MORGAN & CLAYPOOL PUBLISHERS

Outlier Detection for Temporal Data

Manish Gupta
Jing Gao
Charu Aggarwal
Jiawei Han

SYNTHESIS LECTURES ON DATA MINING AND KNOWLEDGE DISCOVERY

Jiawei Han, Lise Getoor, Wei Wang, Johannes Gehrke, Robert Grossman, *Series Editors*

Outlier Detection for Temporal Data

Synthesis Lectures on Data Mining and Knowledge Discovery

Outlier Detection for Temporal Data

Manish Gupta, Jing Gao, Charu Aggarwal, and Jiawei Han

www.morganclaypool.com

ISBN: 9781627053754 paperback
ISBN: 9781627053761 ebook

DOI 10.2200/S00573ED1V01Y201403DMK008

A Publication in the Morgan & Claypool Publishers series
SYNTHESIS LECTURES ON DATA MINING AND KNOWLEDGE DISCOVERY

Lecture #8
Series Editors: Jiawei Han, *University of Illinois at Urbana-Champaign*
Lise Getoor, *University of Maryland*
Wei Wang, *University of North Carolina, Chapel Hill*
Johannes Gehrke, *Cornell University*
Robert Grossman, *University of Chicago*
Series ISSN
Print 2151-0067 Electronic 2151-0075

Outlier Detection for Temporal Data

Manish Gupta
Microsoft, India and International Institute of Technology–Hyderabad, India

Jing Gao
State University of New York, Buffalo, NY

Charu Aggarwal
IBM T. J. Watson Research Center, NY

Jiawei Han
University of Illinois at Urbana-Champaign, IL

SYNTHESIS LECTURES ON DATA MINING AND KNOWLEDGE DISCOVERY #8

MORGAN & CLAYPOOL PUBLISHERS

ABSTRACT

Outlier (or anomaly) detection is a very broad field which has been studied in the context of a large number of research areas like statistics, data mining, sensor networks, environmental science, distributed systems, spatio-temporal mining, etc. Initial research in outlier detection focused on time series-based outliers (in statistics). Since then, outlier detection has been studied on a large variety of data types including high-dimensional data, uncertain data, stream data, network data, time series data, spatial data, and spatio-temporal data. While there have been many tutorials and surveys for general outlier detection, we focus on outlier detection for temporal data in this book.

A large number of applications generate temporal datasets. For example, in our everyday life, various kinds of records like credit, personnel, financial, judicial, medical, etc., are all temporal. This stresses the need for an organized and detailed study of outliers with respect to such temporal data. In the past decade, there has been a lot of research on various forms of temporal data including consecutive data snapshots, series of data snapshots and data streams. Besides the initial work on time series, researchers have focused on rich forms of data including multiple data streams, spatio-temporal data, network data, community distribution data, etc.

Compared to general outlier detection, techniques for temporal outlier detection are very different. In this book, we will present an organized picture of both recent and past research in temporal outlier detection. We start with the basics and then ramp up the reader to the main ideas in state-of-the-art outlier detection techniques. We motivate the importance of temporal outlier detection and brief the challenges beyond usual outlier detection. Then, we list down a taxonomy of proposed techniques for temporal outlier detection. Such techniques broadly include statistical techniques (like AR models, Markov models, histograms, neural networks), distance- and density-based approaches, grouping-based approaches (clustering, community detection), network-based approaches, and spatio-temporal outlier detection approaches. We summarize by presenting a wide collection of applications where temporal outlier detection techniques have been applied to discover interesting outliers.

KEYWORDS

temporal outlier detection, time series data, data streams, distributed data streams, temporal networks, spatiotemporal outliers

To my dear parents, Satyapal Gupta and Madhubala Gupta,
and my cute loving wife Nidhi

–Manish Gupta

To my husband Lu,
and my parents

–Jing Gao

To my wife Lata,
and my daughter Sayani

–Charu Aggarwal

To my wife Dora,
and my son Lawrence

–Jiawei Han

Contents

Preface

Temporal data is omnipresent and growing rapidly. Given such huge amounts of temporal data, an important task is to find surprising instances efficiently. Recently, many effective and efficient temporal anomaly detection techniques have been proposed in a variety of research disciplines including data mining, sensor networks, environmental science, distributed systems, spatio-temporal mining, etc. Although there have been multiple surveys and books on general outlier detection, there is no single survey or book dedicated to a thorough study of the diverse techniques and extensive studies in temporal outlier detection. We believe that an organized and extensive coverage of work from multiple disciplines in this book will greatly benefit researchers in these disciplines and motivate cross fertilization of ideas. We begin by motivating the importance of temporal outlier detection and briefing the challenges beyond usual outlier detection. Then, we list down a taxonomy of proposed techniques for temporal outlier detection. For each temporal data type, we will list down several interesting outlier definitions and present various approaches for efficient and effective detection of such outliers. We summarize by presenting a collection of applications where temporal outlier detection techniques have been applied to discover interesting outliers.

SCOPE OF THIS BOOK

This book covers outlier detection techniques for temporal data popular in the data mining community. Many techniques have also been developed in the statistics community and we will not cover them. Specifically, we will discuss techniques for time series data, data streams, distributed data streams, network data, and spatio-temporal data. We will not cover novelty detection techniques.

DEVELOPMENT OF THE BOOK

Many tutorials dedicated to outlier detection were conducted by researchers in data mining, sensor networks, communication networks, and distributed systems communities. Outlier detection is so popular and useful for industry that many tools have been built for efficient outlier detection. For example, the package “outliers” in R, RapidMiner, Oracle, etc. Besides these, many workshops also focused on the general area of outlier detection. However, all of these events focused on the general area of outlier detection; none of these have focused specifically on temporal outlier detection.

This book is based on three tutorials offered by the authors at CIKM 2013, SDM 2013, and ASONAM 2013. A short version of this book also appeared as a survey paper recently published at TKDE 2014. The networks part of this book draws significant amount of material from the Ph.D. thesis of the first author.

