagnosis of computer systems also requires real-time anomaly detection techniques. However, the data in such cases is often discrete, and the methods used are typically similar to those in intrusion detection and security applications. This will be discussed in a later section of this chapter. For example, methods for system monitoring of large computer clusters are discussed in [470].

A related topic is that of structural defect detection, which attempts to determine structural defects in 2-dimensional or 3-dimensional objects such as a fabric, or a beam [123, 526, 527]. In such cases, measurements may be associated with each spatial location. For example, for the case of fabric fault detection, a 2-dimensional image of the fabric may be analyzed for faults [123].

**Application 13.2.3 (Structural Defect Detection)** *Given a 2- or 3-dimensional surface associated with measurements at each spatial location over time, the goal is to determine significant structural defects, either at a given point in time, or significant changes that occur over time.*

**Discussion:** The nature of the measurements in this case is specific to the problem domain. This problem is inherently spatiotemporal, since multiple spatial measurements are available at different instants in time. Different methods are possible for defining outliers.

- Unusual (spatial) changes in the attribute values on the basis of spatial locations can be used in order to detect anomalies. For example, the neighborhood algorithms discussed in section 11.2.1 of Chapter 11 can be used in order to detect outliers.
- Unusual shapes in the images implied by the attribute values often provide insights about significant patterns of defects in the data. These methods are discussed in section 11.2.4 of Chapter 11.
- The spatial analysis in the two cases can be combined with temporal analysis in order to determine significant changes in the underlying data. This corresponds to the methods discussed in section 11.3 of Chapter 11.
- In many cases, previous examples of specific defects may be available. In such cases, the supervised techniques discussed in Chapter 11 may be used.

Methods for structural defect detection are discussed in [123, 274, 526, 527].

■

A broad review of fault detection methods is provided in [554].

## 13.3 Financial Applications

---

Financial fraud is one of the more common applications of outlier analysis. Such outliers may arise in the context of credit card fraud, insurance transactions, and insider trading. This section will discuss a number of financial applications in the context of outlier analysis.

**Application 13.3.1 (Credit-Card Fraud)** *A credit-card company maintains the data corresponding to card transactions by different users. Each transaction contains a set of attributes, such as the user identifier, amount spent, geographical location, and so on. The card company may also have labeled data containing previous examples of fraudulent transactions. It is desirable to determine fraudulent transactions from the data.*

**Discussion:** One desirable aspect of credit-card applications is that labeled data is often available in order to relate the transactions with the underlying anomalies. Nevertheless, both supervised and unsupervised methods can be used for anomaly detection in such cases.

Many domain-specific characteristics of the data are used for fraud detection. For example, it is well known that transaction amounts consisting of large absolute values may correspond to anomalies. The most common technique is to build user profiles on short segments of transaction sequences. Typically, the ordering among a short segment of the transactions is immaterial. If desired, a single transaction of the user can also be used. Either a single transaction or a short sequence of transactions can be converted into a feature vector, which is compared to the user's profile. The key is to design a similarity function, which can encode the wide diversity of attribute types, the collective profile within a short segment, and domain-specific knowledge (e.g., large transactions or sudden bursts of high-value transactions are more likely to be fraudulent). It is also possible to use geographical location in order to determine the anomalousness of a sequence of transactions with respect to other sequences from the same spatial location.

The major challenge with anomaly detection in credit card data is that false positives are extremely common, and false negatives are expensive, even when rare. In other words, the receiver operating characteristic (ROC) curve usually suggests very noisy behavior of purely unsupervised methods. The quality of the inference can be improved in two ways, both of which incorporate some form of supervision:

- Domain-specific knowledge needs to be encoded into the similarity function in order to account for the differential nature of fraudulent transactions.
- When labels are available, supervision should be used in order to relate the profiles to fraudulent behavior.

In many cases, the automated analysis is combined with manual inspection in order to determine significant cases of fraud. Recently, discrete-sequence methods of Chapter 10 based on hidden Markov models have also been used [509]. In these methods, symbolic values are extracted by discretizing the credit card amounts. A survey on supervised fraud detection methods is provided in [440]. The issue of class imbalance in supervised fraud detection methods is discussed in [441]. A wide variety of methods [45, 124, 218, 440, 513, 514] are available for credit-card fraud detection, although the general experience has been that supervised methods are the most effective. This is not surprising, since supervised methods are better able to distinguish between true anomalies and noise.

■

**Application 13.3.2 (Insurance Claim Fraud)** *In this case, claims are made by different entities on the basis of insurance policies. Significant anomalies need to be discovered from the data on this basis.*

**Discussion:** Although this application shares some resemblance to credit-card fraud detection at a higher level, it is also significantly different in many ways. For example, user-specific profiles cannot be constructed, because a particular user may rarely make a claim. On the other hand, repeated claims by a single user is often an indicator of fraud, and should be incorporated as a feature during pre-processing. Unlike credit fraud applications, geographical location is often *contextually* not relevant.

Once a multidimensional representation of the claims has been created, the problem is essentially an application of multidimensional (point) anomaly detection. The key step is to extract the correct features from the insurance claim documents, which can be used in order to create an unsupervised or supervised anomaly detection system. Feature extraction in insurance claim scenarios is highly domain specific, since it requires the identification of indicators that are highly specific to the particular type of claim. For example, in a life-insurance scenario, a low lag between the initiation of the policy and the death of the subject is sometimes correlated with homicide. In a medical insurance claim scenario, the statistical distribution of the claims over different types of diseases coming from a single medical provider may be skewed when the provider is engaging in fraud. Depending upon the application, such features need to be extracted in a domain-specific way. Therefore, any of the methods in Chapters 2, 3, 4, 5, 6, and 7 may be used once the feature extraction phase has been performed.

Labels are usually available since previous examples of fraud are available. In order to obtain high quality prediction, it is critical to encode the information about previous examples, either in the form of the learning algorithms discussed in Chapter 7, or indirectly by using feature representations that distinguish fraudulent claims from normal ones. Methods for insurance fraud detection are discussed in [166, 440, 556]. In particular, a comprehensive bibliography may be found in [440].



Many financial organizations also track the user behavior at their Web sites. These correspond to discrete sequences that can be analyzed in order to determine significant instances of fraud. These cases will be analyzed in section 13.4 of this chapter on Web log analytics.

**Application 13.3.3 (Stock Market Anomalies)** *The financial tickers of the different stocks and options correspond to time-series data streams. In some cases, significant anomalies may be created by external events. The early detection of such events may be useful in the determination of unknown influencing factors such as insider trading, or automated stock trading glitches (e.g., the flash crash of May 2010).*

**Discussion:** In many cases, external information such as news streams (e.g., *Google News*) are also available for event detection. This is particularly useful for applications such as insider trading detection, where unusual temporal ordering between events in the news and events in the stock tickers can be used in order to detect insider trading. For example, if an anomalous change (in value or transaction volume) of the relevant stock ticker precedes an event for the ticker in the news stream, then this can be used as an indicator of insider trading.

In other cases, such as the unusual behavior of the stock market during the flash crash of May 6, 2010, direct time-series analysis may be used. Such methods are discussed in Chapter 9. Both deviation-based contextual point anomalies and time-series shape-based collective anomalies provide insights about the unusual interactions. Deviation-based anomalies are more useful for early detection, whereas shape-based anomalies are more useful for detailed diagnosis, which is slightly delayed. In many cases, it may be desirable to use streaming methods [9] in order to determine the anomalies in real time. Methods for early detection of insider trading in financial markets are discussed in [172]. Some of the methods for multidimensional change detection discussed in Chapter 9 are useful for tracking other aspects of stock activity, such as specific stock orders or the volume distribution of stock orders. Methods for distribution change detection in stock order data streams are discussed in [372].

Another interesting method for detecting *regime anomalies* [76] from time-series data streams can be used in order to detect sudden changes in the dependencies among different streams. This provides an idea of scenarios in which the relationships among the different stocks have changed over time.

■

Financial entities often interact with one another. Examples include producers and suppliers, customers with sellers, and customers with each other. In such cases, anomalous patterns of interaction can provide useful insights. This leads to the interesting problem of detecting anomalies in financial and customer interaction networks.

**Application 13.3.4 (Financial Interaction Networks)** *A set of financial entities  $V$  are continuously interacting with one another over time. Values on the edges may correspond to the intensity or volume of the interactions. Unusual anomalies may need to be determined in such cases.*

**Discussion:** Financial interaction networks are ubiquitous, and the interactions between different financial participants are often tracked in order to obtain competitive knowledge about the interactions. In some cases, such as mobile phone fraud, the interactions may not specifically correspond to financial transactions, but rather to an interaction between two customers. In most of these cases, values on the edges are available for analytical purposes. The temporal change detection methods in section 12.5 of Chapter 12 can be used in order to identify the relevant regions of change in the network. The challenge in using such methods is that the values on the edges are often critical to the anomaly detection process. For example, high values of the transactions may correspond to anomalies. It is relatively easy to generalize the spectral methods of Chapter 12 to include the values on the edges in the analysis. Such methods are discussed in [280, 519, 520, 522]. Different methods for fraud detection in the context of mobile phone networks are discussed in [275, 418, 440, 441, 536].

■

## 13.4 Web Log Analytics

---

Web logs often contain significant information about security breaches and other kinds of anomalous activity. For example, a bank may keep logs of its Web site accesses. Unusual patterns of accesses may correspond to anomalous activity.

**Application 13.4.1 (Web Log Anomalies)** *Given a sequence of accesses in a Web log, the goal is to determine the unusual patterns of accesses in this log.*

**Discussion:** Web logs are typically pre-processed into a set of user-specific discrete sequences. These discrete sequences correspond to the identifiers of the pages accessed by the users. Many challenges arise during pre-processing, since users can often be distinguished only at the level of their IP addresses. Nevertheless, even in such cases, user-sessions can often be mined from the logs. The initial phase of pre-processing is crucial in such applications, because a single undifferentiated log is provided. This log needs to be decomposed into user-sessions, and then further decomposed into test sequences, and comparison units, as discussed in Chapter 10. General issues related to Web log data preparation are discussed in [150].

In many cases, additional domain knowledge is available about *relevant* sequences. For example, a repeated sequence of accesses to *login* and *password* pages may be indicative of anomalous behavior. In other cases, examples of anomalous discrete sequences may be available. Where possible, such domain knowledge should always be used, because it significantly improves the quality of the underlying results.

Numerous methods discussed in Chapter 10 are applicable to this case, depending on the types of outliers that need to be found.

- Position outliers can be used to determine unexpected accesses. These are contextual anomalies detected by the method, and correspond to a single unpredictable access, which is an outlier because of its *relationship* to adjacent and neighboring values. Markovian and rule-based models are typically used for outlier detection in discrete sequences. Such methods are useful for early anomaly detection, when a single unexpected web access is sufficient to arouse suspicion.
- Combination outliers are useful for determining unusual subsequences in the test sequence. This can be achieved using either unsupervised or supervised methods. In the case of supervised methods, the extraction of relevant features such as  $k$ -grams is crucial for effective anomaly detection. In the case of unsupervised techniques, window-based nearest neighbor and hidden Markov models are typically used.

Methods for anomaly detection in web logs are discussed in [169, 329].

■

Web logs are often analyzed in the context of a wide variety of security and intrusion detection algorithms. These methods will be discussed in this section. In the former case, operating system call traces are analyzed on a particular computer system, whereas in the latter case, the network data is analyzed for anomalies.

## 13.5 Intrusion and Security Applications

---

Intrusions correspond to different kinds of malicious security violations in a computer system. The data is typically streaming, and arrives at a high rate. Intrusions correspond to anomalous events, which need to be inferred from the underlying data. Denning [165] classified intrusions into *host-based intrusion detection systems* and *network-based intrusion detection systems*. These cases are somewhat different both from the perspective of data representation and temporal locality. In general, the former involves the analysis of discrete

Table 13.1: Some examples of user commands in a UNIX system

Time	Hostname	Command	Arg1	Arg2
AM	Sayani-T61	mkdir	dir1	
AM	Guardian	more	file1	
AM	Guardian	cp	file1	file2
AM	Sayani-T61	cd	dir1	
AM	Guardian	find	dir2	-print
AM	Sayani-T61	vi	file1	

sequences with high temporal correlations, whereas the latter involves the analysis of multidimensional streams with (relatively) limited temporal correlations. Therefore, different models are typically used in these cases. Each of these cases will be discussed in this section.

**Application 13.5.1 (Host-based Intrusions)** *Operating-system call traces are available in a computer system, which are symbolic sequences. Anomalous subsequences in these traces correspond to malicious computer programs. It is desired to determine anomalous sequences from these traces.*

**Discussion:** The data in this case are similar to Web logs at a conceptual level, in that it corresponds to symbolic sequences. In this case, operating-system call traces are used instead of web log traces. The calls could correspond to either operating system calls or user calls. Thus, the calls form the base alphabet  $\Sigma$  over which the mining is performed. Different kinds of programs execute different sequential combinations of calls. Therefore, the sequential ordering of the calls provides critical information in order to distinguish between normal and malicious programs. These calls could either be at the user-command level, or they could be at the operating system level. The latter is much more granular, and that can sometimes make it more difficult to mine such sequences. Some examples of user-level calls in an operating system are illustrated in Table 13.1.

During the feature-extraction phase, the logs are transformed into symbolic sequences. In many cases, when the commands are coming from multiple sources, they may need to be separated out into their different hostnames. For example, in the case of Table 13.1, the sequences of commands from the hostnames *Guardian* and *Sayani-T61* need to be separated out into different sequences in order to examine the malicious behavior of a particular host. This is similar to Web log analytics, in which sequences that are specific to each Web user are constructed in the pre-processing phase. Sometimes, the choice of the symbols used in the sequence may also depend upon the application at hand. For example, should only the user command be used, or should some combination of user command and argument be used? These choices are highly application-specific, and the quality of the final results may be sensitive to such choices.

Subsequently, any of the discrete sequence methods discussed in Chapter 10 can be used. In fact, the different kinds of scenarios are very similar to those in Application 13.4.1 on Web log analytics. The reader may refer to the details of the specific methodologies used in Application 13.4.1. In spite of the very different data domains in these cases, it is interesting to see that the underlying methods are often interchangeable.

A comparison of different methods for intrusion detection in these scenarios is provided in [158]. Numerous methods have been proposed in the literature for this scenario, and are



discussed in [158, 188, 197, 198, 199, 200, 201, 216, 217, 272, 338, 339, 340, 349, 350, 351].

■

A second class of methods is that of network-intrusion detection systems, in which the intrusions are inferred from network data. The data on the network can be of different types, depending upon the level of abstraction at which it is presented. For example, the data could correspond to the underlying packets on the network, and the intrusions may result in subtle changes in the multidimensional features extracted from these packets.

**Application 13.5.2 (Network Intrusion Detection)** *Given a stream of network packets or data records, the goal is to determine network intrusions.*

**Discussion:** The temporal relationships between data records are much weaker in this case than in the case of host-based systems, in which the sequential ordering of calls is critical in identifying intrusions. Furthermore, each individual record in this case is multidimensional, and contains the features extracted from the unit of network data (e.g., packet), or raw *tcpdump* data. For example, a common feature used is the number of bytes transferred, which is continuous. Other attributes are discrete. Thus, this problem can be modeled as a multidimensional stream of records, containing both continuous and categorical attributes. An example of a network-intrusion data set is illustrated in Table 13.2. Only five of the basic features are shown. This is the well known *KDD Cup 1999 Intrusion detection data set* [203], which contains a combination of symbolic and continuous attributes. These features are of three types that correspond to the basic characteristics of the connections (service, protocol, bytes transferred etc.), the content characteristics of the connections suggested by domain knowledge (e.g., number of “hot” indicators, failed login attempts), and the traffic characteristics (e.g., number of connections, or the number of connections with specific kinds of errors).

Most of the unsupervised multidimensional outlier detection methods can be generalized to this case. Furthermore, in cases where stream processing is required, the multidimensional streaming outlier detection methods of Chapter 9 may be used. In particular, aggregate change-points may often be helpful in identifying network-wide traffic anomalies. Such change points often correspond to network intrusions and attacks [377, 334, 335]. Since the data is often of a mixed nature, many of these algorithms need to be modified using the general methodologies discussed in Chapter 8 for adapting unsupervised algorithms to mixed data sets.

In some cases, a subset of the data may be labeled, and may correspond either to normal data or to intrusions. The labeled intrusions can never be exhaustive, since new intrusions may always arise over time. Nevertheless, labeled data about existing intrusions is useful for identifying repetitive attacks. At the same time, it is important to also detect novel classes as new intrusions arise. Such scenarios can be addressed using the streaming and supervised novel-class detection methods discussed in section 9.4.3 of Chapter 9. Methods for network-intrusion detection are discussed in [55, 56, 70, 71, 72, 145, 188, 288, 352, 330, 331, 381, 382, 420, 484, 544, 589]. A comparative study of network-intrusion detection schemes may be found in [345].

■

Table 13.2: Examples of five basic features from the network connection records from the *KDD Cup 1999 Network Intrusion Data Set* [203]

Duration	Protocol	Service	Src.Bytes	Dest.Bytes
5	<i>tcp</i>	<i>telnet</i>	183	3855
5	<i>tcp</i>	<i>telnet</i>	183	3855
0	<i>tcp</i>	<i>http</i>	298	2239
0	<i>udp</i>	<i>private</i>	105	146
0	<i>udp</i>	<i>domain.u</i>	44	44
0	<i>tcp</i>	<i>http</i>	188	2199
0	<i>icmp</i>	<i>ecr_i</i>	508	0

## 13.6 Medical Applications

Medical applications typically use different types of diagnostic tools for predictive modeling. Two of the most common kinds of data that are encountered for predictive modeling of medical data are in the form of sensor data (e.g., electrocardiography), and spatial data (e.g., positron emission tomography). Both of these are different types of contextual data. Each of these different cases will be addressed by a different application definition.

**Application 13.6.1 (Medical Sensor Diagnostics)** *Given a set of sensor readings from a given patient, the goal is to determine if the patient has a disease condition.*

**Discussion:** Both supervised and unsupervised methods can be used in order to process medical data. For the unsupervised case, the problem formulation is similar to that of fault diagnosis, except that the nature of the sensor readings are specific to the medical domain. Therefore, all the methods from Chapter 9 can be used for this case as well. In the simplest case, extreme or unexpected value analysis on medical time-series or data distributions may be used [276, 343, 463, 502] in order to determine anomalous values. An approach that uses a probabilistic mixture model is discussed in [528]. Time-series containing subsequences of unusual shapes [360] may also be useful for identifying more complex anomalous conditions.

In the context of medical data, the cost of missing a positive is high, and the diagnosis needs to be specific. Furthermore, any anomalous conditions may need to be reported in real time. In many cases, a specific diagnosis may be distinguished from another potentially incorrect diagnosis by using the signals from multiple sensor data streams. In this context, a supervised method for deviation-based anomaly detection in multivariate time-series data streams was proposed in [9]. Methods for shape-based supervised anomaly detection are discussed in Chapter 9.

Supervised methods are particularly desirable in the medical domain because of the specificity requirements of a diagnosis. In many cases, expert knowledge [50] may need to be combined with the mining algorithm in order to ensure the most effective results. Semi-supervised methods for medical time-series classification are discussed in [564]. The problem of supervised shape discovery in time-series data is discussed in section 9.3.5 of Chapter 9. ■

Another common diagnostic tool used in the medical domain is that of *imaging*. In these cases, an magnetic resonance imaging (MRI) or positron emission tomography (PET) scan

is used to create a 2-dimensional or 3-dimensional image of a part of the body such as the brain. Anomalies in these shapes correspond with significant medical conditions.

**Application 13.6.2 (Medical Imaging Diagnostics)** *Given a multidimensional image of an affected body part, the goal is to determine whether a patient has a disease condition, and what that condition is.*

**Discussion:** This is a classic example of a spatial application that contains both contextual and behavioral attributes. The discovery of anomalies in such data has been addressed extensively in Chapter 11. The most crucial part of the feature extraction is to convert the shape [602] into a time series using the methods discussed in section 11.2.4 of Chapter 11. Subsequently, a variety of unsupervised or supervised methods can be applied to the extracted time series. The problem typically reduces to one of the following formulations:

- Anomalous shapes may correspond to specific disease conditions such as tumors, multiple sclerosis lesions, mammography, or the degraded brain regions of an Alzheimer patient [448, 553, 505, 537]. In the semi-supervised case, examples of normal shapes may be available, and it may be desirable to determine shapes that are very different from these normal profiles. The method of [602] may be used to convert the shapes into a time series, and then anomaly detection can be applied to this time series. For example, these anomalies may be identified using the approach given in [565]. This method has also been described in section 11.2.4 of Chapter 11.
- In many cases, previous examples of anomalous regions may be available. These examples can be used to train a classifier to learn the relationship between the shape and the specific disease condition. A variety of shape learning methods [69, 115, 375, 602] are available for labeled data. A brief review of such techniques is provided in section 11.2.4.4 of Chapter 11.
- In some cases, a temporal sequence of images from the same patient over many years may be available, and it may be desirable to determine *changes* in the shapes of the images [86]. This corresponds to shape-change detection methods discussed in section 11.2.4.5 of Chapter 11.

A variety of shape analysis methods have been used frequently in the medical domain for image diagnostics [86, 251, 448, 553, 477, 505, 537, 543].

■

## 13.7 Text and Social Media Applications

---

Text and social media applications are extremely common because of the ubiquity of text data in social interactions such as email, the Web, and blogs. Some of these methods have already been discussed in Chapter 8.

**Application 13.7.1 (Event Detection in Text and Social Media)** *Given a document corpus  $\mathcal{D}$ , the goal in unusual topic detection is to determine unusual documents that differ significantly from the trend. In first story detection, a stream of documents is available, and it is desirable to determine unusual events corresponding to new topics in the stream of documents. In the context of social media applications, such streams may be generated by user activities such as tweets.*

**Discussion:** This scenario has already been discussed extensively in Chapter 8. Both supervised [584, 586] and unsupervised methods [29, 47, 48, 49, 95, 306, 507, 508, 585, 608, 622] may be used. The reader is referred to Chapter 8 for details.