AUDIENCE

This book is mainly targeted at researchers and practitioners in knowledge management, data mining, distributed systems, and sensor networks. While the audience with a good background on data mining would benefit most from this book, we believe the material would give a general audience and newcomers a complete picture of the current work, introduce important research topics in this field, and inspire them to learn more.

Manish Gupta, Jing Gao, Charu Aggarwal, and Jiawei Han
March 2014

Acknowledgments

The work was supported in part by the U.S. Army Research Laboratory under Cooperative Agreement No. W911NF-11-2-0086 (Cyber-Security) and W911NF-09-2-0053 (NS-CTA), the U.S. Army Research Office under Cooperative Agreement No. W911NF-13-1-0193, and U.S. National Science Foundation grants CNS-0931975, IIS-1017362, and IIS-1320617. The views and conclusions contained in our research publications are those of the authors and should not be interpreted as representing any funding agencies. The support is gratefully acknowledged.

We thank Hui Xiong, Leman Akoglu, and Hanghang Tong for their detailed reviews. We thank Diane Cerra, C.L. Tondo, Sara Kreisman, and other members of the Morgan & Claypool team for their patience and their superb production job. Finally, we wish to thank our families, for their constant support and encouragement.

Manish Gupta, Jing Gao, Charu Aggarwal, and Jiawei Han
March 2014

Figure Credits

Figure 3.4	based on Angiulli, F. and Fassetti, F. (2007). Detecting Distance-based Outliers in Streams of Data. *Proceeding CIKM '07 Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 811-820. Copyright © 2007, Association for Computing Machinery, Inc. DOI: 10.1145/1321440.1321552
Figure 4.2	based on Subramaniam, et al: (2006). Online Outlier Detection in Sensor Data using Non-parametric Models. *Proceeding VLDB '06 Proceedings of the 32nd international conference on Very large data bases*, pages 187-198. Copyright © 2006, Very Large Data Base Endowment Inc.
Figures 7.1, 7.2	based on Hill, D. J. and Minsker, B. S. (2010). Anomaly Detection in Streaming Environmental Sensor Data: A Data-driven Modeling Approach. Environmental Modelling and Software, 25(9): 1014–1022. Copyright © 2010 Published by Elsevier Ltd. DOI: 10.1016/j.envsoft.2009.08.010
Figures 7.3, 7.4	based on Birant, D. and Kut, A. (2006). Spatio-Temporal Outlier Detection in Large Databases. *Journal of Computing and Information Technology (CIT)*, 14(4), pages 291-297. CIT. Journal of Computing and Information Technology is an open access journal.
Figure 7.5	from Cheng, T. and Li, Z. (2006). A Multiscale Approach for Spatio-Temporal Outlier Detection. *Transactions in GIS*, Volume 10, Issue 2, pages 253–263, March 2006. Copyright © 2006 John Wiley & Sons, Inc. DOI: 10.1111/j.1467-9671.2006.00256.x
Figure 7.6	from Lasaponara, R. (2005). On the use of principal component analysis (PCA) for evaluating interannual vegetation anomalies from SPOT/VEGETATION NDVI temporal series. *Ecological Modelling*, Volume 194, Issue 4, 15 April 2006, pages 429–434. Copyright © 2005 Elsevier B.V. Reprinted by permission. DOI: 10.1016/j.ecolmodel.2005.10.035

Figure 7.7	based on Lu, C.-T. and Liang, L. (2004). Wavelet fuzzy classification for detecting and tracking region outliers in meteorological data. *Proceeding GIS '04 Proceedings of the 12th annual ACM international workshop on Geographic information systems*, pages 258-265. Copyright © 2004, Association for Computing Machinery, Inc. DOI: 10.1145/1032222.1032260
Figures 7.8, 7.9	based on Dasgupta, D. and Forrest, S. (1996). Novelty Detection in Time Series Data using Ideas from Immunology. *Proceedings of the 5th International Conference on Intelligent Systems*.
Figures 7.10, 7.11, 7.12, 7.13	from Ge, et al: (2010). Top-Eye: Top-K Evolving Trajectory Outlier Detection. *Proceeding CIKM '10 Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1733-1736. Copyright © 2010, Association for Computing Machinery, Inc. Reprinted by permission. DOI: 10.1145/1871437.1871716
Figure 7.14, 7.15	from Keogh, et al: (2005). HOT SAX: Efficiently Finding the Most Unusual Time Series Subsequence. *Fifth IEEE International Conference on Data Mining*, pages 226-233. Copyright © 2005 IEEE. Used with permission. DOI: 10.1109/ICDM.2005.79
Figure 7.15	based on Keogh, et al: (2005). HOT SAX: Efficiently Finding the Most Unusual Time Series Subsequence. *Fifth IEEE International Conference on Data Mining*, pages 226-233. Copyright © 2005 IEEE. DOI: 10.1109/ICDM.2005.79
Figure 7.16	based on Wei, et al: (2006). SAXually Explicit Images: Finding Unusual Shapes. *Sixth International Conference on Data Mining, 2006. ICDM '06*, pages 711-720. Copyright © 2006 IEEE. DOI: 10.1109/ICDM.2006.138
Figure 7.17	from Wei, et al: (2006). SAXually Explicit Images: Finding Unusual Shapes. *Sixth International Conference on Data Mining, 2006. ICDM '06*, pages 711-720. Copyright © 2006 IEEE. Used with permission. DOI: 10.1109/ICDM.2006.138
Figure 7.19	based on Gupta, et al: (2012). Community Trend Outlier Detection Using Soft Temporal Pattern Mining. *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2012, Bristol, UK, September 24-28, 2012. Proceedings, Part II*, pp 692-708. Copyright © 2012, Springer-Verlag Berlin Heidelberg. DOI: 10.1007/978-3-642-33486-3_44

CHAPTER 1

Introduction and Challenges

Outlier detection is a broad field that has been studied in the context of a large number of application domains. [Aggarwal, 2013], [Chandola et al., 2009], [Hodge and Austin, 2004], and [Zhang et al., 2008] provide an extensive overview of outlier detection techniques. Outlier detection is also referred to as anomaly detection, event detection, novelty detection, deviant discovery, change point detection, fault detection, intrusion detection, or misuse detection. The three main types of outliers studied in the literature are point outliers, contextual outliers, and collective outliers. Point outliers are generally studied in the context of multidimensional data types, whereas contextual outliers are studied in dependency-oriented data types such as time-series, discrete sequences, spatial data, and graphs. Clearly, the choice of the data type has a significant impact on the methodology used for outlier analysis. A variety of supervised, semi-supervised and unsupervised techniques have been used for outlier detection. These include classification-based, clustering-based, nearest neighbor-based, density-based, statistical, information theory-based, spectral decomposition-based, visualization-based, depth-based, and signal processing-based techniques. Outlier detection has been studied in a variety of data domains including high-dimensional data [Aggarwal and Yu, 2001], uncertain data [Aggarwal and Yu, 2008], streaming data [Aggarwal and Subbian, 2012], [Aggarwal, 2005a], [Aggarwal et al., 2011], network data [Aggarwal et al., 2011], [Gao et al., 2010], [Ghoting et al., 2004], [Gupta et al., 2012a], [Gupta et al., 2012b], and time series data [Burman and Otto, 1988], [Fox, 1972]. Outlier detection has been used extensively for intrusion detection, fraud detection, fault detection, system health monitoring, event detection in sensor networks, and detecting eco-system disturbances. Outlier detection is very popular in industrial applications, and therefore many software tools have been built for efficient outlier detection, such as R (packages "outliers"[1] and "outlierD" [Cho et al., 2008]), SAS,[2] RapidMiner,[3] and Oracle datamine.[4]

Different kinds of data, such as credit, personnel, financial, judicial, medical, and web usage data are temporal. Social network data streams, astronomy data, sensor data, computer network traffic, and commercial transactions are all examples of massive amounts of temporal data. As a result, over time, besides time series, a large variety of temporal datasets have become quite popular. These include temporal networks, temporal databases, data streams, distributed data streams, and spatio-temporal data. The quest to mine information from these new forms of temporal data

[1]`http://cran.r-project.org/web/packages/outliers/outliers.pdf`
[2]`http://www.nesug.org/Proceedings/nesug10/ad/ad07.pdf`
[3]`http://www.youtube.com/watch?v=C1KNb1Kw-As`
[4]`http://docs.oracle.com/cd/B28359_01/datamine.111/b28129/anomalies.htm`

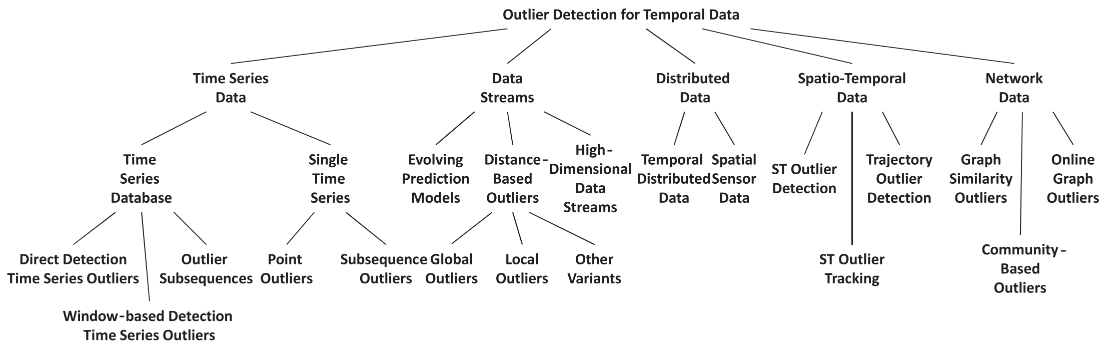

Figure 1.1: Organization of the book.

have led to the rise of the field of temporal data mining [Mitsa, 2010] which includes tasks such as temporal data similarity computation, representation, and summarization, temporal data classification and clustering, prediction, temporal pattern discovery, spatio-temporal data mining, and outlier detection for temporal data. This book focuses on the aspect of outlier detection. A short version of this book has also been published recently [Gupta et al., 2014a] as a journal survey paper. Readers are referred to this survey for a brief overview of the topic.

The different data domains in outlier analysis typically require dedicated techniques of different types. Temporal outlier analysis typically examines anomalies in the behavior of the data *across time*. Therefore the outlierness of a data point can only be understood in the context of the temporal *changes* in the data values or patterns.

1.1 TEMPORAL OUTLIER EXAMPLES

Some real-life examples of outliers in temporal data are as follows.

- *Financial Markets:* An abrupt change in the stock market, or an unusual pattern within a specific window such as the *flash crash* of May 6, 2010, is an anomalous event which needs to be detected early in order to avoid and prevent extensive disruption of markets because of possible weaknesses in trading systems.

- *System Diagnosis:* A significant amount of data generated about the *system state* is discrete in nature. This 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. Anomalies provide information about potentially threatening and failure events in such systems.

- *Biological Data:* While biological data is not temporal in nature, the placement of individual amino-acids is analogous to positions in temporal sequences. Therefore, temporal methods can be directly used for biological data.

- *User-action Sequences:* A variety of sequences abound in daily life that are created by user actions in different domains. These include web browsing patterns, customer transactions, or RFID sequences. Anomalies provide an idea of user-behavior which is deviant for specific reasons (e.g., an attempt to crack a password will contain a sequence of *login* and *password* actions).

This broad diversity in applications is also reflected in the diverse formulations and data types relevant to outlier detection. A common characteristic of all temporal outlier analysis is that *temporal continuity* plays a key role in all these formulations, and unusual *changes*, *sequences*, or *temporal patterns* in the data are used in order to model outliers. In this sense, time forms the *contextual variable* with respect to which all analysis is performed. Temporal outlier analysis is closely related to change point detection, and event detection, since these problems represent two instantiations of a much broader field. The problem of *forecasting* is closely related to many forms of temporal outlier analysis, since outliers are often defined as deviations from *expected* values (or *forecasts*). Nevertheless, while forecasting is a useful *tool* for many forms of outlier analysis, the broader area seems to be much richer and multi-faceted.