A particularly important special case is that of social media streams in which the user activities such as tweets may provide early knowledge of unusual events. In many cases, unusual events in localized regions may show up in social media feeds well before traditional news media, because of the wider authorship of social media sites. Such events will typically be manifested as changes in the topical and linkage distributions of the social media feeds. Both supervised and unsupervised techniques are relevant in such cases, depending upon the availability of training data. In the supervised case, it may be desirable to determine events of specific types. Methods for finding unusual events or changes in text and social streams are discussed in [30, 435, 562]. Both supervised and unsupervised methods for event detection in social media streams are discussed in [30].

■

Another common application in email streams is to detect spam in a sequence of emails.

**Application 13.7.2 (Spam Email)** *Given a stream of emails, the goal is to identify the subset of emails that correspond to spam.*

**Discussion:** While unsupervised methods for unusual topic detection can be used, the results are often likely to be inaccurate. While spam is still a small fraction of the mail in most cases, the volume is large enough to make it difficult to detect with unsupervised methods. This case is best addressed with the use of supervised methods, where the specific features of the emails are learned, and related to spam labels in the training data. Any of a variety of methods for text classification [24] can be used.

A lot of additional domain knowledge is available, which helps determine whether a particular email message is junk or not. For example, an email would be more or less likely to be junk or spam according to some of the following characteristics:

- The domain of the sender can make an email to be more or less likely to be junk. For example, an email address ending in *.edu* is less likely to be junk.
- Phrases such as “*Free Money*” or overemphasized punctuation such as “!!!” can make an email more likely to be junk.
- Whether the recipient of the message is a particular user or part of a larger mailing list can influence the underlying possibility that the email is junk or spam.

The Bayes classifier for text provides a natural way to incorporate such additional information into the classification process by creating new features for each of these characteristics. A method such as this has been discussed in [469]. A survey of methods for email spam filtering may be found in [85, 152].

■

In many social networks, a significant percentage of the links are noisy, and may not provide any useful insights for analysis. Such links are often caused by spam links on the Web to increase search-engine rating, or by low-quality links across weakly related entities.

**Application 13.7.3 (Noisy and Spam Links)** *Given a social network with content at the nodes, determine the noisy and spam links with the use of structure and content information.*

**Discussion:** This problem is discussed extensively in the sections 12.3.2 and 12.4 of Chapter 12. The first section discusses methods for determining linkage outliers on the basis of structure only, whereas the second section discusses methods for determining linkage outliers with the use of both structure and content. Such edges in a social network correspond to relationships that are weak, and often harm the effectiveness of social media algorithms. Techniques for determining such outliers are discussed in [17, 196, 212, 452, 214]. In many cases, supervised methods may be used in order to learn link spam, when labeled information is available. ■

Many types of networks (blogs, bibliographic networks, social networks) contain both texts and link information. Discovering significant patterns of change that constitute anomalous activity in these types of networks may be helpful.

**Application 13.7.4 (Anomalous Activity in Social Networks)** *Given an evolving network with associated text content at the nodes, the goal is to determine the anomalous regions of activity or change in the network.*

**Discussion:** This problem is related to that of evolution of communities in the underlying network. It is possible to use a purely structural approach with the use of the community change analysis methods discussed in section 12.5 in Chapter 12. In the case of blogs and social networks, a significant amount of content is also available in the network. In such cases, community detection algorithms can be enhanced with the use of node content [12]. In addition, a variety of spectral methods [280, 519, 520, 522] can be used, when it is required to use only the linkage structure for analysis. A method that specifically monitors evolving blogs for significant change is discussed in [415].

Social networks are particularly useful tools for the discovery of threat activity such as terrorist interactions. A method for finding threat activity in social networks with the use of eigenspace analysis was proposed in [398]. ■

## 13.8 Earth Science Applications

---

Outlier detection is used in numerous weather, climate, or vegetation cover applications, where anomalous regions are detected in spatial data either at a single snapshot or over time. Therefore, many of these applications are spatial or spatiotemporal in nature. For example, sea surface temperatures are often tracked in order to determine significant and anomalous weather patterns.

**Application 13.8.1 (Sea Surface Temperature Anomalies)** *The temperatures on the sea surface are tracked continuously over time. It is desired to determine:*

- (a) *unusual localized spatial variations in temperature on the sea surface,*
- (b) *regions on the sea surface containing unusual shapes with homogeneous temperature,*
- (c) *sudden and unexpected changes in sea surface temperature that are localized to a specific region, and*
- (d) *the relationship of the spatial temperature patterns to known weather events for predictive modeling.*

**Discussion:** This is a typical spatial or spatiotemporal formulation that arises often in the context of meteorological applications. In this case, the spatial and temporal coordinates are the contextual attributes, whereas the temperature is the behavioral attribute. It requires the determination of both contextual and collective outliers from the data. The determination of unusual spatial variations can be performed by finding neighborhood-based outliers. Such outliers are discussed in section 11.2.1 of Chapter 11.

Unusual shapes on the sea-surface temperature profile can be performed by applying the unusual shape discovery method discussed in section 11.2.4 of Chapter 11. The temperatures may need to be discretized into buckets in order to convert the continuous values into discrete shape contours. The contour of a region with the same discretized value may be used for the anomaly-detection process.

The neighborhood-based spatiotemporal change detection algorithms of section 11.3 in Chapter 11 may be used in order to determine significant outliers. These correspond to specific *points* in space and time at which there are unusual temporal *or* spatial variations. In some cases, it may be desirable to determine significant changes in the *shapes* of temperature patterns. The detection of significant changes in the patterns can be performed by using the shape-change detection methods discussed in section 11.2.4.5 of Chapter 11.

Such characteristic patterns in different kinds of behavioral attributes such as temperature, pressure, or humidity can often be related to unusual weather events such as cyclones. However, such anomalies are best determined with the use of supervised methods in which previous training data relating the weather patterns to the spatiotemporal data is available. Supervised methods for classification of such data are discussed in detail in [69, 115, 375, 602].

■

While the aforementioned application was presented with the use of temperature as a behavioral attribute, a variety of other attributes such as pressure or humidity may be tracked for anomaly detection. As an example, the work in [569] tracks precipitation patterns in order to determine unusual regions of change. Methods for finding region outliers in meteorological data are discussed in [615]. In many cases, multiple behavioral attributes may be available. This is a more challenging case, because it is desired to determine unusual combinations of behavioral attributes. In such cases, a simple approach is to perform the analysis separately on each attribute and combine the anomaly scores. The real-time spatiotemporal analysis of weather patterns can provide predictions of significant events such as hurricanes.

A closely related problem is that of *land-cover anomalies*. The type of land cover or vegetation is often tracked with the use of remote sensing. Virtually all the aforementioned methods for finding meteorological anomalies can also be applied to the problem of land-cover anomalies.

**Application 13.8.2 (Land Cover Anomalies)** *The land cover at different spatial locations are tracked continuously over time. It is desired to determine:*

- (a) *unusual localized spatial variations in land cover type,*
- (b) *regions containing unusual shapes with homogeneous land cover,*
- (c) *sudden and unexpected changes in land cover that are local to a specific region, and*
- (d) *the use of changes in spatial land-cover patterns to uncover unusual and unknown geological, climate, human or wildlife activity.*

**Discussion:** This case is virtually identical to that of uncovering unusual sea-surface temperature anomalies. The major difference is that the land-cover type is the behavioral attribute, which may be discrete. Therefore, the key is to design a similarity function which relates different land cover types to one another. For example, certain types of land cover are more likely to be adjacently located than others. In order to determine such similarity values, the methods [154] for contextual similarity discussed in section 8.4.2 of Chapter 8 may be used. Once such contextual similarity measures have been determined, they can be used in order to determine significant point changes in the data. An example of a recent application to determine significant vegetation changes is discussed in [341]. Methods that correlate land cover changes with other kinds of parameters (such as climate) are discussed in [65, 464].

■

## 13.9 Miscellaneous Applications

---

This section briefly covers miscellaneous applications for outlier detection that do not belong to any of the aforementioned categories.

**Application 13.9.1 (Data Cleaning)** *Given a data set, remove discordants from it. Correct any errors in the data if possible.*

**Discussion:** This is one of the classical applications of outlier analysis, and is often addressed effectively with unsupervised methods. Virtually all the unsupervised methods discussed in this book can be used for noise *removal*. Many of the autoregressive models introduced in Chapter 9 are used for removal of erroneous values from sensor data. For example, the removal of noisy links in social networks can be considered a data cleaning application. Such methods are discussed in Chapter 12.

For noise *correction*, methods such as PCA can provide the best insights. For example, it has been shown in [21, 425] that the use of PCA and SVD methods can be used in order to improve the representation quality of data sets for mining and retrieval. Methods for removing outliers in the context of regression analysis are discussed in [467].

■

A number of applications are designed to determine anomalies in spatiotemporal data. Such data may correspond to movement patterns for wildlife or vehicular traffic.

**Application 13.9.2 (Traffic and Movement Patterns)** *Given a set of entities with trajectory patterns, determine significant outliers from the patterns.*

**Discussion:** This problem arises often in the context of either tracking wildlife with RFID tags, or tracking vehicles with GPS receivers. This problem can be addressed using either spatiotemporal trajectory-mining methods or by network-mining methods depending on how the data is represented. For example, when it is desirable to determine anomalous *entities* in terms of movement patterns, the trajectory mining methods [347] discussed in section 11.4 of Chapter 11 may be used. In some cases, supervision may be used [355] in order to improve the quality of the discovered patterns. Online algorithms for finding outliers in movement patterns are proposed in [102].

In many cases such as traffic data, the movement values are available on an aggregated basis because of privacy concerns. Furthermore, the movement patterns are often associated

with a network corresponding to the road network in the underlying data. In such cases, only the flow values on the different road segments may be available. Significant regions of congestion or anomalous behavior may need to be identified. Methods for finding such anomalies are discussed in [400].

■

One of the most common domains for anomaly detection is image analytics. This was discussed earlier in the context of medical image diagnostics.

**Application 13.9.3 (Image Data)** *Given a set of (possibly labeled) images, or a temporal sequences of almost identical images, it is desirable to determine:*

- (a) *images with anomalous shapes;*
- (b) *changes in the underlying patterns in cases where temporal snapshots are unavailable; and*
- (c) *prediction of a rare class in cases when some images are labeled with this class using training data.*

**Discussion:** The techniques used for this case are similar to those discussed earlier in this chapter in the context of the medical image diagnostics application. Generic image representations also have a number of attributes such as color or texture, which can be leveraged in order to determine anomalies. As discussed earlier, image diagnosis methods are used frequently in the medical domain for determination of anomalies. In the context of anomalous shape discovery, the feature extraction for *shape representation* is the most important. An example was discussed in section 11.2.4 of Chapter 11, where it was shown how to convert a shape into a time-series for further analysis. An extensive discussion of such feature transformation methods is provided in [602]. General references on image outlier detection may be found in [86, 251, 448, 553, 477, 505, 537, 543].

■

Numerous other applications of outlier analysis may be found in the domains of aviation safety [156], internet routing updates [447], malicious URL detection [380, 613], disease outbreaks [568], spatial linking of criminal incidents [370], failure management of large computer clusters [470], astronomical data [177, 261, 577], disturbance events in terrestrial ecosystems [444] and biological sequences [365].

## 13.10 Guidelines for the Practitioner

---

The examples discussed in this chapter illustrate that the diversity in specific problem setting is rather large across different domains. However, many of these models map into the same set of problems. Some critical observations that arise in the context of outlier detection are as follows:

- *Data normalization is important:* A common mistake made by many practitioners is to forget to normalize the data before applying outlier analysis algorithms. Consider an application containing an *Age* attribute (less than a hundred), and a *Salary* attribute (in order of tens of thousands). The use of proximity-based or linear models on such data (without normalization) will be dominated by the *Salary* attribute, and the *Age*



attribute will be almost ignored. Typically, each attribute value needs to be divided by its standard deviation (over the entire data set). This ensures that the different attributes are given an equal level of importance in the outlier analysis process. This process is also referred to as *standardization*, in which each attribute is first mean-centered by subtracting the mean. Then, the values in each attribute are scaled to unit variance.

- *Noise versus interesting anomalies:* Most application domains contain noisy or incomplete data or data that has errors. For example, sensor data often contains noise because of defects in transmission or failure in the data-collection hardware. As discussed in this chapter, data cleaning is itself a key application of outlier analysis. Therefore, it is critical to design a pre-processing phase that can filter out or correct such noise from the data when possible. This can be achieved using a variety of domain-specific methods, which have knowledge of the noise generation process. For example, in the context of sensor data, such noise can often be corrected or filtered by a variety of methods [22]. Nevertheless, in many cases, the use of such filtering methods can also mask interesting anomalies in the data.
- *The feature extraction phase is crucial:* In many domains, the base data is not necessarily specified in a way that can be used directly with an outlier analysis application. For example, in an insurance application, the documents containing details of the claims may be available. In a credit-card fraud application, raw transaction data may be available. In such cases, feature extraction *should implicitly use domain knowledge* to the extent possible. For example, when it is known that large transactions are more important, a feature corresponding to transaction size should be incorporated. Although numerous dedicated feature selection methods exist for other data mining problems such as clustering and classification, this does not seem to be the case for outlier analysis. This is because feature-selection methods require the determination of aggregate trends relating the features to application-specific aspects of the data. On the other hand, since outliers are based on rare observations, aggregate trends are hard to determine in a way that would be relevant for outlier analysis. Therefore, the incorporation of a domain-specific understanding is often the only way to extract meaningful features for outlier analysis. A proper selection of features can also help distinguish noise from true anomalies.
- *Domain knowledge is often easy to incorporate into unsupervised algorithms:* One of the challenges of outlier analysis is that labeled data is rarely available. However, an *indirect* form of supervision is to incorporate domain knowledge into unsupervised algorithms. Such changes require minor modifications to the details of the underlying algorithm such as the similarity function design, or the design of the transition architecture of a hidden Markov model. Some examples of incorporating domain-specific knowledge are as follows.
  - In a credit-card fraud application, the absolute amount spent is known to be a key indicator of fraudulent behavior. A  $k$ -nearest neighbor algorithm can be modified, so as to treat neighbors with higher or lower values of the amount spent in a purchase in a differential way. This knowledge can be incorporated directly into the distance function, without making any other change to the algorithm.
  - In a security monitoring application, specific sequences such as *login password login password* may be known to be indicative of attacks. Such sequences can be

provided higher importance during the construction of the comparison units for sequence anomaly detection.

- The states of a hidden Markov model should reflect an understanding of the process, rather than using a black-box  $k$ -state model, in which all transition probabilities are learned.
- *Labeled data should be used where possible:* This is the easiest way to distinguish between noise and anomalies. *Even a small amount of labeled data can significantly improve the effectiveness of outlier analysis algorithms.* Unfortunately, in many real applications, labeled data is not available. In fact, the unavailability of labeled data is the *raison d'être* for unsupervised outlier analysis.
- *Exploratory and visual analysis can be helpful at all stages of outlier analysis:* One challenge in outlier analysis is that it is often difficult to know which model may work most effectively for a given problem. Should a proximity-based model be used, should a linear model be used, or should a subspace model be used? In this context, visual analysis of the kind introduced at the beginning of Chapter 3 can provide some insights about the distribution of the data. This can often provide an idea of the specific choice of model that might work best for a particular application.
- *A human in the loop can more easily generate labels in conjunction with unsupervised outlier analysis algorithms:* It is hard to generate labeled data in outlier analysis algorithms, because anomalies are rare, and therefore positive examples are often hard to obtain. Furthermore, the manual examination of large amounts of data for anomalies is akin to searching for a needle in a haystack. However, unsupervised and supervised algorithms can be used in an iterative way in conjunction with a human in the loop in order to generate labels. This corresponds to the active learning framework discussed in Chapter 7. Such frameworks can be extremely useful in converting unsupervised outlier detection problems into supervised rare-class detection problems.
- *Outlier ensembles can be used to reduce risk:* The choice of a specific algorithm is a “risk” to the practitioner because the performance of a particular algorithm on a data set cannot be known even after the fact for an unsupervised problem like outlier detection. In such cases, ensembles can be used to combine different models and also reduce the variance of a specific algorithm on a particular data set. By combining a small number of models that are known to work well across a wide variety of data sets (see next section), it is possible to consistently obtain results of good quality. Combining these techniques with methods like subsampling and rotated bagging can further improve the result quality. From this point of view, the ideas in Chapter 6 are very useful to leverage repeatedly across various data domains.

The aforementioned recommendations seem to suggest that the incorporation of supervision and domain knowledge is possible, even when fully labeled data are not available. A careful domain-specific design of the feature selection and algorithmic processes is critical in obtaining the most informative outliers.

### 13.10.1 Which Unsupervised Algorithms Work Best?

It is impossible to identify the optimal algorithm in the absence of ground truth. Since the outlier detection problem is unsupervised, the effectiveness of a particular algorithm on a

data set depends on the “blind luck” of how well the model of normal data reflects the true distribution. Nevertheless, it has been shown [32, 35] that some simple algorithms tend to work well across a broad range of real-world data sets. In fact, testing with real-world data sets seems to show that more complex algorithms do not always yield the performance gains that one would expect from their sophistication. The experiments in [32, 35] seem to suggest that algorithms such as the exact  $k$ -nearest neighbor method, the average  $k$ -nearest neighbor algorithm, the nonlinear Mahalanobis method, and isolation forests seem to perform extremely well. Although there is some distinction between the base performances of these algorithms, they tend to become more similar when they are combined with a method like variable subsampling. In the case of isolation forests and the nonlinear Mahalanobis method, the use of an ensemble-centric implementation is essential.

It is also noteworthy that the linear Mahalanobis method and (the basic version of the) isolation tree algorithm are virtually parameter-free; this makes their applicability particularly simple. Although the raw distance-based algorithms are not parameter-free, their application in conjunction with variable subsampling (Chapter 6) ameliorates this problem to some extent. This is because the variability in the size of the subsample often eliminates the unpredictable effect of a particular value of the parameter  $k$  in the  $k$ -nearest neighbor algorithm.

An important point needs to be kept in mind about the relative evaluation of algorithms that depend on parameters. In some cases, a particular algorithm might perform better than another algorithm at the *best* value of the parameter, but it might not perform as well across a reasonable range of parameters. This is particularly true when the algorithm is *unstable* across the choice of parameters, and therefore performs well only at specific values. In such cases, it is particularly important to use a more stable algorithm that shows better performance across a wider range of parameters. For example, as shown in [32], the well-known LOF algorithm [96] does not seem to perform as well as a simple average  $k$ -nearest neighbor algorithm, when tested *across a wider range of parameters*. Although the performance at the best value of  $k$  for LOF might be better than that at the best value of  $k$  for the raw distance-based methods, it is hard for the analyst to know the correct values of  $k$  for unsupervised settings. In this sense, the raw distance-based detectors are often superior to LOF in real-world settings because they provide stable and high-quality solutions in a *large range of values* of  $k$ . It is also shown [221] that raw distance-based methods tend to be more stable than LOF across the choice of *data sets*. If the data set predominantly contains global outliers, then LOF might give too many false positives.

A detailed evaluation of some of the key outlier detection algorithms is provided in [35]. In the following, we discuss an overview of three key algorithms that seem to be perform well in many settings, along with their primary advantages:

1. **Linear and Nonlinear Mahalanobis method:** The Mahalanobis method is discussed in section 3.3.1 of Chapter 3. The Mahalanobis distance reports the outlier score of a data point as its Mahalanobis distance from the centroid of the data. The approach combines the benefits of multivariate extreme-value analysis and principal component analysis. Furthermore, the approach can be combined with kernel representations for complex data types or nonlinear distributions; in such cases, the approach tends to outperform many kernel models like one-class SVMs and support-vector data descriptions (SVDD). The nonlinear Mahalanobis method tends to work well in many distributions where conventional methods work poorly. This is because of its ability to adapt to the shapes of various data distributions. The approach can be implemented efficiently as an ensemble-centric method [35] on samples of data with size between 50

and 1000 points. The kernel bandwidth is set at thrice the median pairwise distances between points, which is estimated by sampling. An ensemble-centric implementation of the nonlinear Mahalanobis method has been proposed in [35] and has been shown to achieve excellent results. Although the linear Mahalanobis method can be used efficiently as a standalone method, it is crucial to use the nonlinear method only in an ensemble-centric setting. The nonlinear Mahalanobis method generally tends to be more robust and accurate than the linear Mahalanobis method in such settings.

2. **Raw distance-based algorithms:** The exact and average  $k$ -nearest neighbor algorithms are described in section 4.3.1 of Chapter 4. These methods are very simple to implement and often outperform methods like LOF, when tested across a wider range of parameters. However, it is possible for LOF to outperform these methods if the parameters are carefully chosen. Another issue is that raw distance-based algorithms tend to be more robust across a wider range of *data sets*, whether the outliers are global or local [221]. In comparison, algorithms like LOF tend to find too many false positives on data sets containing global outliers. As discussed in section 1.7.2 of Chapter 1, it is more important for unsupervised methods to work well across a wider range of parameter settings and data sets because of the inability to perform model selection and parameter tuning in the absence of ground truth. The merits of raw distance-based methods are discussed in [32, 35].

There are, however, two primary drawbacks with the entire category of neighborhood-based methods. The first drawback is that such methods have quadratic complexity with the number of points. The second drawback is that these methods can be sensitive to the choice of the parameter  $k$ . Both issues can be partially solved with the use of the variable subsampling method discussed in Chapter 6. As shown in [32], variable subsampling blunts the specific effect of parameter choice at least to some extent. Furthermore, by combining variable subsampling with rotated bagging, one can incorporate subspace outlier detection principles to improve both accuracy and efficiency.

3. **Subspace histogram ensembles:** Many subspace histogram methods such as *RS-Hash* [476] (see section 5.2.5) and the isolation forest [367] provide fast and accurate density estimation in subspaces of the data. This can be helpful when many dimensions are locally irrelevant. The ensemble-centric nature of these methods make them robust and accurate. Although the isolation forest does not seem to be a density-estimator at first sight, its scores are intimately related to histogram-based density estimation and are qualitatively similar (see section 5.2.6.3 of Chapter 5). One drawback of such methods is that they tend to be biased against isolated points in the interior regions of the data set. For example, a single outlier at the center of normal data points on a unit sphere in 10 dimensions can receive an inlier-like score with the isolation forest [35]. Furthermore, all histogram-based methods can be challenged by small clusters of anomalies in the data.

In general, one should always wrap an ensemble method (like variable subsampling or rotated bagging) around a base algorithm, unless the algorithm is itself implemented inherently as an ensemble (like *RS-Hash* or isolation forests). As shown in [35], such ensembles have great *equalizing* power; in other words, they can improve base methods (with widely varying quality) to very similar performance that is near-optimal. Furthermore, one can combine the results from these types of robust ensembles to gain further advantages. As a first implementation, we recommend to use a simple ensemble average of the (standardized)

scores of algorithms from each of the three types discussed above, which are themselves implemented as ensembles. Such a combination is referred to as *TRINITY* in [35], which combines the nonlinear Mahalanobis method, the  $k$ -nearest neighbor detector, and the isolation forest. Each of these base algorithms is used in conjunction with variable subsampling. While it is recognized that more sophisticated algorithms might do better on specific types of data sets, the use of this simple approach can be a reasonably robust option when one does not have a deeper semantic understanding of the underlying data set. After all, lack of visibility and blindness to data characteristics are the hallmarks of all unsupervised settings.

## 13.11 Resources for the Practitioner

---

Since the problem of outlier analysis is one of the key problems in data mining, a significant number of software resources exist for this problem. The software available for this problem is both commercial and open-source. Note that the outlier analysis problem is addressed using both supervised and unsupervised methods. A significantly larger number of packages exist for the supervised version of the problem since it is directly related to the problem of classification, which is a much broader field. In this section, the key resources from these two perspectives will be summarized.

A significant number of open source packages exist in the literature for different variations of the problem. A meta-repository containing a description of the different resources for both unsupervised and supervised learning is the *KDD Nuggets Website* [626, 627]. The *Weka* repository [628] is a large general purpose repository, containing different kinds of data mining software for clustering, classification and outlier analysis. In addition, the *ELKI* repository [633] for outlier analysis contains an implementation of many of the advanced algorithms discussed in this book such as *LOF* and its variations, *LOCI*, EM-methods, distance-based methods, and subspace methods. Numerous methods for spatial outlier detection are contained in the same repository. A Python library for outlier detection may be found with *scikit learn* [632]. Packages for outlier detection in R can be found in the *RDataMining* library [634]. Resources for different components of supervised and unsupervised outlier analysis are available from the UCR time-series classification and clustering page [635], and the Symbolic Aggregate Approximation [636] page.

A significant amount of commercial software is also available for outlier analysis. An example is the *IBM Security Network Intrusion Prevention System* [637] for network intrusion detection. The *IBM SPSS Workbench* [638] has numerous tools that can be used for outlier detection in both temporal and non-temporal data. In particular, *IBM SPSS Statistics* [639] contains a significant number of tools which can be used to build models for outlier analysis. The *Oracle Data Miner* [642] has significant data mining capabilities including anomaly detection. *WizSoft* software [643] has designed a software *WizRule*, which can be used for fraud and anomaly detection. *SAS* [640] has developed many different software packages for general statistical modeling and anomaly detection. A particularly relevant one is the *SAS Security Intelligence* [641] software, which is designed to address fraud, compliance and security issues. All of the above can handle both supervised and unsupervised scenarios. Furthermore, for the supervised case, a significant amount of classification software is available both on an open source and commercial basis. An exhaustive list of the different kinds of open source and commercial software may be found in [627]. A number of anomaly detection implementations in Python are available on *scikit-learn* [632].

## 13.12 Conclusions and Summary

---

This chapter provides an overview of the applications of outlier analysis. These applications are distributed across a wide variety of domains; nevertheless, many of these domains map to similar formulations for modeling purposes. The goal of this chapter was to provide the practitioner an understanding of the methods used in different domains. The adaptation of various generic outlier analysis methods to specific domains was also discussed.

Outlier analysis brings numerous challenges with it in different application domains, because of the difficulty in distinguishing between noise and anomalies. Typically, the incorporation of human feedback, domain knowledge, ensemble methods, and explicit supervision can address many of these challenges. In cases in which significant visibility does not exist on the effectiveness of various algorithms, it is helpful to try an ensemble of some simple algorithms such as the Mahalanobis method, the  $k$ -nearest neighbor algorithm, and subspace histograms.

---

---

# Bibliography

---

---

- [1] N. Abe, B. Zadrozny, and J. Langford. Outlier Detection by Active Learning. *ACM KDD Conference*, 2006.
- [2] N. R. Adam, V. P. Janeja, and V. Atluri. Neighborhood-based Detection of Anomalies in High-Dimensional Spatio-temporal Sensor Datasets. *ACM SAC Conference*, 2004.
- [3] A. Adler, M. Elad, Y. Hel-Or, and E. Rivlin. Sparse Coding with Anomaly Detection. *Journal of Signal Processing Systems*, 79(2), pp. 179–188, 2015.
- [4] C. C. Aggarwal and P. S. Yu. Outlier Detection in High Dimensional Data. *ACM SIGMOD Conference*, 2001.
- [5] C. C. Aggarwal, C. Procopiuc, J. Wolf, P. Yu, and J. Park. Fast Algorithms for Projected Clustering. *ACM SIGMOD Conference*, 1999.
- [6] C. Aggarwal, J. Han, J. Wang, and P. Yu. A Framework for Projected Clustering of High Dimensional Data Streams. *VLDB Conference*, 2004.
- [7] C. C. Aggarwal and P. S. Yu. Finding Generalized Projected Clusters in High Dimensional Spaces. *ACM SIGMOD Conference*, 2000.
- [8] C. C. Aggarwal. Re-designing Distance Functions and Distance-based Applications for High Dimensional Data. *ACM SIGMOD Record*, 2001.
- [9] C. C. Aggarwal. On Abnormality Detection in Spuriously Populated Data Streams. *SIAM Conference on Data Mining*, 2005.
- [10] C. C. Aggarwal. *Data Streams: Models and Algorithms*, Springer, 2007.
- [11] C. C. Aggarwal. *Data Classification: Algorithms and Applications*, CRC Press, 2014.
- [12] C. C. Aggarwal. *Social Network Data Analytics*, Springer, 2011.
- [13] C. C. Aggarwal. On Effective Classification of Strings with Wavelets, *ACM KDD Conference*, 2002.
- [14] C. C. Aggarwal and K. Subbian. Evolutionary Network Analysis: A Survey. *ACM Computing Surveys*, May 2014.

- [15] C. C. Aggarwal, N. Ta, J. Wang, J. Feng, and M. J. Zaki. Xproj: A Framework for Projected Structural Clustering of XML Documents. *ACM KDD Conference*, 2007.
- [16] C. C. Aggarwal and P. S. Yu. On String Classification in Data Streams. *ACM KDD Conference*, 2007.
- [17] C. C. Aggarwal, Y. Zhao, and P. S. Yu. Outlier Detection in Graph Streams. *ICDE Conference*, 2011.
- [18] C. C. Aggarwal. Active Learning: A Survey, *Data Classification: Algorithms and Applications*, CRC Press, 2014.
- [19] C. C. Aggarwal. A Framework for Diagnosing Changes in Evolving Data Streams. *ACM SIGMOD Conference*, 2003.
- [20] C. C. Aggarwal and P. S. Yu. Online Analysis of Community Evolution in Data Streams. *SIAM Conference on Data Mining*, 2005.
- [21] C. C. Aggarwal. On the Effects of Dimensionality Reduction on High Dimensional Similarity Search. *ACM PODS Conference*, 2001.
- [22] C. C. Aggarwal. Managing and Mining Sensor Data, *Springer*, 2013.
- [23] C. C. Aggarwal and C. K. Reddy. Data Clustering: Algorithms and Applications, *CRC Press*, 2013.
- [24] C. Aggarwal and C. Zhai. Managing and Mining Text Data. *Springer*, 2012.
- [25] C. C. Aggarwal, A. Hinneburg, and D. Keim. On the Surprising Behavior of Distance Metrics in High Dimensional Space. *ICDT Conference*, 2001.
- [26] C. C. Aggarwal and P. S. Yu. Outlier Detection with Uncertain Data. *SIAM Conference on Data Mining*, 2008.
- [27] C. C. Aggarwal, Y. Xie, and P. S. Yu. On Dynamic Data-Driven Selection of Sensor Streams. *ACM KDD Conference*, 2011.
- [28] C. C. Aggarwal, J. Han, J. Wang, and P. Yu. A Framework for Clustering Evolving Data Streams. *VLDB Conference*, 2003.
- [29] C. C. Aggarwal and P. Yu. On Clustering Massive Text and Categorical Data Streams. *Knowledge and Information Systems*, 24(2), pp. 171–196, 2010.
- [30] C. C. Aggarwal and K. Subbian. Event Detection in Social Streams. *SIAM Conference on Data Mining*, 2012.
- [31] C. C. Aggarwal. Outlier Ensembles: Position Paper. *ACM SIGKDD Explorations*, 14(2), pp. 49–58, December, 2012.
- [32] C. C. Aggarwal and S. Sathe. Theoretical Foundations and Algorithms for Outlier Ensembles. *ACM SIGKDD Explorations*, 17(1), June 2015.
- [33] C. C. Aggarwal. Data Mining: The Textbook. *Springer*, 2015.
- [34] C. C. Aggarwal. Recommender Systems: The Textbook. *Springer*, 2016.



- [35] C. C. Aggarwal and S. Sathe. *Outlier Ensembles: An Introduction*. Springer, 2017.
- [36] R. Agrawal, T. Imielinski, and A. Swami. Mining Association Rules Between Sets of Items in Large Databases. *ACM SIGMOD Conference*, 1993.
- [37] R. Agrawal and R. Srikant. Fast algorithms for finding Association Rules in Large Databases. *VLDB Conference*, 1994.
- [38] M. Agyemang, K. Barker, and R. Alhajj. A Comprehensive Survey of Numeric and Symbolic Outlier Mining Techniques. *Intelligent Data Analysis*, 10(6). pp. 521–538, 2006.
- [39] A. Ahmad and L. Dey. A Method to Compute Distance between two Categorical Values of Same Attribute in Unsupervised Learning for Categorical Data Set. *Pattern Recognition Letters*, 28(1), pp. 110–118, 2007.
- [40] R. Ahuja, J. Orlin, and T. Magnanti. *Network Flows: Theory, Algorithms and Applications*. Prentice Hall, 1993.
- [41] L. Akoglu, M. McGlohon, and C. Faloutsos. Oddball: Spotting Anomalies in Weighted Graphs. *PAKDD Conference*, 2010.
- [42] L. Akoglu, H. Tong, J. Vreeken, and C. Faloutsos. Fast and Reliable Anomaly Detection in Categorical Data. *CIKM Conference*, 2012.
- [43] L. Akoglu, H. Tong, and D. Koutra. Graph-Based Anomaly Detection and Description: A Survey. *Data Mining and Knowledge Discovery*, 29(3), pp. 626–688, 2014.
- [44] L. Akoglu, E. Muller, and J. Vreeken. ACM KDD Workshop on Outlier Detection and Description, 2013. <http://www.outlier-analytics.org/odd13kdd/>
- [45] E. Aleskerov, B. Freisleben, and B. Rao. CARDWATCH: A Neural Network based Database Mining System for Credit Card Fraud Detection. *IEEE Computational Intelligence for Financial Engineering*, pp. 220–226, 1997.
- [46] T. Al-Khateeb, M. Masud, L. Khan, C. Aggarwal, J. Han, and B. Thuraisingham. Recurring and Novel Class Detection using Class-based Ensemble, *ICDM Conference*, 2012.
- [47] J. Allan, R. Papka, and V. Lavrenko. On-line New Event Detection and Tracking. *ACM SIGIR Conference*, 1998.
- [48] J. Allan, V. Lavrenko, and H. Jin. First Story Detection in TDT is Hard. *ACM CIKM Conference*, 2000.
- [49] J. Allan, J. Carbonell, G. Doddington, J. Yamron, and Y. Yang. Topic Detecting and Tracking Pilot Study Final Report, *CMU Technical Report, Paper 341*, 1998.
- [50] F. Alonso, J. Caraca-Valente, A. Gonzalez, and C. Montes. Combining Expert Knowledge and Data Mining in a Medical Diagnosis Domain. *Expert Systems with Applications*, 23(4), pp. 367–375, 2002.
- [51] Y. Altun, I. Tsochantaridis, and T. Hofmann. Hidden Narkov Support Vector Machines. *ICML Conference*, 2003.

- [52] M. Amer. Enhancing One-Class Support Vector Machines for Unsupervised Anomaly Detection. *Masters Thesis*, German University in Cairo, 2013. [http://www.madm.eu/\\_media/theses/ma\\_thesis\\_amer.pdf](http://www.madm.eu/_media/theses/ma_thesis_amer.pdf)
- [53] J. An and S. Cho. Variational Autoencoder based Anomaly Detection using Reconstruction Probability. *Technical Report*, SNU Data Mining Center, 2015. <http://dm.snu.ac.kr/static/docs/TR/SNUDM-TR-2015-03.pdf>
- [54] M. R. Anderberg. Cluster Analysis for Applications, *Academic Press*, New York, 1973.
- [55] D. Anderson, T. Lunt, H. Javitz, A. Tamaru, and A. Valdes. Detecting Unusual Program Behavior using the Statistical Components of NIDES, *Technical Report, SRI-CSL-95-06*, Computer Science Laboratory, SRI International, 1995.
- [56] D. Anderson, T. Frivold, A. Tamaru, and A. Valdes. Next-generation Intrusion Detection Expert System (nides), Software Users Manual, Beta-update Release. *Technical Report SRI-CSL-95-07*, Computer Science Laboratory, SRI International, 1994.
- [57] S. Ando. Clustering Needles in a Haystack: An Information Theoretic Analysis of Minority and Outlier Detection. *ICDM Conference*, 2007.
- [58] F. Angiulli and C. Pizzuti. Fast Outlier Detection in High Dimensional Spaces. *European Conference on Principles of Knowledge Discovery and Data Mining*, 2002.
- [59] F. Angiulli, F. Fassetti, and L. Palopoli. Finding Outlying Properties of Exceptional Objects. *ACM Transactions on Database Systems*, 34(1), 2009.
- [60] F. Angiulli and F. Fassetti. Detecting Distance-based Outliers in Streams of Data. *ACM CIKM Conference*, 2007.
- [61] M. Araujo, S. Papadimitriou, S. Gunnemann, C. Faloutsos, P. Basu, A. Swami, E. Papalexakis, and D. Koutra. Com2: Fast Automatic Discovery of Temporal (Comet) Communities. *PAKDD Conference*, 2014.
- [62] A. Arning, R. Agrawal, and P. Raghavan. A Linear Method for Deviation Detection in Large Databases. *ACM KDD Conference*, 1996.
- [63] I. Assent, P. Kranen, C. Beldauf, and T. Seidl. AnyOut: Anytime Outlier Detection in Streaming Data. *DASFAA Conference*, 2012.
- [64] I. Assent, R. Krieger, E. Muller, and T. Seidl. Subspace Outlier Mining in Large Multimedia Databases. *Parallel Universes and Local Patterns*, 2007.
- [65] A. Auer Jr. Correlation of Land Use and Cover with Meteorological Anomalies. *Journal of Applied Meteorology*, 17(5) pp. 636–643, 1978.
- [66] S. Babbar and S. Chawla. On Bayesian Network and Outlier Detection. *COMAD Conference*, 2010. Detailed Ph.D. thesis at <http://prijipati.library.usyd.edu.au/handle/2123/9371>
- [67] L. Backstrom, D. Huttenlocher, J. Kleinberg, and X. Lan. Group Formation in Large Social Networks: Membership, Growth, and Evolution. *KDD Conference*, 2006.
- [68] P. Baldi. Autoencoders, Unsupervised Learning, and Deep Architectures. *Unsupervised and Transfer Learning Challenges in Machine Learning*, Volume 7, 43, 2012.

- [69] D. Ballard. Generalizing the Hough Transform to Detect Arbitrary Shapes. *Pattern Recognition*, 11(2), pp. 111–122, 1981.
- [70] D. Barbara, Y. Li, J. Couto, J.-L. Lin, and S. Jajodia. Bootstrapping a Data Mining Intrusion Detection System. *Symposium on Applied Computing*, 2003.
- [71] D. Barbara, J. Couto, S. Jajodia, and N. Wu. ADAM: A Testbed for Exploring the Use of Data Mining in Intrusion Detection. *SIGMOD Record*, 30(4), pp. 15–24, 2001.
- [72] D. Barbara, J. Couto, S. Jajodia, and N. Wu. Detecting Novel Network Intrusions using Bayes Estimators. *SIAM Conference on Data Mining*, 2001.
- [73] R. Baragona and F. Battaglia. Outlier Detection in Multivariate Time Series by Independent Component Analysis. *Neural Computation*, 19(1), pp. 1962–1984, 2007.
- [74] V. Barnett and T. Lewis. *Outliers in Statistical Data*. Wiley, 1994.
- [75] S. Bay and M. Schwabacher. Mining Distance-based Outliers in Near Linear Time with Randomization and a Simple Pruning Rule. *ACM KDD Conference*, 2003.
- [76] S. Bay, K. Saito, N. Ueda, and P. Langley. A Framework for Discovering Anomalous Regimes in Multivariate Time-series Data with Local Models. *Technical report, Center for the Study of Language and Information*, Stanford University, 2004.
- [77] R. Beckman and R. Cook. Outliers, *Technometrics*, 25(2), pp. 119–149, 1983.
- [78] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R\*-tree: An Efficient and Robust Access method for Points and Rectangles. *ACM SIGMOD Conference*, 1990.
- [79] Y. Bengio, J. Paiement, P. Vincent, O. Delalleau, N. Le Roux, and M. Ouimet. Out-of-sample Extensions for LLE, Isomap, MDS, Eigenmaps, and Spectral Clustering. *Neural Information Processing Systems*, 16, pp. 177–184, 2006.
- [80] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy Layer-wise Training of Deep Networks. *Neural Information Processing Systems*, 19., pp. 153–160, 2007.
- [81] M. Berlingerio, F. Bonchi, B. Bringmann, and A. Gionis. Mining Graph Evolution Rules. *ECML/PKDD Conference*, 2009.
- [82] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is “nearest neighbor” meaningful? *International Conference on Database Theory*, 1999.
- [83] K. Bhaduri, B. Matthews, and C. Giannella. Algorithms for Speeding up Distance-based Outlier Detection. *ACM KDD Conference*, 2011.
- [84] C. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.
- [85] E. Blanzieri and A. Bryl. A Survey of Learning-based Techniques of Email Spam Filtering. *Artificial Intelligence Review*, 29(1), pp. 63–92, 2008.
- [86] M. Bosc, F. Heitz, J.-P. Armspach, I. Namer, D. Gounot, and L. Rumbach. Automatic Change Detection in Multimodal Serial MRI: Application to Multiple Sclerosis Lesion Evolution. *NeuroImage*, 20(2), 2003, Pages 643–656

- [87] Y. Bilberman. A Context Similarity Measure. *ECML Conference*, 1994.
- [88] P. Billingsley. Probability and Measure, Second Edition. *Wiley*, 1986.
- [89] D. Birant and A. Kut. Detecting Spatio-temporal Outliers in Large Databases. *Journal of Computing and Information Technology*, 14(4), pp. 291–297, 2006.
- [90] D. Blei and J. Lafferty. Dynamic Topic Models. *ICML Conference*, 2006.
- [91] D. Blei, A. Ng, and M. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3: pp. 993–1022, 2003.
- [92] C. Bohm, K. Haegler, N. Muller, and C. Plant. Coco: Coding Cost for Parameter Free Outlier Detection. *ACM KDD Conference*, 2009.
- [93] S. Boriah, V. Chandola, and V. Kumar. Similarity Measures for Categorical Data: A Comparative Evaluation. *SIAM Conference on Data Mining*, 2008.
- [94] J. Branch, B. Szymanski, C. Giannella, R. Wolff, and H. Kargupta. In-Network Outlier Detection in Wireless Sensor Networks. *ICDCS Conference*, 2006.
- [95] T. Brants, F. Chen, and A. Farahat. A System for New Event Detection. *ACM SIGIR Conference*, 2003.
- [96] M. Breunig, H.-P. Kriegel, R. Ng, and J. Sander. LOF: Identifying Density-based Local Outliers. *ACM SIGMOD Conference*, 2000.
- [97] M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. OPTICS-OF: identifying Local Outliers. *PKDD Conference*, 1999.
- [98] L. Brieman. Bagging Predictors. *Machine Learning*, 24: pp. 123–140, 1996.
- [99] L. Brieman and A. Cutler. Random Forests Manual v4.0, *Technical Report, UC Berkeley*, 2003. Available online at: [https://www.stat.berkeley.edu/~breiman/Using\\_random\\_forests\\_v4.0.pdf](https://www.stat.berkeley.edu/~breiman/Using_random_forests_v4.0.pdf)
- [100] M. R. Brito, E. L. Chavez, A. J. Quiroz, and J. E. Yukich. Connectivity of the Mutual  $k$ -Nearest Neighbor Graph in Clustering and Outlier Detection. *Statistics and Probability Letters*, 35(1), pp. 33–42, 1997.
- [101] R. G. Brown and P. Hwang. Introduction to Random Signals and Applied Kalman Filtering. *John Wiley and Sons*, 1997.
- [102] Y. Bu, L. Chen, A. W.-C. Fu, and D. Liu. Efficient Anomaly Monitoring over Moving Object Trajectory Streams. *ACM KDD Conference*, 2009.
- [103] S. Budalakoti, A. Srivastava, R. Akella, and E. Turkov. Anomaly Detection in Large Sets of High-dimensional Symbol Sequences. *NASA Ames Research Center, Technical Report NASA TM-2006-214553*, 2006.
- [104] S. Budalakoti, A. Srivastava, and M. Otey. Anomaly Detection and Diagnosis Algorithms for Discrete Symbol Sequences with Applications to Airline Safety. *IEEE International Conference on Systems, Man, and Cybernetics*, 37(6), 2007.
- [105] P. Buhlmann. Bagging, Subbagging and Bragging for Improving some Prediction Algorithms. *Recent advances and trends in nonparametric statistics*, Elsevier, 2003.