1.2 DIFFERENT FACETS OF TEMPORAL OUTLIER ANALYSIS

Outlier analysis problems in temporal data may be categorized in a wide variety of ways that represent different facets of the analysis. The area is so rich that no single type of abstract categorization can fully capture the complexity of the problems in the area, since these different facets may be present in an arbitrary combination. Some of these facets are as follows.

- *Time-series vs. Multidimensional Data Facet:* In time-series data (e.g., sensor readings) the importance of temporal continuity is paramount, and all analysis is performed with careful use of reasonably small windows of time (the contextual variable). On the other hand, in a multi-dimensional data stream such as a text newswire stream, an application such as *first-story detection*, might not rely heavily on the temporal aspect, and thus the methods are much closer to standard multi-dimensional outlier analysis.

- *The Point vs. Window Facet:* Are we looking for an unusual data *point* in a temporal series (e.g., sudden jump in heart rate in ECG reading), or are we looking for an unusual pattern of changes (contiguous ECG pattern indicative of arrythmia)? The latter scenario is usually far more challenging than the former. Even in the context of a multi-dimensional data stream, a single point deviant (e.g., first story in a newswire stream) may be considered a different kind of outlier than an aggregate change point (e.g., sudden change in the aggregate distribution of stories over successive windows).

- *The Data Type Facet:* Different kinds of data such as continuous series (e.g., sensors), discrete series (e.g., web logs), multi-dimensional streams (e.g., text streams), or network data (e.g., graph and social streams) require different kinds of dedicated methods for analysis.

- *The Supervision Facet:* Are previous examples of anomalies available? This facet is of course common to all forms of outlier analysis, and is not specific to the temporal scenario.

These different facets are largely independent of one another, and a large number of problem formulations are possible with the use of a combination of these different facets. Therefore, this book is largely organized by the facet of data type, and examines different kinds of scenarios along this broad organization.

1.3 SPECIFIC CHALLENGES FOR OUTLIER DETECTION FOR TEMPORAL DATA

Compared to the other data mining tasks like classification and clustering, outlier detection presents its unique challenges as follows.

- Classification makes use of available labeled data to learn a classifier model which can then be used to classify future data points. Outlier detection is an unsupervised technique. Outlier detection techniques need to learn similarities between data points without using any user supplied label. Those data points which are very different from others are then marked as outliers. Outlier detection for temporal data aims at identifying anomalous behavior across time. Thus, due to its unsupervised nature, outlier detection becomes challenging.

- Clustering is closely related to outlier detection. Clustering is also an unsupervised technique like outlier detection. Clustering aims at grouping similar objects. Often times, objects that cannot be assigned to any of the clusters are interesting but ignored by the clustering process. These points could be errors in data, noise, or surprising and therefore interesting data points. Outlier detection aims at identifying such surprising data points and is therefore challenging. Temporal clustering aims at maintaining cluster information across time. Outlier detection for temporal data becomes more challenging because it needs to identify data points with surprising combination of temporal properties.

While temporal outlier detection aims to find rare and interesting instances, as in the case of traditional outlier detection, new challenges arise due to the nature of temporal data. We list them below.

- A wide variety of anomaly models are possible depending upon the specific data type and scenario. For example, even though discrete sequences can be viewed as categorical versions of time series, the methods for finding anomalies are quite different in these cases. In temporal graphs, structural patterns need to be accounted for in anomaly detection. This leads to diverse formulations that need to be designed for the specific problem. For arbitrary applications, it may often not be possible to use off-the-shelf models, because of the wide variations in problem formulations. This is one of the motivating reasons for this book to provide an overview of the most common combinations of facets explored in temporal outlier analysis.

- Since new data arrives at every time instant, the scale of the data is very large. This often leads to processing and resource-constraint challenges. In the streaming scenario, only a single scan is allowed. Traditional outlier detection is much easier, since it is typically an offline task. Unlike the static case, in the dynamic case, an outlier detection system is expected to flag new time points as anomalous or not in real-time. Real-time detection and decision making is a major challenge.

- Outlier detection for temporal data in distributed scenarios poses significant challenges of minimizing communication overhead and computational load in resource-constrained environments.

1.4 CONCLUSIONS AND SUMMARY

Outlier detection is a broad field that has been studied in the context of a large number of application domains. More recently, there has been a large focus on analysis of temporal data and hence a large number of mechanisms for outlier detection for temporal data have been developed. Temporal outliers exist in abundance in real life. Temporal outlier detection is more challenging than the traditional outlier detection because of the scale and the online processing requirement. In this short book, we aim to provide a comprehensive and structured overview of outlier detection techniques for temporal data. Figure 1.1 shows the organization of the book with respect to the data type facet. For each data type, we discuss specific problem classes in various sections. We begin with outlier detection for the easiest scenario for temporal data—discrete time series data in Chapter 2. However, a lot of data gets sampled over very short time intervals, and keeps flowing in infinitely leading to data streams. We study the techniques for outlier detection in streams in Chapter 3. Often times, data is distributed across multiple locations. We study how to extract global outliers in such distributed scenarios in Chapter 4. For some applications like environmental data analysis, data is available over a continuum of both space and time dimensions. We provide an overview of techniques to handle such data in Chapter 5. Finally, networks can capture very rich semantics for almost every domain. Hence, we discuss outlier detection mechanisms for network data in Chapter 6. We also present a few applications where such temporal outlier detection techniques have been successfully employed in Chapter 7. The conclusions are presented in Chapter 8.

CHAPTER 2

Outlier Detection for Time Series and Data Sequences

Time-series and discrete sequences correspond to the numeric and categorical representations of temporal data. A significant amount of work has been performed in both these areas, although time-series is the older topic among the two areas. Therefore, a larger number of models are available for continuous time series data. The first work on outlier detection for time series data corresponds to parametric models [Fox, 1972]. Several models were subsequently proposed in the statistics literature, including autoregressive moving average (ARMA), autoregressive integrated moving average (ARIMA), vector autoregression (VARMA), CUmulative SUM Statistics (CUSUM), and exponentially weighted moving average for continuous time series data. For discrete sequence data, Markovian models are the ones that are used most commonly. While a detailed exposition is beyond the scope of this book, we will provide an overview of the key ideas in this topic, especially from a computer science perspective. We direct the reader to [Barnett and Lewis, 1978], [Hawkins, 1980], [Rousseeuw and Leroy, 1987] for further reading from the statistics point of view. In this chapter, we will focus on two main types of outlier detection techniques for time series studied in the data mining community. The first part concerns techniques to detect outliers over a database of *many different* time series or sequences, whereas the second part deals with outlier points or segments within *a single time series or sequence*.

2.1 OUTLIERS IN TIME SERIES DATABASES

Given a time series database, we will discuss methods to identify a few time series sequences as outliers, or to identify a subsequence in a test sequence as an outlier. An outlier score for a time series can be computed directly, or by first computing scores for overlapping fixed size windows and then aggregating them. We discuss these techniques in this section.

2.1.1 DIRECT DETECTION OF OUTLIER TIME SERIES

Given: A database of time series.
Find: All anomalous time series.
Examples: Anomalies from jet engine vibrations [Nairac et al., 1999]; Unix session from an intruder [Szymanski and Zhang, 2004]; general network intrusion attacks [Portnoy et al., 2001];

password guessing, root privileges access through symbolic links, remote unauthorized access attempt [Ye, 2000].

In this problem setting, it is assumed that most of the time series in the database are normal while a few are anomalous. Similar to traditional outlier detection, the usual recipe for solving such problems is to first learn a model based on all the time series sequences in the database, and then compute an outlier score for each sequence with respect to the model. The model could be *supervised* or *unsupervised* depending on the availability of training data.

Unsupervised Discriminative Approaches

Discriminative approaches rely on the definition of a similarity function that measures the similarity between two sequences. Once a similarity function is defined, such approaches cluster the sequences, such that within-cluster similarity is maximized, while between-cluster similarity is minimized. The anomaly score of a test time series sequence is defined as the distance to the centroid (or medoid) of the closest cluster. The primary variation across such approaches, are the choice of the similarity measure, and the clustering mechanism.

Similarity Measures The most popular sequence similarity measures are the simple *match count-based sequence similarity* [Lane et al., 1997], [Lane and Brodley, 1998], and the normalized length of the longest common subsequence (LCS) [Budalakoti et al., 2006], [Budalakoti et al., 2009], [Chandola et al., 2008], [Sequeira and Zaki, 2002]. The advantage of the former is its greater computational efficiency, whereas the latter can adjust to segments in the sequences containing noise, but is more expensive because of its dynamic programming methodology.

Match count-based sequence similarity is defined as follows. For a length l, the similarity between the sequences $X = (x_0, x_1, \ldots, x_{l-1})$ and $Y = (y_0, y_1, \ldots, y_{l-1})$ is defined by the pair of functions:

$$w(X, Y, i) = \begin{cases} 0, & \text{if } i < 0 \text{ or } x_i \neq y_i \\ 1 + w(X, Y, i-1), & \text{if } x_i = y_i \end{cases} \tag{2.1}$$

where $w(X, Y, i) = 0$ for $i < 0$, so that $w(X, Y, 0)$ is well defined when $x_0 = y_0$ and

$$Sim(X, Y) = \sum_{i=0}^{l-1} w(X, Y, i). \tag{2.2}$$

On the other hand, the normalized length of the longest common subsequence (LCS) is computed as follows.

Given two sequences X and Z, Z is a subsequence of X if removing some characters from X will produce Z. Z is a common subsequence of two sequences X and Y if Z is a subsequence of X and Y. The longest such subsequence between X and Y shall be called the longest common subsequence (LCS).

$$nLCS = \frac{length(LCS)}{\sqrt{m.n}} \tag{2.3}$$

where m and n are the lengths of the sequences X and Y, respectively. The problem of finding the LCS can be easily solved using the following dynamic programming formulation:

$$LCS(X_i, Y_j) = \begin{cases} \phi, & \text{if } i = 0 \text{ or } j = 0 \\ LCS(X_{i-1}, Y_{j-1}), & \text{if } x_i = y_j \\ longest(LCS(X_i, Y_{j-1}), LCS(X_{i-1}, Y_j)), & \text{if } x_i \neq y_j \end{cases} . \tag{2.4}$$

To find the longest subsequences common to X_i and Y_j, compare the elements x_i and y_j. If they are equal, then the sequence $LCS(X_{i-1}, Y_{j-1})$ is extended by that element, x_i. If they are not equal, then the longer of the two sequences, $LCS(X_i, Y_{j-1})$, and $LCS(X_{i-1}, Y_j)$, is retained. (If they are both the same length, but not identical, then both are retained.)

Besides these two similarity measures, [Gupta et al., 2013f] propose a similarity measure for multi-variate time series as follows. Consider two multi-variate time series M_a and M_b. Let the time series be defined over time intervals of length l_1 and l_2, respectively. Thus, the multi-variate time series can be represented using matrices ${M_a}^{l_1 \times c}$ and ${M_b}^{l_2 \times c}$, respectively, where c is the number of metric variables. For both the matrices, we can compute the top k principal components (that can capture $\geq$ 95% variance) to obtain smaller subspaces $P^{c \times k}$ and $Q^{c \times k}$, respectively. Similarity between the two multi-variate time series can be expressed in terms of the similarity between their principal components as follows:

$$sim(M_a, M_b) = \frac{trace(P'QQ'P)}{k}. \tag{2.5}$$

Geometrically, this means that the similarity between the two multivariate time series is the average sum of squares of the cosines of the angles between each principal component in P and Q. Thus, we can rewrite Eq. 2.5 as follows:

$$sim(M_a, M_b) = \frac{1}{k} \sum_{i=1}^{k} \sum_{j=1}^{k} \cos^2 \theta_{ij}, \tag{2.6}$$

where θ_{ij} is the angle between the i^{th} and the j^{th} principal components in P and Q, respectively.