- [106] P. Buhlmann and B. Yu. Analyzing Bagging. *Annals of Statistics*, pp. 927–961, 2002.
- [107] A. Buja and W. Stuetzle. Observations on Bagging. *Statistica Sinica*, 16(2), 323, 2006.
- [108] S. Burdakis and A. Deligiannakis. Detecting Outliers in Sensor Networks using the Geometric Approach. *ICDE Conference*, 2012.
- [109] T. Burnaby. On a Method for Character Weighting a Similarity Coefficient: Employing the Concept of Information. *Mathematical Geology*, 2(1), 25–38, 1970.
- [110] S. Byers and A. Raftery. Nearest-Neighbor Clutter Removal for Estimating Features in Spatial Point Processes. *JASIS*, 93, pp. 577–584, June 1998.
- [111] I. Cadez, D. Heckerman, C. Meek, P. Smyth, and S. White. Visualization of Navigation Patterns on a Web Site using Model-based Clustering. *ACM SIGMOD Conference*, 2000.
- [112] P. H. Calamai. Projected Gradient Methods for Linearly Constrained Problems. *Mathematical Programming*, 39: pp, 93–116, 1987.
- [113] C. Campbell and K. P. Bennett. A Linear-Programming Approach to Novel Class Detection. *Neural Information Processing Systems*, 2000.
- [114] G. O. Campos, A. Zimek, J. Sander, R. J. G. B. Campello, B. Micenkova, E. Schubert, I. Assent, and M. E. Houle. On the Evaluation of Unsupervised Outlier Detection: Measures, Datasets, and an Empirical Study. *DMKD Journal*, 30(4), pp. 891–927, 2016.
- [115] M. J. Canty. Image Analysis, Classification and Change Detection in Remote Sensing: with Algorithms for ENVI/IDL. *CRC Press*, 2006.
- [116] C. Caroni. Outlier Detection by Robust Principal Component Analysis. *Communications in Statistics – Simulation and Computation*, 29: pp. 129–151, 2000.
- [117] L. E. Carr and R. L. Elsberry. Monsoonal Interactions Leading to Sudden Tropical Cyclone Track Changes. *Monthly Weather Review*, 123(2), pp. 265–290, Feb. 1995.
- [118] P. Castro, D. Zhang, C. Chen, S. Li, and G. Pan. From Taxi GPS Traces to Social and Community Dynamics: A Survey. *ACM Computing Surveys (CSUR)*, 46(2), 17, 2013.
- [119] D. Chakrabarti and C. Faloutsos. Evolutionary Clustering. *ACM KDD Conference*, 2006.
- [120] D. Chakrabarti. AutoPart: Parameter-Free Graph Partitioning and Outlier Detection. *PKDD Conference*, 2004.
- [121] K. Chakrabarti and S. Mehrotra. Local Dimensionality Reduction: A New Approach to Indexing High Dimensional Spaces. *VLDB Conference Proceedings*, 2000.
- [122] S. Chakrabarti, S. Sarawagi, and B. Dom. Mining Surprising Patterns using Temporal Description Length. *VLDB Conference*, 1998.
- [123] C.-H. Chan and G. Pang. Fabric Defect Detection by Fourier Analysis. *IEEE Transactions on Industry Applications*, 36(5), 2000.

- [124] P. K. Chan and S. J. Stolfo. Toward Scalable Learning with Non-uniform Class and Cost Distributions: A Case Study in Credit Card Fraud Detection. *KDD Conference*, pp. 164–168, 1998.
- [125] V. Chandola, A. Banerjee, and V. Kumar. Anomaly Detection: A Survey. *ACM Computing Surveys*, 41(3), 2009.
- [126] V. Chandola, A. Banerjee, and V. Kumar. Anomaly Detection for Discrete Sequences: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, 24(5): pp. 823–839, 2012.
- [127] V. Chandola, V. Mithal, and V. Kumar. A Comparative Evaluation of Anomaly Detection Techniques for Sequence Data. *ICDM Conference*, 2008.
- [128] C. Chang and C. J. Lin. LIBSVM: a Library for Support Vector Machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3), 27, 2011.
- [129] I. Chang, G. C. Tiao, and C. Chen. Estimation of Time Series Parameters in the Presence of Outliers. *Technometrics*, 30(2), pp. 193–204, 1988.
- [130] O. Chapelle. Training a Support Vector Machine in the Primal. *Neural Computation*, 19(5), pp. 1155–1178, 2007.
- [131] A. Chaudhary, A. S. Szalay, and A. W. Moore. Very Fast Outlier Detection in Large Multidimensional Data Sets. *DMKD Workshop*, 2002.
- [132] N. V. Chawla, N. Japkowicz, and A. Kotcz. Editorial: Special Issue on Learning from Imbalanced Data Sets. *ACM SIGKDD Explorations Newsletter*, 6(1), 1–6, 2004.
- [133] N. V. Chawla, K. W. Bower, L. O. Hall, and W. P. Kegelmeyer. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research (JAIR)*, 16, pp. 321–356, 2002.
- [134] N. V. Chawla, A. Lazarevic, L. Hall, and K. Bowyer. SMOTEBoost: Improving Prediction of the Minority Class in Boosting. *PKDD*, pp. 107–119, 2003.
- [135] N. V. Chawla, D. A. Cieslak, L. O. Hall, and A. Joshi. Automatically Counteracting Imbalance and its Empirical Relationship to Cost. *Data Mining and Knowledge Discovery*, 17(2), pp. 225–252, 2008.
- [136] C. Chen and L.-M. Liu. Joint Estimation of Model Parameters and Outlier Effects in Time Series. *Journal of the American Statistical Association*, 88(421), pp. 284–297, March 1993.
- [137] C. Chen, D. Zhang, P. Castro, N. Li, L. Sun, and S. Li. Real-time Detection of Anomalous Taxi Trajectories from GPS Traces. *Mobile and Ubiquitous Systems: Computing, Networking, and Services*, pp. 63–74, 2011.
- [138] D. Chen, C.-T. Lu, Y. Chen, and D. Kou. On Detecting Spatial Outliers. *Geoinformatica*, 12: pp. 455–475, 2008.
- [139] Y. Chen and L. Tu. Density-based Clustering for Real Time Stream Data. *ACM KDD Conference*, 2007.

- [140] H. Cheng, P.-N. Tan, C. Potter, and S. Klooster. Detection and Characterization of Anomalies in Multivariate Time Series. *SIAM Conference on Data Mining*, 2009.
- [141] T. Cheng and Z. Li. A Hybrid Approach to Detect Spatialtemporal Outliers. *International Conference on Geoinformatics*, 2004.
- [142] T. Cheng and Z. Li. A Multiscale Approach for Spatio-temporal Outlier Detection. *Transactions in GIS*, 10(2), pp. 253–263, March 2006.
- [143] Y. Chi, X. Song, D. Zhou, K. Hino, and B. Tseng. Evolutionary Spectral Clustering by Incorporating Temporal Smoothness. *ACM KDD Conference*, 2007.
- [144] A. Chiu and A. Fu. Enhancements on Local Outlier Detection. *Database Engineering and Applications Symposium*, 2003.
- [145] C. Chow and D. Yeung. Parzen-Window Network Intrusion Detectors. *International Conference on Pattern Recognition*, 4, 2002.
- [146] M. Chow, R. Sharpe, and J. Hung. On the Application and Design of Artificial Neural Networks for Motor Fault Detection. *IEEE Transactions on Industrial Electronics*, 40(2), 1993.
- [147] V. Ciesielski and V. Ha. Texture Detection using Neural Networks Trained on Examples of One Class. *LNCS*, Vol. 5866, pp. 140–149, *Springer*, 2009.
- [148] W. Cohen. Fast Effective Rule Induction. *ICML Conference*, 1995.
- [149] D. Cohn, R. Atlas, and N. Ladner. Improving Generalization with Active Learning. *Machine Learning*, 15, pp. 201–221.
- [150] R. Cooley, B. Mobashar, and J. Srivastava. Data Preparation for Mining World Wide Web Browsing Patterns. *Knowledge and Information Systems*, 1, pp. 5–32, 1999.
- [151] D. Cook and L. Holder. Graph-Based Data Mining. *IEEE Intelligent Systems*, 15(2), pp. 32–41, 2000.
- [152] G. Cormack. Email Spam Filtering: A Systematic Review. *Foundations and Trends in Information Retrieval*, 1(4), pp. 335–455, 2007.
- [153] X. Dang, B. Micenkova, I. Assent, and R. Ng. Outlier Detection with Space Transformation and Spectral Analysis. *SIAM Conference on Data Mining*, pp. 225–233, 2013.
- [154] G. Das and H. Mannila. Context-based Similarity Measures for Categorical Databases. *PKDD Conference*, 2000.
- [155] K. Das, J. Schneider, and D. Neill. Anomaly Pattern Detection in Categorical Data Sets. *ACM KDD Conference*, 2008.
- [156] S. Das, B. Matthews, A. Srivastava, and N. Oza. Multiple Kernel Learning for Heterogeneous Anomaly Detection: Algorithm and Aviation Safety Case Study. *ACM KDD Conference*, 2010.
- [157] D. Dasgupta and S. Forrest. Novelty Detection in Time Series using Ideas from Immunology. *International Conference on Intelligent Systems*, 1996.

- [158] D. Dasgupta and F. Nino. A Comparison of Negative and Positive Selection Algorithms in Novel Pattern Detection. *IEEE Conference on Systems, Man, and Cybernetics*, 1, pp. 125–130, 2000.
- [159] T. Dasu, S. Krishnan, S. Venkatasubramanian, and K. Yi. An Information-Theoretic Approach to Detecting Change in Multi-dimensional Data Streams. *Symposium on the Interface of Computer Science, Statistics, and Applications*, 2006.
- [160] H. Dau, V. Ciesielski, and A. Song. Anomaly Detection using Replicator Neural Networks Trained on Examples of One Class. In *Simulated Evolution and Learning*, pp. 311–322, Springer, 2014.
- [161] N. Delannay, C. Archambeau, and M. Verleysen. Improving the Robustness to Outliers of Mixtures of Probabilistic PCAs. *PAKDD Conference*, 2008.
- [162] S. T. Deerwester, S. T. Dumais, G. Furnas, and R. Harshman. Indexing by Latent Semantic Analysis, *JASIS*, 1990.
- [163] K. A. De Jong. Analysis of the Behavior of a Class of Genetic Adaptive Systems. *Ph.D. Dissertation, University of Michigan*, Ann Arbor, MI, 1975.
- [164] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society*, B, vol. 39(1), pp. 1–38, 1977.
- [165] D. Denning. An Intrusion Detection Model. *IEEE Transactions of Software Engineering*, 13(2), pp. 222–232.
- [166] R. Derrig. Insurance Fraud. *Journal of Risk and Insurance*, 69(3), pp. 271–287, 2002.
- [167] M. Desforges, P. Jacob, and J. Cooper. Applications of Probability Density Estimation to the Detection of Abnormal Conditions in Engineering. *Proceedings of Institute of Mechanical Engineers*, Vol. 212, pp. 687–703, 1998.
- [168] M. Deshpande and G. Karypis. Evaluation of Techniques for Classifying Biological Sequences. *PAKDD Conference*, 2002.
- [169] M. Deshpande and G. Karypis. Selective Markov Models for Predicting Web Page Accesses. *ACM Transactions on Internet Technology*, 4(2), pp. 163–184, 2004.
- [170] T. Dietterich. Ensemble Methods in Machine Learning. *First International Workshop on Multiple Classifier Systems*, 2000.
- [171] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh. Querying and Mining of Time Series Data: Experimental Comparison of Representations and Distance Measures. *PVLDB*, 1(2), pp. 1542–1552, 2008.
- [172] S. Donoho. Early Detection of Insider Trading in Option Markets. *ACM KDD Conference*, 2004.
- [173] C. Drummond and R. Holte. C4.5, Class Imbalance, and Cost Sensitivity: Why Undersampling beats Oversampling. *ICML Workshop on Learning from Imbalanced Data Sets*, 2003.



- [174] C. Drummond and R. Holte. Explicitly Representing Expected Cost: An Alternative to ROC representation. *ACM KDD Conference*, pp. 198–207, 2001.
- [175] P. Domingos. MetaCost: A General Framework for Making Classifiers Cost-Sensitive, *ACM KDD Conference*, 1999.
- [176] R. Duda, P. Hart, and D. Stork. *Pattern Classification*. Wiley, 2001.
- [177] H. Dutta, C. Giannella, K. Borne, and H. Kargupta. Distributed top- $k$  Outlier Detection in Astronomy Catalogs using the Demac System. *SIAM Conference on Data Mining*, 2007.
- [178] J. Dutta, B. Banerjee, and C. Reddy. RODS: Rarity based Outlier Detection in a Sparse Coding Framework. *IEEE Transactions on Knowledge and Data Engineering*, 28(2), p. 483–495, 2016.
- [179] W. Eberle and L. B. Holder. Mining for Structural Anomalies in Graph-based Data. *DMIN*, 2007.
- [180] F. Y. Edgeworth. On Discordant Observations. *Philosophical Magazine*, 23(5), pp. 364–375, 1887.
- [181] M. Elahi, K. Li, W. Nisar, X. Lv, and H. Wang. Efficient Clustering-Based Outlier Detection Algorithm for Dynamic Data Stream. *FSKD Conference*, 2008.
- [182] C. Elkan. The Foundations of Cost-Sensitive Learning. *IJCAI*, 2001.
- [183] C. Elkan and K. Noto. Learning Classifiers from only Positive and Unlabeled Data. *ACM KDD Conference*, 2008.
- [184] A. Emmott, S. Das, T. Dietterich, A. Fern, and W. Wong. Systematic Construction of Anomaly Detection Benchmarks from Real Data. *arXiv:1503.01158*, 2015.  
<https://arxiv.org/abs/1503.01158>
- [185] D. Endler. Intrusion detection: Applying Machine Learning to Solaris Audit Data. *Annual Computer Security Applications Conference*, 1998.
- [186] D. Erhan, Y. Bengio, A. Courville, P. Manzagol, P. Vincent, and S. Bengio. Why does Unsupervised Pre-training help Deep Learning? *Journal of Machine Learning Research*, 11, 625-660, 2010.
- [187] E. Eskin. Anomaly Detection over Noisy Data using Learned Probability Distributions. *ICML Conference*, 2000.
- [188] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo. A Geometric Framework for Unsupervised Anomaly Detection. *Applications of Data Mining in Computer Security*. Kluwer, 2002.
- [189] E. Eskin, W. Lee, and S. Stolfo, Modeling System Call for Intrusion Detection using Dynamic Window Sizes. *DISCEX*, 2001.
- [190] H. Fan, O. Zaiane, A. Foss, and J. Wu. A Nonparametric Outlier Detection for Efficiently Discovering top- $n$  Outliers from Engineering Data. *PAKDD Conference*, 2006.

- [191] W. Fan, S. Stolfo, J. Zhang, and P. Chan. AdaCost: Misclassification Cost Sensitive Boosting. *ICML Conference*, 1999.
- [192] T. Fawcett. ROC Graphs: Notes and Practical Considerations for Researchers. *Technical Report HPL-2003-4*, Palo Alto, CA: HP Laboratories, 2003.
- [193] T. Fawcett and F. Provost. Activity Monitoring: Noticing Interesting Changes in Behavior. *ACM KDD Conference*, 1999.
- [194] W. Feng, C. Zhang, W. Zhang, J. Han, J. Wang, C. Aggarwal, and J. Huang. STREAMCUBE: Hierarchical Spatio-temporal Hashtag Clustering for Event Exploration over the Twitter Stream. *IEEE ICDE Conference*, pp. 1561–1572, 2015.
- [195] M. Fernandez-Delgado, E. Cernadas, S. Barro, and D. Amorim. Do we Need Hundreds of Classifiers to Solve Real World Classification Problems? *The Journal of Machine Learning Research*, 15(1), pp. 3133–3181, 2014.
- [196] D. Fetterly, M. Manasse, and M. Najork. Spam, Damn Spam, and Statistics: using Statistical Analysis to Locate Spam Web Pages. *WebDB*, 2004.
- [197] S. Forrest, C. Warrender, and B. Pearlmutter. Detecting Intrusions using System Calls: Alternate Data Models. *IEEE ISRSP*, 1999.
- [198] S. Forrest, S. Hofmeyr, A. Somayaji, and T. A. Longstaff. A Sense of Self for Unix Processes. *ISRSP*, 1996.
- [199] S. Forrest, P. D’Haeseleer, and P. Helman. An Immunological Approach to Change Detection: Algorithms, Analysis and Implications. *IEEE Symposium on Security and Privacy*, 1996.
- [200] S. Forrest, F. Esponda, and P. Helman. A Formal Framework for Positive and Negative Detection Schemes. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, pp. 357–373, 2004.
- [201] S. Forrest, A. Perelson, L. Allen, and R. Cherukuri. Self-Nonself Discrimination in a Computer. *IEEE Symposium on Security and Privacy*, 1994.
- [202] A. Fox. Outliers in Time Series. *Journal of the Royal Statistical Society. Series B (Methodological)*, 34(3), pp. 350–363, 1972.
- [203] A. Frank and A. Asuncion. UCI Machine Learning Repository, Irvine, CA: University of California, School of Information and Computer Science, 2010. <http://archive.ics.uci.edu/ml>
- [204] C. Franke and M. Gertz. ORDEN: Outlier Region Detection and Exploration in Sensor Networks. *ACM SIGMOD Conference*, 2009.
- [205] A. Fu, O. Leung, E. Keogh, and J. Lin. Finding Time Series Discords based on Haar Transform. *Advanced Data Mining and Applications*, 2006.
- [206] R. Fujumaki, T. Yairi, and K. Machida. An Approach to Spacecraft Anomaly Detection Problem using Kernel Feature Space. *ACM KDD Conference*, 2005.
- [207] R. Fujamaki. Anomaly Detection Support Vector Machine and Its Application to Fault Diagnosis. *ICDM Conference*, 2008.

- [208] P. Galeano, D. Pea, and R. S. Tsay. Outlier detection in Multivariate Time Series via Projection Pursuit. *Statistics and Econometrics Working Papers WS044221*, Universidad Carlos III, 2004.
- [209] P. Gambaryan. A Mathematical Model of Taxonomy. *Izvest. Akad. Nauk Armen, SSR*, 17(12), pp. 47–53, 1964.
- [210] V. Ganti, J. Gehrke, and R. Ramakrishnan. CACTUS: Clustering Categorical Data using Summaries. *ACM KDD Conference*, 1999.
- [211] B. Gao, H.-Y. Ma, and Y.-H. Yang. HMMs (Hidden Markov Models) based on Anomaly Intrusion Detection Method. *International Conference on Machine Learning and Cybernetics*, 2002.
- [212] H. Gao, X. Wang, J. Tang, and H. Liu. Network Denoising in Social Media. *Technical Report, Arizona State University*, 2011.
- [213] J. Gao and P.-N. Tan. Converting Outlier Scores from Outlier Detection Algorithms into Probability Estimates. *ICDM Conference*, 2006.
- [214] J. Gao, F. Liang, W. Fan, C. Wang, Y. Sun, and J. Han. On Community Outliers and their Efficient Detection in Information Networks. *ACM KDD Conference*, pp. 813–822, 2010.
- [215] Y. Ge, H. Xiong, Z.-H. Zhou, H. Ozdemir, J. Yu, and K. Lee. Top-Eye: Top- $k$  Evolving Trajectory Outlier Detection. *CIKM Conference*, 2010.
- [216] A. Ghosh, J. Wanken, and F. Charron. Detecting Anomalous and Unknown Intrusions against Programs. *Annual Computer Security Applications Conference*, 1998.
- [217] A. Ghosh, A. Schwartzbard, and M. Schatz. Learning Program Behavior Profiles for Intrusion Detection. *USENIX Workshop on Intrusion Detection and Network Monitoring*, pp. 51–62, 1999.
- [218] S. Ghosh and D. Reilly. Credit Card Fraud Detection with a Neural Network. *International Conference on System Sciences: Information Systems: Decision Support and Knowledge-Based Systems*, 3, pp. 621–630, 1994.
- [219] A. Ghoting, S. Parthasarathy, and M. Otey. Fast Mining of Distance-based Outliers in High Dimensional Spaces. *SIAM Conference on Data Mining*, 2006.
- [220] D. Gibson, J. Kleinberg, and P. Raghavan. Clustering Categorical Data: An Approach Based on Dynamical Systems. *The VLDB Journal*, 8(3), pp. 222–236, 2000.
- [221] M. Goldstein and S. Uchida. A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data. *PLoS One*, 11(4), e0152173, 2016.
- [222] D. W. Goodall. A new similarity index based on probability. *Biometrics*, 22(4), pp. 882–907, 1966.
- [223] I. Goodfellow, Y. Bengio, and A. Courville. Deep Learning. *MIT Press*, 2016. <http://goodfeli.github.io/dlbook/>,
- [224] N. Gornitz, M. Kloft, K. Rieck, and U. Brefeld. Toward Supervised Anomaly Detection. *Journal of Artificial Intelligence Research*, 2013.