Further, Eq. 2.6 can be modified using a weighted average, with the amount of variance explained by each principal component as weights, as follows:

$$sim(M_a, M_b) = \frac{\sum_{i=1}^{k} \sum_{j=1}^{k} \lambda_{Pi} \lambda_{Qj} cos^2 \theta_{ij}}{\sum_{i=1}^{k} \lambda_{Pi} \lambda_{Qi}}, \tag{2.7}$$

where λ_{Pi} is the eigen value corresponding to the i^{th} principal component (eigen vector) in P. Note that the similarity value lies between 0 and 1.

Clustering Methods Popular clustering methods include k-Means [Nairac et al., 1999], Expectation-Maximization (EM) [Pan et al., 2010] phased k-Means [Rebbapragada et al., 2009], dynamic clustering [Sequeira and Zaki, 2002], k-Medoids [Budalakoti et al., 2006], [Budalakoti et al., 2009], single-linkage clustering [Portnoy et al., 2001], clustering of multi-variate time series in the principal components space [Gupta et al., 2013f], one-class SVM [Eskin et al., 2002], [Evangelista et al., 2005], [Ma and Perkins, 2003b], [Szymanski and Zhang, 2004], and self-organizing maps [González and Dasgupta, 2003]. The choice of the clustering method is application-specific, because different clustering methods have different complexity, with varying adaptability to clusters of different numbers, shapes and sizes.

We describe the above clustering methods in brief here. k-Means [Nairac et al., 1999] clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. The iterative algorithm performs two steps per iteration. (1) Assignment: Assign each observation to the closest cluster; and (2) Update: Calculate the new means to be the centroids of the observations in the new clusters. Expectation-maximization (EM) algorithm is an iterative method for finding maximum likelihood or maximum a posteriori (MAP) estimates of parameters. The EM iteration alternates between performing an expectation (E) step, which creates a function for the expectation of the log-likelihood evaluated using the current estimate for the parameters, and a maximization (M) step, which computes parameters maximizing the expected log-likelihood found on the E step. These parameter-estimates are then used to determine the distribution of the latent variables in the next E step. The two steps are very much like the two steps of k-Means; indeed, EM can be considered as a soft k-Means algorithm. Phased k-Means [Rebbapragada et al., 2009] differs from k-Means in that it re-phases each time series prior to similarity calculation and updates centroids from these rephased curves. k-Means has two main drawbacks: (1) random allocation of cluster centers reduces its accuracy and (2) k (number of centers) is hard to set. Dynamic clustering [Sequeira and Zaki, 2002] avoids these drawbacks by growing clusters when needed. In contrast to the k-Means algorithm, k-Medoids [Budalakoti et al., 2006], [Budalakoti et al., 2009] chooses datapoints as centers (medoids or exemplars) and works with an arbitrary matrix of distances between datapoints instead of l_2 norm. This method was proposed in 1987 [Kaufman and Rousseeuw, 1987] for the work with l_1 norm and other distances. Single-linkage clustering [Portnoy et al., 2001] is one of several methods of agglomerative hierarchical clustering. The algorithm starts with an empty set of clusters, and generates the clusters with a single pass through the dataset. For each new data instance retrieved from the normalized training set, it computes the distance between it and each of the centroids of the clusters in the cluster set so far. The cluster with the shortest distance is selected, and if that distance is less than some constant W (cluster width) then the instance is assigned to that cluster. Otherwise, a new cluster is created with the instance as its center. [Gupta et al., 2013f] describe a modified k-Means algorithm where clustering of multi-variate time series is performed. Due to the special nature of the objects to be clustered (multi-variate time series), the assignment and the update steps of the k-Means algorithm need to be appropriately modified. The unsupervised one-class SVM [Eskin et al., 2002, Evangelista et al., 2005, Ma and Perkins, 2003b, Szymanski and Zhang, 2004] attempts to separate the entire set of training data from the origin. This is done by solving a quadratic program that penalizes any points not separated from the origin while simultaneously trying to maximize the distance of this hyperplane from the origin. At the end of this optimization, this hyperplane then acts as the decision function, with those points that it separates from the origin classified as normal, and those which are on the other side of the hyperplane, are classified as abnormal. Figure 2.1 shows a one-class SVM. A self-organizing map (SOM) [González and Dasgupta, 2003] is a type of neural network that uses competitive learning for training. A SOM is able to capture the important features contained in the input space and provides a structural representation that preserves a topological structure. The output neurons of a SOM are organized in a one- or two-dimensional lattice. The weight vectors of these neurons represent prototypes of the input data that can be interpreted as the centroids of clusters of similar samples.