- [225] F. Grubbs. Procedures for Detecting Outlying Observations in Samples. *Technometrics*, 11(1), pp. 1–21, 1969.
- [226] S. Guha, R. Rastogi, and K. Shim. ROCK: A Robust Clustering Algorithm for Categorical Attributes. *Information Systems*, 25(5), pp. 345–366, 2000.
- [227] S. Guha, N. Mishra, G. Roy, and O. Schrijver. Robust Random Cut Forest Based Anomaly Detection On Streams. *ICML Conference*, pp. 2712–2721, 2016.
- [228] S. Gunnemann, N. Gunnemann, and C. Faloutsos. Detecting Anomalies in Dynamic Rating Data: A Robust Probabilistic Model for Rating Evolution. *ACM KDD Conference*, 2014.
- [229] D. Gunopulos and G. Das. Time-series Similarity Measures, and Time Series Indexing, *ACM SIGMOD Conference*, 2001.
- [230] S. Gunter, N. N. Schraudolph, and S. V. N. Vishwanathan. Fast Iterative Kernel Principal Component Analysis. *Journal of Machine Learning Research*, 8, pp 1893–1918, 2007.
- [231] M. Gupta, J. Gao, C. Aggarwal, and J. Han. Outlier Detection for Temporal Data. *Morgan and Claypool*, 2014.
- [232] M. Gupta, J. Gao, C. Aggarwal, and J. Han. Outlier Detection for Temporal Data: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, 26(9), pp. 1, 2014.
- [233] M. Gupta, C. Aggarwal, J. Han, and Y. Sun. Evolutionary Clustering and Analysis of Bibliographic Networks. *ASONAM Conference*, 2011.
- [234] M. Gupta, C. Aggarwal, and J. Han. Finding Top- $k$  Shortest Path Distance Changes in an Evolutionary Network. *SSDBM Conference*, 2011.
- [235] M. Gupta, J. Gao, Y. Sun, and J. Han. Community Trend Outlier Detection Using Soft Temporal Pattern Mining. *ECML/PKDD Conference*, 2012.
- [236] M. Gupta, J. Gao, Y. Sun, and J. Han. Integrating Community Matching and Outlier Detection for Mining Evolutionary Community Outliers. *KDD Conference*, 2012.
- [237] M. Gupta, A. Mallya, S. Roy, J. Cho, and J. Han. Local Learning for Mining Outlier Subgraphs from Network Datasets. *SIAM Conference on Data Mining*, pp. 73–81, 2014.
- [238] M. Gupta, J. Gao, X. Yan, H. Cam, and J. Han. Top- $k$  Interesting Subgraph Discovery in Information Networks. *International Conference on Data Engineering (ICDE)*, 2014.
- [239] D. Gusfield. Algorithms for Strings, Trees and Sequences. *Cambridge University Press*, 1997.
- [240] D. Guthrie, L. Guthrie, and Y. Wilks. An Unsupervised Approach for the Detection of Outliers in Corpora. *LREC*, 2008.
- [241] S. Guttormsson, R. Marks, M. El-Sharkawi, and I. Kerszenbaum. Elliptical Novelty Grouping for Online Short-turn Detection of Excited Running Rotors. *IEEE Transactions on Energy Conversion*, 14(1), pp. 16–22, 1999.

- [242] R. Gwadera, M. Atallah, and W. Szpankowski. Markov Models for Identification of Significant Episodes. *SIAM Conference on Data Mining*, 2005.
- [243] R. Gwadera, M. Atallah, and W. Szpankowski. Detection of Significant Sets of Episodes in Event Sequences. *IEEE ICDM Conference*, 2004.
- [244] R. Gwadera, M. Atallah, and W. Szpankowski. Reliable Detection of Episodes in Event Sequences. *Knowledge and Information Systems*, 7(4), pp. 415–437, 2005.
- [245] F. Hampel. A General Qualitative Definition of Robustness. *Annals of Mathematics and Statistics*, 43, pp. 1887–1896, 1971.
- [246] J. Hanley and B. McNeil. The Meaning and Use of the Area under a Receiver Operating Characteristic (ROC) Curve. *Radiology*, 143, pp. 29–36, 1982.
- [247] J. Haslett, R. Brandley, P. Craig, A. Unwin, and G. Wills. Dynamic Graphics for Exploring Spatial Data With Application to Locating Global and Local Anomalies. *The American Statistician*, 45: pp. 234–242, 1991.
- [248] V. Hautamaki, I. Karkkainen, and P. Franti. Outlier Detection Using  $k$ -Nearest Neighbor Graph. *International Conference on Pattern Recognition*, 2004.
- [249] D. Hawkins. Identification of Outliers, *Chapman and Hall*, 1980.
- [250] S. Hawkins, H. He, G. Williams, and R. Baxter. Outlier Detection using Replicator Neural Networks. *International Conference on Data Warehousing and Knowledge Discovery*, pp. 170–180, Springer, 2002.
- [251] G. G. Hazel. Multivariate Gaussian MRF for Multispectral Scene Segmentation and Anomaly Detection. *GeoRS*, 38(3), pp. 1199–1211, 2000.
- [252] J. He, and J. Carbonell. Nearest-Neighbor-Based Active Learning for Rare Category Detection. *CMU Computer Science Department*, Paper 281, 2007.  
<http://repository.cmu.edu/compsci/281>
- [253] Z. He, S. Deng, and X. Xu. Outlier Detection Integrating Semantic Knowledge. *Web Age Information Management (WAIM)*, 2002.
- [254] Z. He, X. Xu, J. Huang, and S. Deng. FP-Outlier: Frequent Pattern-based Outlier Detection. *COMSIS*, 2(1), 2005.
- [255] Z. He, X. Xu, and S. Deng. Discovering Cluster-based Local Outliers. *Pattern Recognition Letters*, Vol 24(9–10), pp. 1641–1650, 2003.
- [256] Z. He, X. Xu, and S. Deng. An Optimization Model for Outlier Detection in Categorical Data. *International Conference on Intelligent Computing*, 2005.
- [257] Z. He, S. Deng, X. Xu, and J. Huang. A Fast Greedy Algorithm for Outlier Mining. *PAKDD Conference*, 2006.
- [258] Z. He, S. Deng, and X. Xu. A Unified Subspace Outlier Ensemble Framework for Outlier Detection. *Advances in Web-Age Information Management*, pp. 632–637, 2005.
- [259] R. Hecht-Nielsen. Replicator Neural Networks for Universal Optimal Source Coding. *Science*, 269, 1860–1863, 1995.

- [260] P. Helman and J. Bhangoo. A Statistically-based System for Prioritizing Information Exploration under Uncertainty. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 27(4), pp. 449–466, 1997.
- [261] M. Henrion, D. Hand, A. Gandy, and D. Mortlock. CASOS: A Subspace Method for Anomaly Detection in High Dimensional Astronomical Databases. *Statistical Analysis and Data Mining*, 6(1), pp. 53–72, 2013.
- [262] S. Hido, Y. Tsuboi, H. Kashima, M. Sugiyama, and T. Kanamori. Statistical Outlier Detection using Direct Density Ratio Estimation. *Knowledge and Information Systems*, 26(2), pp. 309–336, 2011.
- [263] A. Hinneburg, C. Aggarwal, and D. Keim. What is the Nearest Neighbor in High-dimensional Spaces?, *VLDB Conference*, 2000.
- [264] G. Hinton and R. Salakhutdinov. Reducing the Dimensionality of Data with Neural Networks. *Science*, 313, (5786), pp. 504–507, 2006.
- [265] H. O. Hirschfeld. A Connection between Correlation and Contingency. *Proc. Cambridge Philosophical Society*, 31, pp. 520–524, 1935.
- [266] T. K. Ho. Random Decision Forests. *Third International Conference on Document Analysis and Recognition*, 1995. Extended version appears in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8), pp. 832–844, 1998.
- [267] S.-S. Ho. A Martingale Framework for Concept Change Detection in Time-Varying Data Streams. *ICML Conference*, 2005.
- [268] V. Hodge and J. Austin. A Survey of Outlier Detection Methodologies, *Artificial Intelligence Review*, 22(2), pp. 85–126, 2004.
- [269] J. Hodges. Efficiency in Normal Samples and Tolerance of Extreme Values for Some Estimates of Location. *Fifth Berkeley Symposium on Mathematics, Statistics and Probability*, 1, pp. 163–168, 1967.
- [270] H. Hoffmann. Kernel PCA for Novelty Detection. *Pattern Recognition*, 40(3), pp. 863–874, 2007.
- [271] T. Hofmann. Probabilistic Latent Semantic Indexing. *ACM SIGIR Conference*, 1999.
- [272] S. Hofmeyr, S. Forrest, and A. Somayaji. Intrusion Detection using Sequences of System Calls. *Journal of Computer Security*, 6(3), pp. 151–180, 1998.
- [273] J. H. Holland. Adaptation in Natural and Artificial Systems. *University of Michigan Press*, Ann Arbor, MI, 1975.
- [274] G. Hollier and J. Austin. Novelty Detection for Strain-gauge Degradation using Maximally Correlated Components. *European Symposium on Artificial Neural Networks*, 2002.
- [275] J. Hollmen and V. Tresp. Call-based Fraud Detection in Mobile Communication Networks using a Hierarchical Regime-switching Model. *Neural Information Processing Systems*, pp. 889–895, 1998.

- [276] P. Horn, L. Feng, Y. Li, and A. J. Pesce. Effect of Outliers and Nonhealthy Individuals on Reference Interval Estimation. *Clinical Chemistry*, 47(12), pp. 2137–2145, 2001.
- [277] R. Hu, C. Aggarwal, S. Ma, and J. Huai. An Embedding Approach to Anomaly Detection, *IEEE ICDE Conference*, 2016.
- [278] L. Huang, M. I. Jordan, A. Joseph, M. Garofalakis, and N. Taft. In-network PCA and Anomaly Detection. *Neural Information Processing Systems*, 2006.
- [279] J. W. Hunt and T. G. Szlymski. A Fast Algorithm for Computing Longest Common Subsequences. *Communications of the ACM*, 20(5), pp. 350–353, 1977.
- [280] T. Ide and H. Kashima. Eigenspace-based Anomaly Detection in Computer Systems. *ACM KDD Conference*, 2004.
- [281] P. Indyk, R. Motwani, P. Raghavan, and S. Vempala. Locality-preserving Hashing in Multidimensional Spaces. *ACM STOC Conference*, 1997.
- [282] D. Jackson and Y. Chen. Robust Principal Component Analysis and Outlier Detection with Ecological Data. *Environmetrics*, 15, pp. 129–139, 2004.
- [283] A. Jain and R. Dubes. Algorithms for Clustering Data. *Prentice Hall*, 1988.
- [284] H. Jagadish, N. Koudas, and S. Muthukrishnan. Mining Deviants in a Time-Series Database. *VLDB Conference*, 1999.
- [285] V. Janeja and V. Atluri. Random Walks to Identify Anomalous Free-form Spatial Scan Windows. *IEEE Transactions on Knowledge and Data Engineering*, 20(10), 2008.
- [286] P. Janeja and V. Atluri. Spatial Outlier Detection in Heterogeneous Neighborhoods. *Intelligent Data Analysis*, 13(1), 2008.
- [287] R. Jarvis and E. Patrick. Clustering Using a Similarity Measure based on Shared Near Neighbors. *IEEE Transactions on Computers*, 100(11), pp. 1025–1034, 1973.
- [288] H. Javitz and A. Valdez. The SRI IDES Statistical Anomaly Detector. *IEEE Symposium on Security and Privacy*, 1991.
- [289] Y. Jeong, M. Jeong, and O. Omiaomu. Weighted Dynamic Time Warping for Time Series Classification, *Pattern Recognition*, 44, pp. 2231–2240, 2010.
- [290] B. Jiang and J. Pei. Outlier Detection on Uncertain Data: Objects, Instances, and Inferences. *ICDE Conference*, 2011.
- [291] M. F. Jiang, S. S. Tseng, and C. M. Su. Two-phase Clustering Process for Outliers Detection. *Pattern Recognition Letters*, 22, 6–7, pp. 691–700, 2001.
- [292] R. Jiang, H. Fei, and J. Huan. Anomaly Localization for Network Data Streams with Graph Joint Sparse PCA. *ACM KDD Conference*, 2011.
- [293] W. Jin, A. Tung, and J. Han. Mining Top- $n$  Local Outliers in Large Databases. *ACM KDD Conference*, 2001.
- [294] W. Jin, A. Tung, J. Han, and W. Wang. Ranking outliers using symmetric neighborhood relationship. *PAKDD Conference*, 2006.

- [295] T. Johnson, I. Kwok, and R. Ng. Fast Computation of 2-dimensional Depth Contours. *ACM KDD Conference*, 1998.
- [296] I. Jolliffe. *Principal Component Analysis*. Springer, 2002.
- [297] A. Ng, M. Jordan, and Y. Weiss. On Spectral Clustering Analysis and an Algorithm. *Neural Information Processing Systems*, pp. 849–856, 2001.
- [298] M. Joshi, R. Agarwal, and V. Kumar. Mining Needles in a Haystack: Classifying Rare Classes via Two-Phase Rule Induction, *ACM SIGMOD Conference*, 2001.
- [299] M. Joshi, V. Kumar, and R. Agarwal. Evaluating Boosting Algorithms to Classify Rare Classes: Comparison and Improvements. *ICDM Conference*, pp. 257–264, 2001.
- [300] M. Joshi and R. Agarwal. PNRule: A Framework for Learning Classifier Models in Data Mining (A Case Study in Network Intrusion Detection). *SIAM Conference on Data Mining*, 2001.
- [301] M. Joshi, R. Agarwal, and V. Kumar. Predicting Rare Classes: Can Boosting Make Any Weak Learner Strong? *ACM KDD Conference*, 2002.
- [302] M. Joshi. On Evaluating Performance of Classifiers on Rare Classes. *ICDM Conference*, 2003.
- [303] P. Juszczak and R. P. W. Duin. Uncertainty Sampling Methods for One-class Classifiers. *ICML Workshop on Learning from Imbalanced Data Sets*, 2003.
- [304] J. Kang, S. Shekhar, C. Wennen, and P. Novak. Discovering Flow Anomalies: A SWEET Approach. *ICDM Conference*, 2008.
- [305] G. Karakoulas and J. Shawe-Taylor. Optimising Classifiers for Imbalanced Training Sets. *Neural Information Processing Systems*, 1998.
- [306] S. Kasiviswanathan, P. Melville, and A. Banerjee. Emerging Topic Detection using Dictionary Learning. *CIKM Conference*, 2011.
- [307] L. Kaufman and P. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley-Interscience, 1990.
- [308] F. Keller, E. Muller, K. Bohm. HiCS: High-Contrast Subspaces for Density-based Outlier Ranking. *IEEE ICDE Conference*, 2012.
- [309] M. Kendall and J. Ord. *Time-series*. Hodder Arnold, 1990.
- [310] E. Keogh, S. Lonardi, and B. Y.-C. Chiu. Finding Surprising Patterns in a Time Series Database in Linear Time and Space. *ACM KDD Conference*, 2002.
- [311] E. Keogh, J. Lin, and A. Fu. HOT SAX: Finding the Most Unusual Time Series Subsequence: Algorithms and Applications. *ICDM Conference*, 2005.
- [312] E. Keogh, S. Lonardi, and C. Ratanamahatana. Towards Parameter-Free Data Mining. *ACM KDD Conference*, 2004.
- [313] S. Khan and M. Madden. One-class Classification: Taxonomy of Study and Review of Techniques. *Knowledge Engineering Review*, 29(03), 345–374, 2014.



- [314] N. L. D. Khoa and S. Chawla. Incremental Commute Time using Random Walks and Online Anomaly Detection. *ECML/PKDD Conference*, 2016.
- [315] D. Kifer, S. Ben-David, and J. Gehrke. Detecting Change in Data Streams. *VLDB Conference*, 2004.
- [316] J. Kim and C. Scott. Robust Kernel Density Estimation. *Journal of Machine Learning Research*, 13, pp. 2529–2565, 2012. <http://www.jmlr.org/papers/volume13/kim12b/kim12b.pdf>
- [317] E. Knorr and R. Ng. Algorithms for Mining Distance-based Outliers in Large Datasets. *VLDB Conference*, 1998.
- [318] E. Knorr and R. Ng. Finding Intensional Knowledge of Distance-Based Outliers. *VLDB Conference*, 1999.
- [319] E. M. Knorr, R. T. Ng, and V. Tucakov. Distance-based Outliers: Algorithms and Applications. *VLDB Journal*, 8(3), pp. 237–253, February 2000.
- [320] J. Koh, M.-L. Lee, W. Hsu, and W. Ang. Correlation-based Attribute Outlier Detection in XML. *ICDE Conference*, 2008.
- [321] G. Kollios, D. Gunopulos, N. Koudas, and S. Berchtold. Efficient Biased Sampling for Approximate Clustering and Outlier Detection in Large Data Sets. *IEEE Transactions on Knowledge and Data Engineering*, 15(5), pp. 1170–1187, 2003.
- [322] M. Kontaki, A. Gounaris, A. Papadopoulos, K. Tsihlias, and Y. Manolopoulos. Continuous Monitoring of Distance-based Outliers over Data Streams. *ICDE Conference*, 2011.
- [323] K. Kontonasis and T. Bie. An Information-Theoretic Approach to Finding Noisy Tiles in Binary Databases. *SIAM Conference on Data Mining*, 2003.
- [324] Y. Kou, C. T. Lu, and D. Chen. Spatial Weighted Outlier Detection. *SIAM Conference on Data Mining*, 2006.
- [325] H.-P. Kriegel, M. Schubert, and A. Zimek. Angle-based Outlier Detection in High-Dimensional Data. *ACM KDD Conference*, 2008.
- [326] H.-P. Kriegel, P. Kroger, and A. Zimek. Outlier Detection Techniques. *Conference Tutorial at SIAM Data Mining Conference*, 2010. <http://www.siam.org/meetings/sdm10/tutorial3.pdf>
- [327] H.-P. Kriegel, P. Kroger, E. Schubert, and A. Zimek. Outlier Detection in Axis-Parallel Subspaces of High Dimensional Data. *PAKDD Conference*, 2009.
- [328] H.-P. Kriegel, P. Kroger, E. Schubert, and A. Zimek. Outlier Detection in Arbitrarily Oriented Subspaces. *ICDM Conference*, 2012.
- [329] C. Kruegel and G. Vigna. Anomaly-Detection of Web-based Attacks, *CCS*, 2005.
- [330] C. Kruegel, D. Mutz, W. Robertson, and F. Valeur. Bayesian Event Classification for Intrusion Detection. *Computer Security Applications Conference*, 2003.

- [331] C. Kruegel, T. Toth, and E. Kirda. Service Specific Anomaly Detection for Network Intrusion Detection. *ACM Symposium on Applied computing*, 2002.
- [332] M. Kubat and S. Matwin. Addressing the Curse of Imbalanced Training Sets: One Sided Selection. *ICML Conference*, 1997.
- [333] L. Kuncheva. Change Detection in Streaming Multivariate Data using Likelihood Detectors. *IEEE Transactions on Knowledge and Data Engineering*, 25(5), pp. 1175–1180, 2013.
- [334] A. Lakhina, M. Crovella, and C. Diot. Mining Anomalies using Traffic Feature Distributions. *ACM SIGCOMM Conference*, pp. 217–228, 2005.
- [335] A. Lakhina, M. Crovella, and C. Diot. Diagnosing Network-wide Traffic Anomalies. *ACM SIGCOMM Conference*, pp. 219–230, 2004.
- [336] G. Lanckriet, L. Ghaoui, and M. Jordan. Robust Novelty Detection with Single Class MPM. *Neural Information Processing Systems*, 2002.
- [337] J. Lafferty, A. McCallum, and F. Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. *ICML Conference*, 2001.
- [338] T. Lane and C. Brodley. Temporal Sequence Learning and Data Reduction for Anomaly Detection. *ACM Transactions on Information and Security*, 2(3), pp. 295–331, 1999.
- [339] T. Lane and C. Brodley. An Application of Machine Learning to Anomaly Detection. *NIST-NCSC National Information Systems Security Conference*, 1997.
- [340] T. Lane and C. Brodley. Sequence matching and learning in anomaly detection for computer security. *AI Approaches to Fraud Detection and Risk Management*, pp. 43–49, 1997.
- [341] R. Lasaponara. On the Use of Principal Component Analysis (PCA) for Evaluating Interannual Vegetation Anomalies from SPOT/VEGETATION NDVI Temporal Series. *Ecological Modeling*, 194(4), pp. 429–434, 2006.
- [342] L. Latecki, A. Lazarevic, and D. Pokrajac. Outlier Detection with Kernel Density Functions. *Machine Learning and Data Mining in Pattern Recognition*, pp. 61–75, 2007.
- [343] J. Laurikkala, M. Juhola, and E. Kentala. Informal Identification of Outliers in Medical Data. *Fifth International Workshop on Intelligent Data Analysis in Medicine and Pharmacology*, pp. 20–24, 2000.
- [344] A. Lazarevic and V. Kumar. Feature Bagging for Outlier Detection. *ACM KDD Conference*, 2005.
- [345] A. Lazarevic, L. Ertöz, V. Kumar, A. Ozgur, and J. Srivastava. A Comparative Study of Anomaly Detection Schemes in Network Intrusion Detection. *SIAM Conference on Data Mining*, 2003.
- [346] S. Q. Le and T. B. Ho. An Association-based Dissimilarity Measure for Categorical Data. *Pattern Recognition Letters*, 26(16), pp. 2549–2557, 2005.

- [347] J.-G. Lee, J. Han, and X. Li. Trajectory Outlier Detection: A Partition-and-detect Framework. *ICDE Conference*, 2008.
- [348] W. Lee and B. Liu. Learning with Positive and Unlabeled Examples using Weighted Logistic Regression. *ICML Conference*, 2003.
- [349] W. Lee, S. Stolfo, and P. Chan. Learning Patterns from Unix Execution Traces for Intrusion Detection. *AAAI Workshop on AI methods in Fraud and Risk Management*, 1997.
- [350] W. Lee, S. Stolfo, and K. Mok. Adaptive Intrusion Detection: A Data Mining Approach. *Artificial Intelligence Review*, 14(6), pp. 533–567, 2000.
- [351] W. Lee and S. Stolfo. Data Mining Approaches for Intrusion Selection. *USENIX Security Symposium*, 1998.
- [352] W. Lee and D. Xiang. Information Theoretic Measures for Anomaly Detection. *IEEE Symposium on Security and Privacy*, 2001.
- [353] N. Lesh, M. J. Zaki, and M. Ogihara. Mining Features for Sequence Classification. *ACM KDD Conference*, 1999.
- [354] C. Leslie, E. Eskin, and W. Noble. The Spectrum Kernel: A String Kernel for SVM Protein Classification. *Pacific Symposium on Biocomputing*, pp. 566–575, 2002.
- [355] X. Li, J. Han, S. Kim, and H. Gonzalez. ROAM: Rule and Motif-based Anomaly Detection in Massive Moving Object Data Sets. *SIAM Conference on Data Mining*, 2007.
- [356] X. Li, B. Liu, and S. Ng. Negative Training Data can be Harmful to Text Classification. *EMNLP*, 2010.
- [357] X. Li, Z. Li, J. Han, and J.-G. Lee. Temporal Outlier Detection in Vehicle Traffic Data. *ICDE Conference*, 2009.
- [358] D. Lin. An Information-theoretic Definition of Similarity. *ICML Conference*, pp. 296–304, 1998.
- [359] Y. Lin and Y. Jeon. Random Forests and Adaptive Nearest Neighbors. *Journal of the American Statistical Association*, 101(474), pp. 578–590, 2006.
- [360] J. Lin, E. Keogh, A. Fu, and H. V. Herle. Approximations to Magic: Finding Unusual Medical Time Series. *Mining Medical Data (CBMS)*, 2005.
- [361] J. Lin, E. Keogh, S. Lonardi, and B. Y.-C. Chiu. A Symbolic Representation of Time Series, with Implications for Streaming Algorithms. *DMKD Workshop*, 2003.
- [362] B. Liu, W. S. Lee, P. Yu, and X. Li. Partially Supervised Text Classification. *ICML Conference*, 2002.
- [363] B. Liu, Y. Dai, X. Li, W. S. Lee, and P. Yu. Building Text Classifiers Using Positive and Unlabeled Examples. *ICDM Conference*, 2003.
- [364] B. Liu, W. Hsu, and Y. Ma. Integrating Classification and Association Rule Mining. *ACM KDD Conference*, pp. 80–86, 1998.

- [365] G. Liu, T. McDaniel, S. Falkow, and S. Karlin. Sequence Anomalies in the *cag7* Gene of the Helicobacter Pylori Pathogenicity Island. *National Academy of Sciences of the United States of America*, 96(12), pp. 7011–7016, 1999.
- [366] L. Liu and X. Fern. Constructing Training Sets for Outlier Detection. *SIAM Conference on Data Mining*, 2012.
- [367] F. T. Liu, K. M. Ting, and Z.-H. Zhou. Isolation Forest. *ICDM Conference*, 2008.
- [368] F. T. Liu, K. M. Ting, and Z. H. Zhou. Isolation-based Anomaly Detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 6(1), 3, 2012.
- [369] F. T. Liu, K. M. Ting, and Z. H. Zhou. On Detecting Clustered Anomalies using SCiForest. *Machine Learning and Knowledge Discovery in Databases*, pp. 274–290, 2010.
- [370] S. Lin and D. Brown. An Outlier-based Data Association Method for Linking Criminal Incidents. *SIAM Conference On Data Mining*, 2003.
- [371] X.-Y. Liu, J. Wu, and Z.-H. Zhou. Exploratory Undersampling for Class-Imbalance Learning. *IEEE Transactions on Systems, Man and Cybernetics– Part B, Cybernetics*, 39(2), pp. 539–550, April 2009.
- [372] X. Liu, X. Wu, H. Wang, R. Zhang, J. Bailey, and K. Ramamohanarao. Mining Distribution Change in Stock Order Data Streams. *ICDE Conference*, 2010.
- [373] Z. Liu, W. Shi, D. Li, and Q. Qin. Partially Supervised Classification – based on Weighted Unlabeled Samples Support Vector Machine. *ADMA*, 2005.
- [374] X. Liu, P. Zhang, and D. Zeng. Sequence Matching for Suspicious activity Detection in Anti-money Laundering. *Lecture Notes in Computer Science*, Vol. 5075, pp. 50–61, 2008.
- [375] S. Loncarin. A Survey of Shape Analysis Techniques. *Pattern Recognition*, 31(5), pp. 983–1001, 1998.
- [376] C.-T. Lu, D. Chen, and Y. Kou. Algorithms for Spatial Outlier Detection. *ICDM Conference*, 2003.
- [377] A. Lung-Yut-Fong, C. Levy-Leduc, and O. Cappe. Distributed Detection/localization of Change-points in High-dimensional Network Traffic Data. *arXiv:0909.5524*, 2009.
- [378] U. von Luxburg. A Tutorial on Spectral Clustering. *Statistics and computing*, 17(4), pp. 395–416, 2007.
- [379] J. Ma and S. Perkins. Online Novelty Detection on Temporal Sequences. *ACM KDD Conference*, 2003.
- [380] J. Ma, L. Saul, S. Savage, and G. Volker. Learning to Detect Malicious URLs. *ACM Transactions on Intelligent Systems and Technology*, 2(3), Article 30, April 2011.
- [381] M. Mahoney and P. Chan. Learning Nonstationary Models of Normal Network Traffic for Detecting Novel Attacks. *ACM KDD Conference*, 2002.

- [382] M. Mahoney and P. Chan. Learning Rules for Anomaly Detection of Hostile Network Traffic. *ICDM Conference*, 2003.
- [383] F. Malliaros, V. Megalooikonomou, and C. Faloutsos. Fast Robustness Estimation in Large Social Graphs: Communities and Anomaly Detection. *SIAM Conference on Data Mining*, 2012.
- [384] L. M. Manevitz and M. Yousef. One-class SVMs for Document Classification. *Journal of Machine Learning Research*, 2: pp. 139–154, 2001.
- [385] L. Manevitz and M. Yousef. One-class Document Classification via Neural Networks. *Neurocomputing*, 70(7), pp. 1466–1481, 2007.
- [386] E. Manzoor, S. Milajerdi, and L. Akoglu. Fast Memory-Efficient Anomaly Detection in Streaming Heterogeneous Graphs. *ACM KDD Conference*, 2016.
- [387] C. Marceau. Characterizing the Behavior of a Program using Multiple-length n-grams. *Workshop on New Security Paradigms*, pp. 101–110, 2000.
- [388] M. Markou and S. Singh. Novelty detection: A Review, Part 1: Statistical Approaches. *Signal Processing*, 83(12), pp. 2481–2497, 2003.
- [389] M. Markou and S. Singh. Novelty Detection: A Review, Part 2: Neural Network-based Approaches. *Signal Processing*, 83(12), pp. 2481–2497, 2003.
- [390] M. Masud, Q. Chen, L. Khan, C. Aggarwal, J. Gao, J. Han, and B. Thuraisingham. Addressing Concept-Evolution in Concept-Drifting Data Streams. *ICDM Conference*, 2010.
- [391] M. Masud, T. Al-Khateeb, L. Khan, C. Aggarwal, J. Gao, J. Han, and B. Thuraisingham. Detecting Recurring and Novel Classes in Concept-Drifting Data Streams. *ICDM Conference*, 2011.
- [392] M. Masud, Q. Chen, L. Khan, C. Aggarwal, J. Gao, J. Han, A. Srivastava, and N. Oza. Classification and Adaptive Novel Class Detection of Feature-Evolving Data Streams. *IEEE Transactions on Knowledge and Data Engineering*, 25(7), pp. 1484–1497, 2013.
- [393] C. McDiarmid. On the Method of Bounded Differences. In *Surveys in Combinatorics*, pp. 148–188, *Cambridge University Press*, Cambridge, 1989.
- [394] P. Melville and R. Mooney. Diverse Ensembles for Active Learning. *ICML Conference*, 2004.
- [395] B. Micenkova, B. McWilliams, and I. Assent. Learning Outlier Ensembles: The Best of Both Worlds – Supervised and Unsupervised. *Outlier Detection and Description Workshop*, 2014. Extended version: <http://arxiv.org/abs/1507.08104>
- [396] C. Michael and A. Ghosh. Two State-based Approaches to Program-based Anomaly Detection. *Computer Security Applications Conference*, pp. 21, 2000.
- [397] B. Miller, N. Bliss, and P. Wolfe. Subgraph Detection using Eigenvector L1-Norms. *Neural Information Processing Systems*, 2010.
- [398] B. Miller, M. Beard, and N. Bliss. Eigenspace Analysis for Threat Detection in Social Networks. *International Conference on Information Fusion*, 2011.

- [399] D. Mladenic and M. Grobelnik. Feature Selection for Unbalanced Class Distribution and Naive Bayes. *ICML Conference*, 1999.
- [400] M. Mongiovi, P. Bogdanov, R. Ranca, A. Singh, E. Papalexakis, and C. Faloutsos. SIGSPOT: Mining Significant Anomalous Regions from Time-evolving Networks. *ACM SIGMOD Conference*, 2012.
- [401] F. Moosmann, B. Triggs, and F. Jurie. Fast Discriminative Visual Codebooks using Randomized Clustering Forests. *Neural Information Processing Systems*, pp. 985–992, 2006.
- [402] E. Muller, M. Schiffer, and T. Seidl. Statistical Selection of Relevant Subspace Projections for Outlier Ranking. *ICDE Conference*, pp. 434–445, 2011.
- [403] E. Muller, M. Schiffer, P. Gerwert, M. Hannen, T. Jansen, and T. Seidl. SOREX: Subspace Outlier Ranking Exploration Toolkit. *Joint ECML PKDD Conference*, 2010.
- [404] E. Muller, M. Schiffer, and T. Seidl. Adaptive Outlierness for Subspace Outlier Ranking. *CIKM Conference*, 2010.
- [405] E. Muller, F. Keller, S. Blanc, and K. Bohm. OutRules: A Framework for Outlier Descriptions in Multiple Context Spaces. *ECML/PKDD Conference*, 2012.
- [406] E. Muller, I. Assent, P. Iglesias, Y. Mülle, and K. Bohm. Outlier Analysis via Subspace Analysis in Multiple Views of the Data. *ICDM Conference*, 2012.
- [407] R. Motwani and P. Raghavan. Randomized Algorithms. *Cambridge University Press*, 1995.
- [408] A. Mueen, E. Keogh, and N. Young. Logical-Shapelets: An Expressive Primitive for Time Series Classification. *ACM KDD Conference*, 2011.
- [409] A. Naftel and S. Khalid. Classifying Spatiotemporal Object Trajectories using Unsupervised Learning in the Coefficient Feature Space. *Multimedia Systems*, 12(3), pp. 227–238, 2006.
- [410] K. Narita and H. Kitagawa. Outlier Detection for Transaction Databases using Association Rules. *WAIM*, 2008.
- [411] H. Nguyen, V. Gopalkrishnan, and I. Assent. An Unbiased Distance-based Outlier Detection Approach for High Dimensional Data. *DASFAA*, 2011.
- [412] H. Nguyen, H. Ang, and V. Gopalakrishnan. Mining Ensembles of Heterogeneous Detectors on Random Subspaces. *DASFAA*, 2010.
- [413] H. Nguyen, E. Muller, J. Vreeken, F. Keller, and K. Bohm. CMI: An Information-Theoretic Contrast Measure for Enhancing Subspace Cluster and Outlier Detection. *SIAM Conference on Data Mining (SDM)*, pp. 198–206, 2013.
- [414] V. Niennattrakul, E. Keogh, and C. Ratanamahatana. Data Editing Techniques to Allow the Applicability of Distance-based Outlier Detection in Streams. *ICDM Conference*, 2010.

- [415] H. Ning, W. Xu, Y. Chi, Y. Gong, and T. Huang. Incremental Spectral Clustering With Application to Monitoring of Evolving Blog Communities. *SIAM Conference on Data Mining*, 2007.
- [416] C. Noble, and D. Cook. Graph-based Anomaly Detection. *ACM KDD Conference*, 2003.
- [417] K. Noto, C. Brodley, and D. Slonim. FRaC: A Feature-Modeling Approach for Semi-Supervised and Unsupervised Anomaly Detection. *Data Mining and Knowledge Discovery*, 25(1), pp. 109–133, 2012.
- [418] P. Olmo Vaz de Melo, L. Akoglu, C. Faloutsos, and A. Loureiro. Surprising Patterns for the Call Duration Distribution of Mobile Phone Users. *ECML/PKDD Conference*, 2010.
- [419] G. Orair, C. Teixeira, W. Meira Jr, Y. Wang, and S. Parthasarathy. Distance-Based Outlier Detection: Consolidation and Renewed Bearing. *Proceedings of the VLDB Endowment*, 3(1–2), pp. 1469–1480, 2010.
- [420] M. Otey, S. Parthasarathy, A. Ghoting, G. Li, S. Narravula, and D. Panda. Towards NIC-based Intrusion Detection. *ACM KDD Conference*, 2003.
- [421] M. Otey, S. Parthasarathy, and A. Ghoting. Fast Distributed Outlier Detection in Mixed Attribute Data Sets. *Data Mining and Knowledge Discovery*, 12(2–3), pp. 203–228, 2006.
- [422] L. Ott, L. Pang, F. Ramos, and S. Chawla. On Integrated Clustering and Outlier Detection. *Neural Information Processing Systems*, pp. 1359–1367, 2014.
- [423] C. R. Palmer and C. Faloutsos. Electricity based External Similarity of Categorical Attributes. *PAKDD Conference*, 2003.
- [424] Y. Panatier. Variowin. Software For Spatial Data Analysis in 2D. *New York: Springer-Verlag*, 1996.
- [425] C. Papadimitriou, P. Raghavan, H. Tamakai, and S. Vempala. Latent Semantic Indexing: A Probabilistic Analysis. *ACM PODS Conference*, 1998.
- [426] S. Papadimitriou, H. Kitagawa, P. Gibbons, and C. Faloutsos. LOCI: Fast Outlier Detection using the Local Correlation Integral. *ICDE Conference*, 2003.
- [427] S. Papadimitriou, J. Sun, and C. Faloutsos. SPIRIT: Streaming Pattern Discovery in Multiple Time-Series. *VLDB Conference*, 2005.
- [428] L. Parra, G. Deco, and S. Andmiesbach. Statistical Independence and Novelty Detection with Information Preserving Nonlinear Maps. *Neural Computation*, 8(2), pp. 260–269, 1996.
- [429] H. Paulheim and R. Meusel. A Decomposition of the Outlier Detection Problem into a Set of Supervised Learning Problems. *Machine Learning*, 100(2–3), pp. 509–531, 2015.
- [430] Y. Pei, O. Zaiane, and Y. Gao. An Efficient Reference-based Approach to Outlier Detection in Large Datasets. *ICDM Conference*, 2006.

- [431] D. Pelleg and A. Moore. Active Learning for Anomaly and Rare Category Detection. *Neural Information Processing Systems*, 2004.
- [432] Z. Peng and F. Chu. Review Application of the Wavelet Transform in Machine Condition Monitoring and Fault Diagnostics. *Mechanical Systems and Signal Processing*, 18(2), pp. 199–221, March 2004.
- [433] B. Perozzi, L. Akoglu, P. Iglesias Sanchez, and E. Muller. Focused Clustering and Outlier Detection in Large Attributed Graphs. *ACM KDD Conference*, 2014.
- [434] B. Perozzi and L. Akoglu. Scalable Anomaly Ranking of Attributed Neighborhoods. *SIAM Conference on Data Mining*, 2016.
- [435] S. Petrovic, M. Osborne, and V. Lavrenko. Streaming First Story Detection with Application to Twitter. *ACL Conference*, pp. 181–189, 2010.
- [436] T. Pevny. Loda: Lightweight On-line Detector of Anomalies. *Machine Learning*, 102(2), pp. 275–304, 2016.
- [437] J. Pickands. Statistical Inference using Extreme Order Statistics. *The Annals of Statistics*, 3(1), pp. 119–131, 1975.
- [438] B. Pincombe. Anomaly Detection in Time Series of Graphs using ARMA Processes. *ASOR Bulletin*, 24(4): 2–10, 2005.
- [439] M. Pinsky. Introduction to Fourier Analysis and Wavelets. *American Mathematical Society*, 2009.
- [440] C. Phua, V. Lee, K. Smith, and R. Gayler. A Comprehensive Survey of Data Mining-based Fraud Detection Research, 2010.  
<http://arxiv.org/abs/1009.6119>.
- [441] C. Phua, D. Alahakoon, and V. Lee. Minority Report in Fraud Detection: Classification of Skewed Data. *ACM SIGKDD Explorations Newsletter*, 6(1), pp. 50–59, 2004.
- [442] B. R. Preiss. Data Structures and Algorithms with Object Oriented Design Patterns in Java. *Wiley*, 1999.
- [443] D. Pokrajac, A. Lazarevic, and L. Latecki. Incremental Local Outlier Detection for Data Streams. *CIDM Conference*, 2007.
- [444] C. Potter, P. N. Tan, M. Steinbach, S. Klooster, V. Kumar, R. Myneni, and V. Genovese. Major Disturbance Events in Terrestrial Ecosystems detected using Global Satellite Data Sets. *Global Change Biology*, pp. 1005–1021, 2003.
- [445] A. Pires and C. Santos-Pereira. Using Clustering and Robust Estimators to Detect Outliers in Multivariate Data. *International Conference on Robust Statistics*, 2005.
- [446] L. Portnoy, E. Eskin, and S. Stolfo. Intrusion Detection with Unlabeled Data using Clustering. *ACM Workshop on Data Mining Applied to Security*, 2001.
- [447] B. Prakash, N. Valler, D. Andersen, M. Faloutsos, and C. Faloutsos. BGP-lens: Patterns and Anomalies in Internet Routing Updates. *ACM KDD Conference*, 2009.



- [448] M. Prastawa, E. Bullitt, S. Ho, and G. Gerig. A Brain Tumor Segmentation Framework based on Outlier Detection. *Medical Image Analysis*, 8, pp. 275–283, 2004.
- [449] C. Priebe, J. Conroy, D. Marchette, and Y. Park. Scan Statistics on Enron Graphs. *Computational and Mathematical Organizational Theory*, 11(3), pp. 229–247, 2005.
- [450] F. Provost, and T. Fawcett. Analysis and Visualization of Classifier Performance: Comparison under Imprecise Class and Cost Distributions. *KDD Conference*, 1997.
- [451] F. Provost, T. Fawcett, and R. Kohavi. The Case against Accuracy Estimation while Comparing Induction Algorithms. *ICML Conference*, 1998.
- [452] G. Qi, C. Aggarwal, and T. Huang. On Clustering Heterogeneous Social Media Objects with Outlier Links. *WSIAM Conference on Data Mining*, 2012.
- [453] J. Quinlan. Learning with Continuous Classes. *Australian Joint Conference on Artificial Intelligence*, 92, pp. 343–348, 1992.
- [454] L. R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, 77(2), pp. 257–285, Feb. 1989.
- [455] M. Radovanovic, A. Nanopoulos, and M. Ivanovic. On the Existence of Obstinate Results in Vector Space Models. *ACM SIGIR Conference*, 2010.
- [456] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient Algorithms for Mining Outliers from Large Data Sets. *ACM SIGMOD Conference*, pp. 427–438, 2000.
- [457] S. Ranshous, S. Shen, D. Koutra, S. Harenberg, C. Faloutsos, and N. Samatova. Anomaly Detection in Dynamic Networks: A Survey. *Wiley Interdisciplinary Reviews: Computational Statistics*, 7(3), pp. 223–247, 2015.
- [458] S. Ranshous, S. Harenberg, K. Sharma, and N. Samatova. A Scalable Approach for Outlier Detection in Edge Streams Using Sketch-based Approximations. *SIAM Conference on Data Mining*, 2016.
- [459] M. Ranzato, C. Poultney, S. Chopra, and Y. LeCun. Efficient Learning of Sparse Representations with an Energy-Based Model. *Neural Information Processing Systems*, 19 pp. 11371144, 2006
- [460] B. Raskutti and A. Kowalczyk. Extreme Rebalancing for SVMs: A Case Study. *SIGKDD Explorations*, 6(1): pp. 60–69, 2004.
- [461] S. Rayana and L. Akoglu. Less is More: Building Selective Anomaly Ensembles. *ACM Transactions on Knowledge Discovery and Data Mining*, 10(4), 42, 2016.
- [462] S. Roberts. Novelty Detection using Extreme Value Statistics. *IEEE Proceedings on Vision, Image and Signal Processing*, 146(3). pp. 124–129, 1999.
- [463] S. Roberts. Extreme Value Statistics for Novelty Detection in Biomedical Signal Processing. *International Conference on Advances in Medical Signal and Information Processing*. pp. 166–172, 2002.
- [464] J. Rogan, J. Miller, D. Stow, J. Franklin, L. Levien, and C. Fischer. Land-Cover Change Monitoring with Classification Trees Using Landsat TM and Ancillary Data. *Photogrammetric Engineering and Remote Sensing*, 69(7), pp. 793–804, 2003.

- [465] D. Ron, Y. Singer, and N. Tishby. The Power of Amnesia: Learning Probabilistic Automata with Variable Memory Length. *Machine Learning*, 25(2–3) pp. 117–149, 1996.
- [466] V. Roth. Kernel Fisher Discriminants for Outlier Detection. *Neural Computation*, 18(4), pp. 942–960, 2006.
- [467] P. Rousseeuw and A. Leroy. Robust Regression and Outlier Detection. *Wiley*, 2003.
- [468] I. Ruts and P. Rousseeuw. Computing Depth Contours of Bivariate Point Clouds. *Computational Statistics and Data Analysis*, 23, pp. 153–168, 1996.
- [469] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz. A Bayesian Approach to Filtering Junk E-mail. *AAAI Workshop on Learning for Text Categorization. Tech. Rep. WS-98-05*, 1998.  
<http://robotics.stanford.edu/users/sahami/papers-dir/spam.pdf>
- [470] R. K. Sahoo, A. J. Oliner, I. Rish, M. Gupta, J. E. Moreira, S. Ma, R. Vilalta, and A. Sivasubramaniam. Critical Event Prediction for Proactive Management in Large-scale Computer Clusters. *ACM KDD Conference*, 2003.
- [471] M. Salehi, C. Leckie, M. Moshtaghi, and T. Vaithianathan. A Relevance Weighted Ensemble Model for Anomaly Detection in Switching Data Streams. *Advances in Knowledge Discovery and Data Mining*, pp. 461–473, 2014.
- [472] G. Salton and M. J. McGill. Introduction to Modern Information Retrieval. *McGraw Hill*, 1986.
- [473] P. Sanchez, E. Muller, F. Laforet, F. Keller, and K. Bohm. Statistical Selection of Congruent Subspaces for Mining Attributed Graphs. *ICDM Conference*, pp. 647–656, 2013.
- [474] S. Sarawagi, R. Agrawal, and N. Megiddo. Discovery-driven Exploration of OLAP Data Cubes. *EDBT Conference*, 1998.
- [475] S. Sathe and C. Aggarwal. LODS: Local Density Meets Spectral Outlier Detection. *SIAM Conference on Data Mining*, 2016.
- [476] S. Sathe and C. Aggarwal. Subspace Outlier Detection in Linear Time with Randomized Hashing. *ICDM Conference*, 2016.
- [477] G. Scarth, M. McIntyre, B. Wowk, and R. Somorjai. Detection of Novelty in Functional Images using Fuzzy Clustering. *Meeting of International Society for Magnetic Resonance in Medicine*, 1995.
- [478] R. Schapire and Y. Singer. Improved Boosting Algorithms using Confidence-rated Predictions. *Annual Conference on Computational Learning Theory*, 1998.
- [479] B. Scholkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the Support of a High-Dimensional Distribution. *Neural Computation*, 13(7), pp. 1443–1472, 2001.
- [480] B. Scholkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt. Support-vector Method for Novelty Detection. *Neural Information Processing Systems*, 2000.