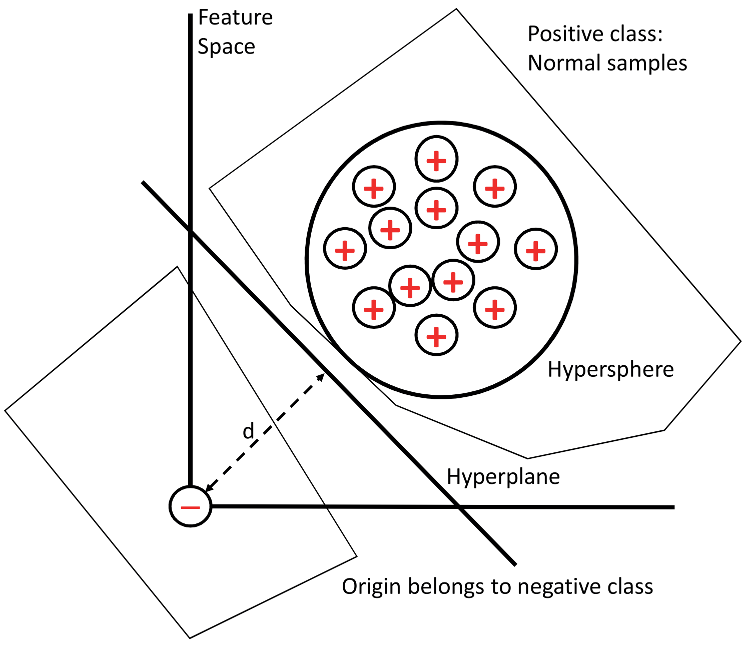
Feature Space
Positive class: Normal samples
Hypersphere
d
Hyperplane
Origin belongs to negative class

Figure 2.1: One-class SVM for anomaly detection.

Unsupervised Parametric Approaches

In unsupervised parametric approaches, anomalous instances are not specified, and a summary model is constructed on the base data. A test sequence is then marked anomalous if the probability of generation of the sequence from the model is very low. The anomaly score for the entire time series is computed in terms of the probability of each element. Popular models include Finite State Automata (FSA), Markov Models, and Hidden Markov Models (HMMs).

Finite State Automata (FSA) FSA can be learned from length l subsequences in normal training data. During testing, all length l subsequences can be extracted from a test time series and fed into the FSA. An anomaly is then detected if the FSA reaches a state from where there is no outgoing edge corresponding to the last symbol of the current subsequence. FSAs have been used for outlier detection in [Chandola et al., 2008], [Marceau, 2000], [Michael and Ghosh, 2000], [Salvador and Chan, 2005]. The techniques differ in state representations for the FSA and the methods used to generate the state transition rules.

For example, in [Chandola et al., 2008] and [Marceau, 2000], the states of the FSA are simply the unique elements of the sequence, and the state transition occurs in the FSA based on consecutive symbols in the training sequences.

In [Michael and Ghosh, 2000], each state corresponds to one or more length-l subsequences of input sequences. During training, each sequence is split into sub-sequences of size $l + k$ by a sliding window. The first l elements of the window are used to define a state, and the last k elements are used to label a transition coming out of that state. The first l elements of the next window define the next state that the automaton should enter for this particular training sequence. If we define the current state to be the one defined by the first l elements of the current window, then there are three possibilities.

- The current state already has an outgoing edge that corresponds to the last k elements in the window, and that edge leads to the correct state (i.e., the state defined by the first l elements of the next window). In this case, no modifications are made to the FSA.
- The current state has outgoing edges that correspond to the last k elements in the window, but none of these edges lead to the right state. In this case, the FSA may contain the correct state (but no edge from the current state to the desired state), or else the FSA may not even have a state corresponding the next l-gram. We simply create a state for the new l-gram if one doesn't already exist. In either case, we create a transition from the current state to the new state, and label that transition with the last k elements of the current window.
- The current state has no outgoing edges that correspond to the last k elements of the window. If there is already a state assigned to the next l-gram, then we simply create a transition to that state, and label it with the k elements as in the previous case. However, if the next l-gram doesn't have any state assigned to it, we create a transition back to the current state, and assign the new l-gram to the current state (where it joins whatever l-grams were assigned to that state previously).

[Salvador and Chan, 2005] describe a different and a very interesting way of using FSAs for anomaly detection. Their FSA consists of k normal states and an anomaly state where k is a constant. They first segment the input time series into k clusters. Each cluster corresponds to a normal state of the FSA. The transition rules for the FSA are then generated as follows.

- If the input matches current state's characteristics then remain in current state.
- If the input matches the next state's characteristics then transition to the next state.
- If the input matches neither the current state's nor the next state's characteristics then transition to an anomaly state.

Markov Models Some Markov methods store conditional information for a fixed history size=k, while others use a variable history size to capture richer temporal dependencies. [Ye, 2000] proposes a technique where a Markov model with k=1 is used. In [Yang and Wang, 2003], [Sun et al., 2006], the conditional probability distributions (CPD) are stored in probabilistic suffix trees (PSTs) for efficient computations. Sparse Markovian techniques estimate the conditional probability of an element based on symbols within the previous k symbols, which may not be contiguous or immediately preceding to the element [Eskin et al., 2001], [Lee et al., 1997].

Hidden Markov Models (HMMs) HMMs can be viewed as *temporal dependency-oriented* mixture models, where hidden states and transitions are used to model temporal dependencies among mixture components. HMMs do not scale well to real life datasets. The training process may require judicious selection of the model, the parameters, and initialization values of the parameters. On the other hand, HMMs are interpretable and theoretically well motivated. Approaches that use HMMs for outlier detection include [Chandola et al., 2008], [Florez-Larrahondo et al., 2005], [Gao et al., 2002], [Qiao et al., 2002], [Zhang et al., 2003].

Unsupervised OLAP-Based Approach

Besides traditional uni-variate time series data, richer time series are quite popular. For example, a time series database may contain a set of time series, each of which are associated with multidimensional attributes. Thus, the database can be represented using an OLAP cube, where the time series could be associated with each cell as a measure. It is desirable that this search can automatically find anomalies like: the sales of sports apparel in the past year to one subgroup, males in the same age group who are married and have at least one child, have been declining even though overall group sales have been increasing.

Given a query probe cell c in such a data cube, one would expect that a descendant cell, which is a subset, should roughly follow a similar evolutionary behavior. This leads to the computation of an expected time series and also the anomaly measure, which measures the difference between the expected and observed time series.

[Li and Han, 2007] define anomalies in such a setting, where given a probe cell c, a descendant cell is considered an anomaly if the trend, magnitude, or phase of its associated time series are significantly different from the expected value, using the time series for the probe cell c.