- [481] B. Scholkopf, A. Smola, and K.-R. Muller. Nonlinear Component Analysis as a Kernel Eigenvalue Problem. *Neural Computation*, 10(5), pp. 1299–1319, 1998.
- [482] R. Schoen, T. Habetler, F. Kamran, and R. Bartfield. Motor Bearing Damage Detection using Stator Current Monitoring. *IEEE Transactions on Industry Applications*, 31(6), pp. 1275–1279, 1995.
- [483] E. Schubert, A. Zimek, and H.-P. Kriegel. Generalized Outlier Detection with Flexible Kernel Density Estimates. *SIAM Conference on Data Mining*, pp. 542–50, 2014.
- [484] K. Sequeira and M. Zaki. ADMIT: Anomaly-based Data Mining for Intrusions. *ACM KDD Conference*, 2002.
- [485] H. Seung, M. Opper, and H. Sompolinsky. Query by Committee. *ACM Workshop on Computational Learning Theory*, 1992.
- [486] B. Settles. Active Learning. *Morgan and Claypool*, 2012.
- [487] S. Shekhar, C. T. Lu, and P. Zhang. Detecting Graph-based Spatial Outliers: Algorithms and Applications. *ACM KDD Conference*, 2001.
- [488] S. Shekhar, C. T. Lu, and P. Zhang. A Unified Approach to Detecting Spatial Outliers. *Geoinformatica*, 7(2), pp. 139–166, 2003.
- [489] S. Shekhar and S. Chawla. A Tour of Spatial Databases. *Prentice Hall*, 2002.
- [490] S. Shekhar, C. T. Lu, and P. Zhang. Detecting Graph-based Spatial Outliers. *Intelligent Data Analysis*, 6, pp. 451–468, 2002.
- [491] T. Shi and S. Horvath. Unsupervised Learning with Random Forest Predictors. *Journal of Computational and Graphical Statistics*, 15(1), pp. 118–138, 2006.
- [492] P. Showbridge, M. Kraetzl, and D. Ray. Detection of Abnormal Change in Dynamic Networks. *International Conference on Information, Decision and Control*, pp. 557–562, 1999.
- [493] M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, and L. Chang. A Novel Anomaly Detection Scheme based on Principal Component Classifier. *ICDM Conference*, 2003.
- [494] A. Siebes, J. Vreeken, and M. van Leeuwen. Itemsets than Compress. *SIAM Conference on Data Mining*, 2006.
- [495] J. Silva and R. Willett. Detection of Anomalous Meetings in a Social Network. *SocialCom*, 2008.
- [496] B. W. Silverman. Density Estimation for Statistics and Data Analysis. *Chapman and Hall*, 1986.
- [497] K. Smets and J. Vreeken. The Odd One Out: Identifying and Characterising Anomalies. *SIAM Conference on Data Mining*, 2011.
- [498] E. S. Smirnov. On Exact Methods in Systematics. *Systematic Zoology*, 17(1), pp. 1–13, 1968.

- [499] R. Smith, A. Bivens, M. Embrechts, C. Palagiri, and B. Szymanski. Clustering Approaches for Anomaly Based Intrusion Detection. *Intelligent Engineering Systems through Artificial Neural Networks*, 2002.
- [500] P. Smyth. Clustering Sequences with Hidden Markov Models. *Neural Information Processing*, 1997.
- [501] P. Smyth. Markov Monitoring with Unknown States. *IEEE Journal on Selected Areas in Communications*, 12(9), pp. 1600-1612, 1994.
- [502] H. Solberg and A. Lahti. Detection of Outliers in Reference Distributions: Performance of Horn's Algorithm. *Clinical Chemistry*, 51(12), pp. 2326-2332, 2005.
- [503] X. Song, M. Wu, C. Jermaine, and S. Ranka. Conditional Anomaly Detection. *IEEE Transaction on Knowledge and Data Engineering*, 19(5), pp. 631-645, 2007.
- [504] X. Song, M. Wu, C. Jermaine, and S. Ranka. Statistical Change Detection for Multi-dimensional Data. *ACM KDD Conference*, 2007.
- [505] C. Spence, L. Parra, and P. Sajda. Detection, Synthesis and Compression in Mammographic Image Analysis with a Hierarchical Image Probability Model. *IEEE Workshop on Mathematical Methods in Biomedical Image Analysis*, 2001.
- [506] A. Srivastava. Discovering System Health Anomalies using Data Mining Techniques. *Joint Army Navy NASA Airforce Conference on Propulsion*, 2005.
- [507] A. Srivastava. Enabling the Discovery of Recurring Anomalies in Aerospace Problem Reports using High-dimensional Clustering Techniques. *Aerospace Conference*, 2006.
- [508] A. Srivastava and B. Zane-Ulman. Discovering Recurring Anomalies in Text Reports regarding Complex Space Systems. *Aerospace Conference*, 2005.
- [509] A. Srivastava, A. Kundu, S. Sural, and A. Majumdar. Credit Card Fraud Detection using Hidden Markov Model. *IEEE Transactions on Dependable and Secure Computing*, 5(1), pp. 37-48, 2008.
- [510] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(1), pp. 1929-1958, 2014.
- [511] P. Srivastava, D. Desai, S. Nandi, and A. Lynn. HMM-ModE-Improved Classification using Profile Hidden Markov Models by Optimizing the Discrimination Threshold and Modifying Emission Probabilities with Negative Training Sequences. *BMC Bioinformatics*, 8 (104), 2007.
- [512] M. Stephens. Use of the Kolmogorov-Smirnov, Cramer-von Mises and Related Statistics without Extensive Tables. *Journal of the Royal Statistical Society. Series B*, pp. 115-122, 1970.
- [513] S. Stolfo, D. Fan, W. Lee, A.L. Prodromidis, and P. Chan. Credit Card Fraud Detection Using Meta-Learning: Issues and Initial Results. *AAAI Workshop AI Methods in Fraud and Risk Management*, pp. 83-90, 1997.

- [514] S. Stolfo, D. Fan, W. Lee, A. Prodromidis, and P. Chan. Cost-Based Modeling for Fraud and Intrusion Detection: Results from the JAM Project. *DARPA Information Survivability Conf. and Exposition*, 2, pp. 130–144, 2000.
- [515] G. Strang. An Introduction to Linear Algebra. *Wellesley Cambridge Press*, 2009.
- [516] A. Strehl and J. Ghosh. Cluster Ensembles– A Knowledge Reuse Framework for Combining Multiple Partitions. *Jour. of Machine Learning Research*, 3, pp. 583–617, 2003.
- [517] S. Subramaniam, T. Palpanas, D. Papadopoulos, V. Kalogeraki, and D. Gunopulos. Online Outlier Detection in Sensor Data using Non-parametric Models. *VLDB Conference*, 2006.
- [518] H. Sun, Y. Bao., F. Zhao, G. Yu, and D. Wang. CD-Trees: An Efficient Index Structure for Outlier Detection. *Web-Age Information Management (WAIM)*, pp. 600–609, 2004.
- [519] J. Sun, S. Papadimitriou, P. Yu, and C. Faloutsos. Graphscope: Parameter-free Mining of Large Time-Evolving Graphs. *ACM KDD Conference*, 2007.
- [520] J. Sun, D. Tao, and C. Faloutsos. Beyond Streams and Graphs: Dynamic Tensor Analysis. *ACM KDD Conference*, 2006.
- [521] J. Sun, H. Qu, D. Chakrabarti, and C. Faloutsos. Neighborhood Formation and Anomaly Detection in Bipartite Graphs. *ICDM Conference*, 2005.
- [522] J. Sun, Y. Xie, H. Zhang, and C. Faloutsos. Less is More: Compact Matrix Representation of Large Sparse Graphs. *SIAM Conference on Data Mining*, 2007.
- [523] P. Sun and S. Chawla. On Local Spatial Outliers. *IEEE ICDM Conference*, 2004.
- [524] P. Sun, S. Chawla, and B. Arunasalam. Mining for Outliers in Sequential Databases. *SIAM Conference on Data Mining*, 2006.
- [525] Y. Sun, Y. Yu, and J. Han. Ranking-based Clustering of Heterogeneous Information Networks with Star Network Schema. *ACM KDD Conference*, 2009.
- [526] C. Surace and K. Worden. A Novelty Detection Method to Diagnose Damage in Structures: An Application to an Offshore Platform. *International Conference of Offshore and Polar Engineering*, 4, pp. 64–70, 1998.
- [527] C. Surace, K. Worden, and G. Tomlinson. A Novelty Detection Approach to Diagnose Damage in a Cracked Beam. *Proceedings of SPIE*, Vol. 3089, pp. 947–953, 1997.
- [528] E. Suzuki, T. Watanabe, H. Yokoi, and K. Takabayashi. Detecting Interesting Exceptions from Medical Test Data with Visual Summarization. *ICDM Conference*, pp. 315–322, 2003.
- [529] P. Sykacek. Equivalent Error Bars for Neural Network Classifiers trained by Bayesian Inference. *ESANN*, 1997.
- [530] T. Takahashi, R. Tomioka, and K. Yamanishi. Discovering Emerging Topics in Social Streams via Link Anomaly Detection. *ICDM Conference*, 2011.
- [531] P.-N. Tan and V. Kumar. Discovery of Web Robot Sessions based on their Navigational Patterns. *Data Mining and Knowledge Discovery*, 6(1), pp. 9–35, 2002.

- [532] S. C. Tan, K. M. Ting, and T. F. Liu. Fast Anomaly Detection for Streaming Data. *IJCAI Conference*, 2011.
- [533] Y. Tao, X. Xiao, and S. Zhou. Mining Distance-based Outliers from Large Databases in any Metric Space. *ACM KDD Conference*, 2006.
- [534] J. Tang, Z. Chen, A. W.-C. Fu, and D. W. Cheung. Enhancing Effectiveness of Outlier Detections for Low Density Patterns. *PAKDD Conference*, 2002.
- [535] Y. Tang, Y.-Q. Zhang, N. V. Chawla, and S. Krasser. SVMs Modeling for Highly Imbalanced Classification. *IEEE Transactions on Systems, Man and Cybernetics- Part B: Cybernetics*, 39(1), pp. 281–288, 2009.
- [536] M. Taniguchi, M. Haft, J. Hollmen, and V. Tresp. Fraud Detection in Communications Networks using Neural and Probabilistic Methods. *IEEE International Conference in Acoustics, Speech and Signal Processing*, 2, pp. 1241–1444, 1998.
- [537] L. Tarassenko. Novelty detection for the Identification of Masses in Mammograms. *IEEE International Conference on Artificial Neural Networks*, 4, pp. 442–447, 1995.
- [538] D. Tax. One Class Classification: Concept-learning in the Absence of Counter-examples, *Doctoral Dissertation, University of Delft, Netherlands*, 2001. <http://prlab.tudelft.nl/sites/default/files/thesis.pdf>
- [539] D. Tax and R. Duin. Support Vector Data Description. *Machine learning*, 54(1), 45-66, 2004.
- [540] D. Tax and R. Duin. Combining One-Class Classifiers. *Multiple Classifier Systems*, pp. 299–308, 2001.
- [541] D. Tax and P. Juszczak. Kernel Whitening for One-Class Classification. *Pattern Recognition with Support Vector Machines*, pp. 40–52, 2002.
- [542] J. Tenenbaum, V. De Silva, and J. C. Langford. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, 290, 5500, pp. 2319–2323, 2000.
- [543] P. Thompson, D. MacDonald, M. Mega, C. Holmes, A. Evans, and A. Toga. Detection and Mapping of Abnormal Brain Structure with a Probabilistic Atlas of Cortical Surfaces. *Journal of Computer Assisted Tomography*, 21(4), pp. 567–581, 1997.
- [544] M. Thottan and C. Ji. Anomaly Detection in IP Networks. *IEEE Transactions on Signal Processing*, 51(8), pp. 2191–2204, 2003.
- [545] S. Tian, S. Mu, and C. Yin. Sequence-similarity Kernels for SVMs to Detect Anomalies in System Calls. *Neurocomputing*, 70(4–6), pp. 859–866, 2007.
- [546] K. M. Ting. An Instance-Weighting Method to Induce Cost-sensitive Trees. *IEEE Transaction on Knowledge and Data Engineering*, 14: pp. 659–665, 2002.
- [547] K. M. Ting, G. T. Zhou, F. T. Liu, and S. C. Tan. Mass Estimation. *Machine Learning*, 90(1), pp. 127–160, 2013. [Short version in ACM KDD Conference, 2010]
- [548] K. M. Ting, Y. Zhu, M. Carman, and Y. Zhu. Overcoming Key Weaknesses of Distance-Based Neighbourhood Methods using a Data Dependent Dissimilarity Measure. *ACM KDD Conference*, 2016.

- [549] M. E. Tipping and C. M. Bishop. Probabilistic Principal Component Analysis. *Journal of the Royal Statistical Society*, B 61, pp. 611–622, 1999.
- [550] W. Tobler. Cellular Geography. In *Philosophy in Geography*, Dordrecht Reidel Publishing Company, pp. 379–386, 1979.
- [551] H. Tong and C.-Y. Lin. Non-Negative Residual Matrix Factorization with Application to Graph Anomaly Detection. *SIAM Conference on Data Mining*, 2011.
- [552] L. Toth and G. Gosztolya. Replicator Neural Networks for Outlier Modeling in Segmental Speech Recognition. *LNCS*, Vol. 3172, pp. 996–1001, 2004.
- [553] K. van Leemput, F. Maes, D. Vandermeulen, A. Colchester, and P. Suetens. Automated Segmentation of Multiple Sclerosis Lesions by Model Outlier Detection. *IEEE Transactions on Medical Imaging*, vol. 20, pp. 677–688, August 2001.
- [554] V. Venkatasubramanian, R. Rengaswamy, K. Yin, and S. N. Kavuri. A Review of Process Fault Detection and Diagnosis Part I: Quantitative Model-based Methods. *Computers and Chemical Engineering*, 27(3), pp. 293–311, 2003.
- [555] C. Vens and F. Costa. Random Forest based Feature Induction. *IEEE ICDM Conference*, pp. 744–753, 2011.
- [556] S. Viaene, R. Derrig, B. Baesens, and G. Dedene. A Comparison of State-of-the-Art Classification Techniques for Expert Automobile Insurance Claim Fraud Detection. *Journal of Risk and Insurance*, 69(3), pp. 373–421, 2002.
- [557] T. De Vries, S. Chawla, and M. Houle. Finding Local Anomalies in Very High Dimensional Space. *ICDM Conference*, 2010.
- [558] N. Wale, X. Ning, and G. Karypis. Trends in Chemical Data Mining. *Managing and Mining Graph Data*, Springer, 2010.
- [559] B. Wang, G. Xiao, H. Yu, and X. Yang. Distance-based Outlier Detection on Uncertain Data. *International Conference on Computer and Information Technology*, 2009.
- [560] B. Wang, X. Yang, G. Wang, and G. Yu. Outlier Detection over Sliding Windows for Probabilistic Data Streams. *Journal of Computer Science and Technology*, 25(3), pp. 389–400, 2010.
- [561] M. Wang, C. Zhang, and J. Yu. Native API-based Windows Anomaly Intrusion Detection Method using SVM. *International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*, 2006.
- [562] X. Wang, C. Zhai, X. Hu, and R. Sproat. Mining Correlated Bursty Topic Patterns from Coordinated Text Streams. *ACM KDD Conference*, 2007.
- [563] L. Wei, W. Qian, A. Zhou, and W. Jin. HOT: Hypergraph-based Outlier Test for Categorical Data. *PAKDD Conference*, 2007.
- [564] L. Wei and E. Keogh. Semi-supervised Time Series Classification. *ACM KDD Conference*, 2006.
- [565] L. Wei, E. Keogh, and X. Xi. SAXually Explicit Images: Finding Unusual Shapes. *ICDM Conference*, 2006.

- [566] G. Weiss and F. Provost. Learning When Training Data are Costly: The Effect of Class Distribution on Tree Induction. *Journal of Artificial Intelligence Research*, 19: pp. 315–354, 2003.
- [567] G. Williams, R. Baxter, H. He, S. Hawkins, and L. Gu. A Comparative Study of RNN for Outlier Detection in Data Mining. *IEEE ICDM Conference*, 2002.
- [568] W.-K. Wong, A. Moore, G. Cooper, and M. Wagner. Rule-based Anomaly Pattern Detection for Detecting Disease Outbreaks. *National Conference on Artificial Intelligence*, 2002.
- [569] E. Wu, W. Liu, and S. Chawla. Spatio-temporal Outlier Detection in Precipitation Data. *Knowledge Discovery from Sensor Data, Springer, LNCS 5840*, 2008.
- [570] G. Wu and E. Y. Chang. Class-boundary Alignment for Imbalanced Dataset Learning. *ICML Workshop on Learning from Imbalanced Data Sets*, 2003.
- [571] K. Wu, K. Zhang, W. Fan, A. Edwards, and P. Yu. RS-Forest: A Rapid Density Estimator for Streaming Anomaly Detection. *ICDM Conference*, pp. 600–609, 2014.
- [572] M. Wu and C. Jermaine. Outlier Detection by Sampling with Accuracy Guarantees. *ACM KDD Conference*, 2006.
- [573] M. Wu, X. Song, C. Jermaine, S. Ranka, and J. Gums. A LRT Framework for Fast Spatial Anomaly Detection. *ACM KDD Conference*, 2009.
- [574] X. Xi, E. Keogh, C. Shelton, L. Wei, and C. Ratanamahatana. Fast Time Series Classification using Numerosity Reduction. *ICML Conference*, 2006.
- [575] Z. Xing, J. Pei, and E. Keogh. A Brief Survey on Sequence Classification. *ACM SIGKDD Explorations*, 12(1), 2010.
- [576] L. Xiong, X. Chen, and J. Schneider. Direct Robust Matrix Factorization for Anomaly Detection. *ICDM Conference*, 2011.
- [577] L. Xiong, B. Poczos, J. Schneider, A. Connolly, and J. VanderPlas. Hierarchical Probabilistic Models for Group Anomaly Detection, *Artificial Intelligence and Statistics*, 2011.
- [578] K. Yaminshi, J. Takeuchi, and G. Williams. Online Unsupervised Outlier Detection using Finite Mixtures with Discounted Learning Algorithms. *ACM KDD Conference*, 2000.
- [579] K. Yaminshi and J. Takeuchi. A Unified Framework for Detecting Outliers and Change Points from Time Series Data. *ACM KDD Conference*, 2002.
- [580] R. Yan, Y. Liu, R. Jin, and A. Hauptmann. On Predicting Rare Classes with SVM Ensembles in Scene Classification. *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2003.
- [581] J. Yang and W. Wang. CLUSEQ: Efficient and Effective Sequence Clustering. *ICDE Conference*, 2003.
- [582] P. Yang and Q. Zhu. Finding Key Outlying Subspaces for Outlier Detection. *Knowledge-based Systems*, 24(2), pp. 269–274, 2011.



- [583] X. Yang, L. Latecki, and D. Pokrajac. Outlier Detection with Globally Optimal Exemplar-based GMM. *SIAM Conference on Data Mining*, 2009.
- [584] Y. Yang, J. Carbonell, R. Brown, T. Pierce, B. T. Archibald, and X. Liu. Learning Approaches for Detecting and Tracking News Events. *IEEE Intelligent Systems*, 14(4):32–43, 1999.
- [585] Y. Yang, T. Pierce, and J. Carbonell. A Study on Retrospective and On-line Event Detection. *ACM SIGIR Conference*, 1998.
- [586] Y. Yang, J. Zhang, J. Carbonell, and C. Jin. Topic-conditioned Novelty Detection. *ACM KDD Conference*, 2002.
- [587] D. Yankov, E. Keogh, and U. Rebbapragada. Disk-aware Discord Discovery: Finding Unusual Time Series in Terabyte Sized Data Sets. *ICDM Conference*, 2007.
- [588] N. Ye. A Markov Chain Model of Temporal Behavior for Anomaly Detection. *IEEE Information Assurance Workshop*, 2004.
- [589] N. Ye and Q. Chen. An Anomaly Detection Technique based on a Chi-square Statistic for Detecting Intrusions into Information Systems. *Quality and Reliability Engineering International*, 17, pp. 105–112, 2001.
- [590] L. Ye and E. Keogh. Time Series Shapelets: a New Primitive for Data Mining. *ACM KDD Conference*, 2009.
- [591] Q. Ye and W. Zhi. Outlier Detection in the Framework of Dimensionality Reduction. *International Journal of Pattern Recognition and Artificial Intelligence*, 29(4), 2015.
- [592] B.-K. Yi, N. D. Sidiropoulos, T. Johnson, H. Jagadish, C. Faloutsos, and A. Biliris. Online Data Mining for Co-evolving Time Sequences. *ICDE Conference*, 2000.
- [593] A. Ypma and R. Duin. Novelty Detection using Self-Organizing Maps. *Progress in Connectionist-Based Information Systems*, 2, pp. 1322–1325, 1997.
- [594] D. Yu, G. Sheikholeslami, and A. Zhang. Findout: Finding Outliers in Very Large Datasets. *Knowledge And Information Systems*, 4(4), pp. 387–412, 2002.
- [595] H. Yu, J. Han, and K. C.-C. Chang. PEBL: Web Page Classification without Negative Examples. *IEEE Transactions on Knowledge and Data Engineering*, 16(1), pp. 70–81, 2004.
- [596] J. X. Yu, W. Qian, H. Lu, and A. Zhou. Finding Centric Local Outliers in Categorical/Numeric Spaces. *Knowledge and Information Systems*, 9(3), pp. 309–338, 2006.
- [597] W. Yu, C. Aggarwal, S. Ma, and H. Wang. On Anomalous Hotspot Discovery in Graph Streams. *IEEE ICDM Conference*, 2013.
- [598] W. Yu, C. Aggarwal, and W. Wang. Temporally Factorized Network Modeling for Evolutionary Network Analysis. *WSDM Conference*, 2017.
- [599] B. Zadrozny and C. Elkan. Transforming Classifier Scores into Accurate Multiclass Probability Estimates. *ACM KDD Conference*, 2002.

- [600] B. Zadrozny, J. Langford, and N. Abe. Cost-Sensitive Learning by Cost-Proportionate Example Weighting, *ICDM Conference*, 2003.
- [601] B. Zadrozny and C. Elkan. Learning and Making Decisions when Costs and Probabilities are Unknown. *KDD Conference*, 2001.
- [602] D. Zhang and G. Lu. Review of Shape Representation and Description Techniques. *Pattern Recognition*, 37(1), pp. 1–19, 2004.
- [603] D. Zhang and W. S. Lee. A Simple Probabilistic Approach to Learning from Positive and Unlabeled Examples. *Annual UK Workshop on Computational Intelligence*, 2005.
- [604] J. Zhang, Q. Gao, and H. Wang. SPOT: A System for Detecting Projected Outliers from High-Dimensional Data Stream. *ICDE Conference*, 2008.
- [605] J. Zhang, M. Lou, T. W. Ling, and H. Wang. HOS-Miner: A System for Detecting Outlying Subspaces of High-dimensional Data. *VLDB Conference*, 2004.
- [606] J. Zhang, Q. Gao, and H. Wang. A Novel Method for Detecting Outlying Subspaces in High-dimensional Databases Using Genetic Algorithm. *ICDM Conference*, 2006.
- [607] J. Zhang and H. Wang. Detecting Outlying Subspaces for High-Dimensional Data: the New Task, Algorithms and Performance. *Knowledge and Information Systems*, 10(3), pp. 333–355, 2006.
- [608] J. Zhang, Z. Ghahramani, and Y. Yang. A Probabilistic Model for Online Document Clustering with Application to Novelty Detection. *Neural Information Processing Systems*, 2005.
- [609] J. Zhang and I. Mani. KNN Approach to Unbalanced Data Distributions: A Case Study Involving Information Extraction. *ICML Workshop on Learning from Imbalanced Datasets*, 2003.
- [610] K. Zhang, M. Hutter, and H. Jin. A New Local Distance-Based Outlier Detection Approach for Scattered Real-World Data. *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 813–822, 2009.
- [611] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An Efficient Data Clustering Method for Very Large Databases. *ACM SIGMOD Conference*, 1996.
- [612] X. Zhang, P. Fan, and Z. Zhu. A New Anomaly Detection Method based on Hierarchical HMM. *International Conference on Parallel and Distributed Computing, Applications, and Technologies*, 2003.
- [613] Y. Zhang, J. Hong, and L. Cranor. CANTINA: A Content-based Approach to Detecting Phishing Web Sites. *WWW Conference*, 2007.
- [614] Y. Zhang, P. Meratnia, and P. Havinga. Outlier Detection for Wireless Sensor Networks: A Survey. *IEEE Communications Surveys and Tutorials*, 12(2), 2010.
- [615] J. Zhao, C.-T. Lu, and Y. Kou. Detecting Region Outliers in Meteorological Data. *ACM GIS Conference*, 2003.
- [616] Z. Zheng, X. Wu, and R. Srihari. Feature Selection for Text Categorization on Imbalanced Data. *SIGKDD Explorations*, 6(1), pp. 80–89, 2004.

- [617] Z.-H. Zhou. Ensemble Methods: Foundations and Algorithms. *Chapman and Hall/CRC Press*, 2012.
- [618] C. Zhu, H. Kitagawa, S. Papadimitriou, and C. Faloutsos. OBE: Outlier by Example. *PAKDD Conference*, 2004.
- [619] C. Zhu, H. Kitagawa, and C. Faloutsos. Example-based Robust Outlier Detection in High Dimensional Data Sets. *ICDM Conference*, 2005.
- [620] A. Zimek, A. Schubert, and H.-P. Kriegel. A Survey on Unsupervised Outlier Detection in High-dimensional Numerical Data. *Statistical Analysis and Data Mining*, 5(5), pp. 363–387, 2012.
- [621] A. Zimek, M. Gaudet, R. Campello, and J. Sander. Subsampling for Efficient and Effective Unsupervised Outlier Detection Ensembles. *KDD Conference*, 2013.
- [622] <http://www.itl.nist.gov/iad/mig/tests/tdt/tasks/fsd.html>
- [623] D. D. Lewis. Reuters-21578 Data Set.  
<http://www.daviddlewis.com/resources/test-collections/reuters21578>.
- [624] <http://kdd.ics.uci.edu/databases/20newsgroups>
- [625] <http://www.informatik.uni-trier.de/~ley/db/>
- [626] <http://www.kdnuggets.com/software/deviation.html>
- [627] <http://www.kdnuggets.com/software/index.html>
- [628] <http://www.cs.waikato.ac.nz/ml/weka/>
- [629] [http://scikit-learn.org/stable/auto\\_examples/svm/plot\\_oneclass.html](http://scikit-learn.org/stable/auto_examples/svm/plot_oneclass.html)
- [630] <http://scikit-learn.org/dev/modules/generated/sklearn.ensemble.IsolationForest.html>
- [631] <https://sourceforge.net/projects/iforest/>
- [632] [http://scikit-learn.org/stable/modules/outlier\\_detection.html](http://scikit-learn.org/stable/modules/outlier_detection.html)
- [633] <http://elki.dbs.ifi.lmu.de/>
- [634] <http://www.rdatamining.com/examples/outlier-detection>
- [635] [http://www.cs.ucr.edu/~eamonn/time\\_series\\_data/](http://www.cs.ucr.edu/~eamonn/time_series_data/)
- [636] <http://www.cs.ucr.edu/~eamonn/SAX.htm>
- [637] <http://www-03.ibm.com/software/products/en/network-ips>
- [638] <http://www.ibm.com/software/analytics/spss>
- [639] <http://www-01.ibm.com/software/analytics/spss/products/statistics/>
- [640] <http://www.sas.com/>
- [641] <http://www.sas.com/software/security-intelligence/index.html>

[642] <http://www.oracle.com/technetwork/database/options/advanced-analytics/odm/index.html>

[643] <http://www.wizsoft.com>

---

---

# Index

---

---

- Activation Function of Neural Network, 99  
Active Learning, 26, 236  
Adaboost, 230  
AdaCost, 230  
Adaptive Re-sampling, 228  
Aggregate Change Points, 301  
Aggregate Statistical Similarity, 256  
ALSO, 239  
Angle-based Outlier Detection, 49  
Applications of Outlier Analysis, 399  
Apriori, 166  
AR Models, 276  
Arbitrarily Oriented Subspaces, 170  
ARIMA Model, 279  
ARMA Model, 278  
Astronomical Applications, 416  
Autoencoder, 102  
Autoregression Integrated Moving Average, 279  
Autoregressive Models for Spatial Data, 352  
Autoregressive Models for Time Series, 276  
Autoregressive Moving Average, 278  
Aviation Safety, 416
- Baum-Welch Algorithm, 335  
Bayes Classifier, 225  
Bayesian Network, 62  
Behavioral Attribute, 345  
Binary Data Outliers, 260  
Binary Labels, 2  
Biological Sequences, 311, 416  
Boosting for Rare Class Classification, 230  
Box and Whisker Diagrams, 45  
Box Plots, 45
- Categorical Outlier Detection, 249  
Categorization of Outlier Models, 10  
CBLOF, 145  
Cell-based Methods, 122  
Central Limit Theorem, 42  
Centroid Distance Signature, 356  
Change Analysis, 273  
Change Detection, 273  
Chebychev Inequality, 38  
Chernoff Bound (Lower Tail), 39  
Chernoff Bound (Upper Tail), 41  
Class Imbalance, 219, 221  
CLUSEQ, 327  
Clustering for Outlier Analysis, 112  
COF, 145  
Collective Outlier, 22  
Combination Outliers, 320  
Combination Outliers in Sequences, 312  
Combining Novel and Rare Class Detection, 234  
Complex Sequence Outliers, 338  
Compression-based Dissimilarity Measure, 325  
Conditional Outlier, 22  
Contextual Attribute, 345  
Contextual Outlier, 22  
Contextual Similarity in Categorical Data, 257  
Cosine Similarity, 268  
Cost-Sensitive Learning, 223  
Covariance Matrix Diagonalization, 75  
Credit-Card Fraud, 404
- Data Cleaning, 415  
Data Types, 21

- Data-Dependent Similarity, 128, 257
- Decision Trees, 226
- Dependent Variable Regression Analysis, 70
- Depth-based Outliers, 47
- Deviation-based Outliers, 48
- DFT, 288
- Differential Graph, 390
- Dimensionality Reduction, 13
- Discrete Attribute Outlier Detection, 249
- Discrete Fourier Transform, 288
- Discrete Sequence Outlier Detection, 24, 311
- Discrete Wavelet Transform, 288
- Disease Outbreaks, 416
- Distance Distribution-based Outliers, 51
- Distance-based Outliers, 118
- Distance-based Sequence Outliers, 324
- Distance-based Subspace Outlier Detection, 152
- Distribution Change, 304
- Distribution-based Outlier Modeling, 54
- Disturbance Events in Ecosystems, 416
- Dropout Training of Deep Networks, 106
- DWT, 288
- Dynamic Programming in HMM, 335
- Dynamic Time Warping, 293
  
- Early Discrete Sequence Anomalies, 340
- Early Termination Trick, 125
- Edit Distance, 325
- Egonet, 372
- EM Algorithm for Categorical Data, 250
- EM Algorithm for Continuous Data, 54
- ENetClus, 388
- Ensemble-based Outlier Detection, 19, 32
- Ensembles for Rare Classes, 229
- ERC-Forest, 144, 164, 182
- Eskin Measure, 256
- Evaluating Outlier Detection, 26
- Events in Social Streams, 412
- Evolutionary Network Analysis, 370
- Evolving Blogs, 413
- Evolving Social Networks, 413
- Explaining Sequence Anomalies, 334
- Exploratory Analysis, 418
- External Validity Measures, 27
- Extreme-Value Analysis, 11, 37, 46
  
- Failure Management of Computer Clusters, 416
  
- Fault Detection, 402
- Feature Bagging, 157, 158
- Feature Engineering, 235
- Feature Selection, 10
- Financial Applications, 404
- Financial Interaction Networks, 406
- Finite State Automaton, 316
- First Story Detection, 269, 411
- Flow Anomalies, 367
- FocusCO Algorithm, 395
- Forward Algorithm, 334
- Forward-backward Algorithm, 335
- FP-Outlier, 261
- Frequency-based Sequence Outliers, 327
- Frequent Pattern Mining Methods, 260
  
- GASP, 244
- Generalized Subspaces, 170
- Generative Models, 54
- Geometric Subsampling, 210
- Goodall Measure, 256
- Grammar Correction, 320
- Graph Evolution Rules, 396
- Graph Kernel, 372
- Graph Outliers, 369
- Graph-based Spatial Outliers, 351
- GraphScope, 390
- Grid-based Projected Outliers, 153
  
- Haar Wavelet, 288
- Half-Space Trees, 164, 182
- Harmonic  $k$ -Nearest Neighbor Detector, 120
- Heterogeneous Markov Random Field, 384
- HiCS, 164
- Hidden Markov Models, 313, 329
- Hidden Variables, 282
- High-Contrast Subspaces, 164
- High-Dimensional Outliers, 17, 149
- Histograms, 137, 160
- HMM, 329
- HMM Design Choices, 331
- Hoeffding Inequality, 41
- HOS-Miner, 157
- Host-based Intrusion Detection, 408
- HOTSAX, 293, 295
- Hotspot in Network, 385
  
- IBM SPSS, 421
- Image Anomalies, 416

- Incomplete Data-based Outliers, 96
- Independent Ensembles, 20
- Indexing for Distance-based Outliers, 126
- INFLO, 145
- Information Theoretic Measures, 48, 261, 307, 325, 339, 343, 381
- Information-Theoretic Models, 16
- Infrequently Recurring Classes, 306
- Inlier, 2
- Insider Trading Detection, 405
- Intensional Knowledge, 130
- Intensional Knowledge of Distance-based Outliers, 130
- Internal Validity Measures, 27
- Intrusion Detection, 305
- Intrusion Detection Applications, 407
- Inverse Document Frequency, 256, 268
- Inverse Occurrence Frequency, 256
- Isolation Forests, 129, 161, 203, 420
- ISOMAP, 86
- Isomorphism, 371
- Iterated Contextual Distance, 257
  
- KDD Nuggets, 421
- Kernel Density Estimation, 138
- Kernel PCA, 85
- Kernel Trick, 85
- Kernel Whitening, 84
- Kolmogorov Complexity, 17, 343
- Kurtosis, 10, 163
  
- Land Cover Anomalies, 414
- Latent Dirichlet Allocation, 268
- Latent Semantic Analysis, 21, 264
- Latent Semantic Indexing, 80
- Law Enforcement, 2
- LDA, 268
- LDOF, 133
- Leaky Bucket, 326
- LFC, 326
- Linear Models for Categorical Data, 254
- Linear Regression Models, 68
- Linkage Outliers, 5
- Linking Criminal Incidents, 416
- Local Correlation Integral, 135
- Local Outlier Factor, 132
- Local Spatial Outliers, 145, 350
- Local Subspace Selection, 166
- Locality Frame Count, 326
- LOCI, 135
- LOCI Plot, 136
- LODA, 141, 217
- LOF, 132
- LSA, 264
- LSA and PCA Relationship, 265
  
- MA Model, 278
- Mahalanobis Distance, 52, 114
- Malicious URL Detection, 416
- Market Basket Outliers, 260
- Markov Inequality, 37
- Markovian Models, 316
- Matrix Factorization, 95, 109, 374, 392
- MDEF, 135
- Medical Applications, 410
- Medical Imaging Diagnostics, 411
- Medical Sensor Diagnostics, 410
- MetaCost, 223
- Minimum Bounding Rectangles, 126
- Minimum Description Length, 339, 343, 381
- Mixed Attribute Outlier Detection, 249
- Mixture Modeling, 54
- Mobile Phone Fraud, 406
- Movement Pattern Outliers, 415
- Moving Average Model, 278
- Multidimensional Change Points, 301
- Multidimensional Spatial Outliers, 350
- Multiple Time Series Models, 279
- Multiple-Proclus, 159
- Multivariate Discrete Sequences, 338
- Multivariate Spatial Outliers, 351
- MUSCLES, 281
  
- NDCG, 30
- Nearest Neighbor Classifier, 225
- Neighborhood-based Spatial Outliers, 349
- NetClus, 388
- Network Intrusion Detection, 409
- Network Outliers, 25, 369
- Neural Networks, 98
- Noise Correction with PCA, 80
- Noise vs Anomaly, 3
- Non-negative Matrix Factorization, 395
- Normalization for PCA, 80
- Normalized Discounted Cumulative Gain, 30
- Novel Class Detection, 232
- Novelties in Temporal Transactions, 262
- Novelty Detection, 262, 273

- ODIN, 129
- One Class Learning, 88, 98, 233, 298
- One Class Neural Networks, 98
- One Class Novelty Detection, 88, 234, 298
- One Class SVM, 88, 233, 298
- One Class SVM for Novelty Detection, 88, 298
- Online Discrete Sequence Anomalies, 340
- Online Novelty Detection, 234, 269, 300, 305
- Open Source Software, 421
- Oracle Data Miner, 421
- Outlier Detection and Description, 10
- Outlier Ensembles, 18, 185
- Outlier Scores, 2
- Outliers from Small Graphs, 371
- OutRank, 159
- OUTRES, 167
- Overlap Measure, 255
  
- Parzen-Rosenblatt Density Estimator, 139
- PCA for Categorical Data, 254
- Perceptron, 99
- PLSA, 267
- Pocket Plots, 348
- Pooled Active Learning, 236
- Position Outliers, 312, 313
- Positive Unlabeled Classification, 231
- PPCA, 109
- PR Curve, 28
- Precision-Recall Curve, 28
- Pretraining Deep Networks, 104
- Primitive Sequence Anomaly, 322
- Principal Component Analysis, 13, 282
- Probabilistic and Statistical Models, 12
- Probabilistic Latent Semantic Analysis, 267
- Probabilistic Models for Categorical Data, 250
- Probabilistic Models for Mixed Data, 253
- Probabilistic Outlier Modeling, 54
- Probabilistic PCA, 109
- Probabilistic Suffix Trees, 318
- Projected Outlier Detection, 149, 151
- Projected Outliers, 153
- Proximity Models for Categorical Data, 255
- Proximity Models for Mixed Data, 259
- Proximity-based Classifiers, 225
- PST, 318
- PUC, 231
  
- Quality Control, 39, 401
- Quality Control Applications, 401
- Query by Committee, 238
  
- Random Forests, 161
- Random Subspace Ensembles, 153, 158
- Randomized Feature Weighting, 210
- Ranking Subspace Outliers, 153
- Rare Class Detection, 221
- Receiver Operating Characteristic, 28
- Recommender System Outliers, 96
- Regime Anomalies in Time Series, 308
- Regression Model, 7
- Regression Modeling with Dependent Variables, 70
- Relabeling, 223
- Replicator Neural Networks, 99, 102
- Reservoir Sampling, 386
- Reverse Nearest Neighbor, 129
- RIPPER, 316
- ROAM, 365
- ROC Curve, 28
- RODS, 109
- ROF, 144
- Rotated Bagging, 175
- Rotated Subspace Sampling, 175
- RS-Hash, 138, 141, 146, 160, 182, 217, 309, 420
- RS-Stream, 161, 309
- Rule-based Classifiers, 226
- Rule-based Models for Position Outliers, 315
  
- Sampling for Distance-based Outliers, 124
- SAS, 421
- SAS Security Intelligence, 421
- SAX, 291
- Sea Surface Temperature Anomalies, 413
- SELECTIVE MUSCLES, 281
- Semi-supervised Outlier Detection, 232
- Sequence Classification, 340
- Sequence Outlier Detection, 311
- Sequential Ensembles, 19
- Set-based Sequences, 339
- Shape Change Detection in Spatial Data, 361
- Shared Nearest Neighbor Similarity, 128
- Short Memory Property, 315
- Shortest Path Distance Changes, 392
- Similarity in Categorical Data, 255
- Similarity in Mixed Data, 259



- Simple Matching Coefficient, 324
- SLOM, 145, 350
- SMOTE, 229
- SMOTEBoost, 231
- SMT, 318
- Social Media Applications, 411
- Social Stream Evolution, 271, 396
- SOD, 169
- Software Resources, 421
- Space-filling Curves, 143
- Spam Filtering, 412
- Spam Link Detection, 412
- Sparse Coding, 109
- Sparse Markov Transducers, 318
- Spatial Heteroscedasticity, 348
- Spatial Autocorrelations, 348
- Spatial Outlier Detection, 345
- Spectral Methods, 14, 86, 108, 116, 378, 395
- SPIRIT, 282
- SPOT Algorithm, 301
- Stacking, 18, 236
- Statistical Tail Confidence Tests, 43
- Stock Market Anomalies, 405
- STORM Algorithm, 299
- Streaming Novel Class Detection, 306
- Streaming Novelty Detection, 234, 269, 300, 305
- Streaming Outlier Detection, 22, 182, 273, 298, 309
- Streaming Rare Class Detection, 305
- Strong Outliers, 4
- Structural Defect Detection, 403
- Structural Reservoir Sampling, 386
- Student's t-distribution, 43
- SUBDUE, 381
- Subgraph Outliers, 381
- Subsampling, 206
- Subspace Ensembles, 153
- Subspace Histograms, 160
- Subspace Methods for Transaction Data, 260
- Subspace Outlier Degree, 169
- Subspace Outlier Detection, 18, 149, 151
- Supervised Outlier Detection, 25, 219
- Supervised Sequence Outliers, 340
- Supervised Shape Anomalies in Time Series, 298
- Supervised Shape Discovery in Spatial Data, 360
- Support Vector Data Description, 93
- SVDD, 93
- SVM Classifier, 88, 226
- Symbolic Aggregate Approximation, 291
- Synthetic Over-sampling, 229
- Systems Diagnosis, 402
- t-Distribution, 43
- t-value Test, 43
- Tail Confidence Tests, 43
- Tail Inequalities, 37
- TARZAN, 328
- Temporal Description Length, 339
- Temporal Graph Outliers, 384
- Temporal Outlier Detection, 273
- Temporal Smoothness in Networks, 392
- Text Applications, 411
- Text Outliers, 262
- Time Series Outlier Detection, 22
- Top- $n$  Local Outliers, 145
- Topic Detection and Tracking, 268
- Topic Modeling, 267
- Traffic Anomalies, 415
- Trajectory Outliers, 291, 363, 415
- Transaction Data Outliers, 260
- TRINITY, 421
- TROAD, 293, 363
- Tukey Box Plot, 46
- Unsupervised Regression Modeling, 74
- Unusual Shapes of Time-Series, 286
- Variable Subsampling, 207
- Variable Subsampling with Rotated Bagging, 209
- Variogram Clouds, 353
- Velocity Density Estimation, 302
- Viterbi Algorithm, 334
- VR, 209
- Wagging, 210
- Weak Outliers, 4
- Web Log Analytics, 406
- Weighting for Supervision, 225
- Whole Sequence Anomaly, 323
- Wilcoxon Test, 304
- WizRule, 421
- Z-value test, 6