Supervised Approaches

In the presence of labeled training data, the following supervised approaches have been proposed in the literature: positional system calls features with the RIPPER classifier [Lee and Stolfo, 1998], subsequences of positive and negative strings of behavior as features with string matching classifier [González and Dasgupta, 2003], [Cabrera et al., 2001], neural networks [Dasgupta and Nino, 2000], [Endler, 1998], [Gosh et al., 1998], [Ghosh et al., 1999a], [Ghosh and Schwartzbard, 1999], Elman network [Ghosh et al., 1999a], motion features with SVMs [Li et al., 2006], bag of system calls features with decision tree, Naïve Bayes, SVMs [Kang et al., 2005]. Sliding window subsequences have also been used as features with SVMs [Tian et al., 2007], [Wang et al., 2006], rule-based classifiers [Li et al., 2007], and HMMs [Gao et al., 2002].

2.1.2 WINDOW-BASED DETECTION OF OUTLIER TIME SERIES

Given: A database of time series.
Find: All anomalous time windows, and hence anomalous time series.
Examples: General network intrusion attacks [Hofmeyr et al., 1998]; Unix sessions from intruders [Forrest et al., 1996].

Compared to the techniques in the previous section, for window-based outlier detection, the test sequence is broken into multiple overlapping subsequences (windows). The anomaly score is computed for each window, and then the anomaly score (AS) for the entire test sequence is computed in terms of that of the individual windows. Window-based techniques can perform better localization of anomalies, compared to the techniques that output the entire time series as outliers directly. These techniques need the window length as a parameter. Windows are called fingerprints, pattern fragments, detectors, sliding windows, motifs, and n-grams in various contexts. In this methodology, the techniques usually maintain a normal pattern database, but some approaches also maintain a negative pattern or a mixed pattern database. Figure 2.2 shows a general sketch of the method.

Normal Pattern Database Approach

In this approach, normal sequences are divided into size w overlapping windows. Each such window subsequence is stored in a database with its frequency. For a test sequence, subsequences of size w are obtained, and those subsequences that do not occur in normal database are considered mismatches. If a test sequence has a large number of mismatches, it is marked as an anomaly [Gao et al., 2002], [Cabrera et al., 2001], [Endler, 1998], [Ghosh et al., 1999a], [Ghosh et al., 1999b]. Rather than looking for exact matches, if a subsequence is not in the database, soft mismatch

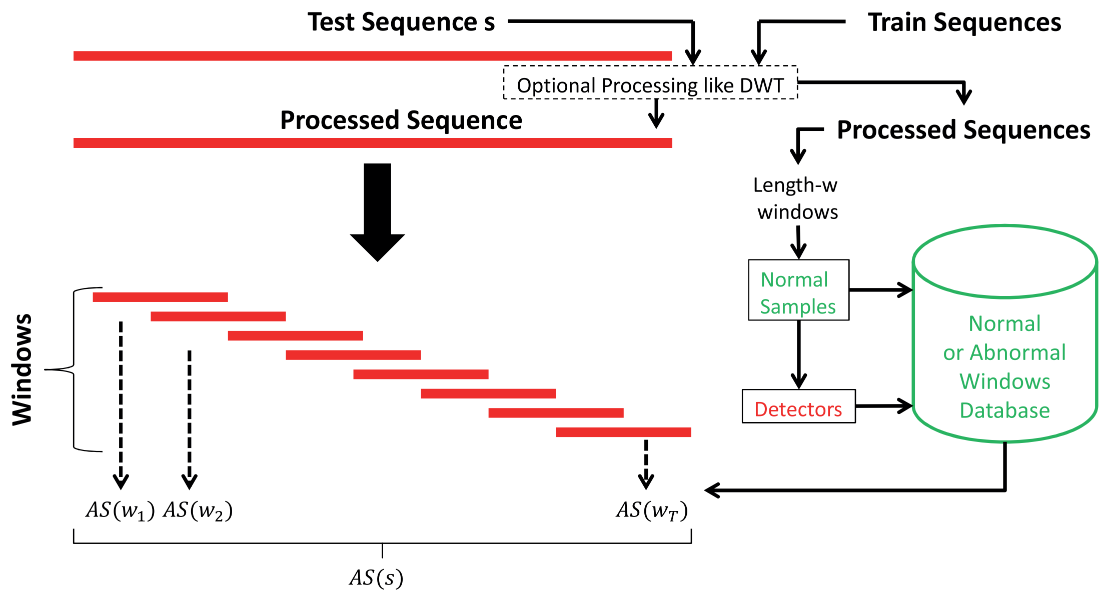

Figure 2.2: Window-based time series outlier detection.

scores can also be computed [Lane et al., 1997], [Hofmeyr et al., 1998], [Lane and Brodley, 1997], [Lane and Brodley, 1998].

Besides contiguous window subsequences, a lookahead-based method can also be used for building a normal database [Forrest et al., 1996]. For every element in every normal sequence, the elements occurring at distance 1,2,. . ., k in the sequence are noted. A normal database of such occurrences is created. Given a new test sequence, a lookahead of the same size k is used. Each pair of element occurrence is checked with the normal database, and the number of mismatches is computed.

Negative and Mixed Pattern Database Approaches

Besides the dictionaries for normal sequences, anomaly dictionaries can also be created [González and Dasgupta, 2003], [Dasgupta and Nino, 2000], [Dasgupta and Majumdar, 2002], [D'haeseleer et al., 1996], [Forrest et al., 1994]. All normal subsequences of length w are obtained from the input string. Next, all sequences of size w not in the set are considered detectors or negative subsequences. Detectors can be generated randomly or by using some domain knowledge of situations that are not expected to occur in normal sequences. A test sequence is then monitored for presence of any detector. If any detector matches, the sequence can be considered an outlier.

2.1.3 OUTLIER SUBSEQUENCES IN A TEST TIME SERIES

Given: A database of time series D and a test time series t.
Find: An outlier subsequence (or a pattern) p in t.
Examples: Weeks of unexpected power consumption [Keogh et al., 2002]; detection of episodes in event sequences [Atallah et al., 2004], [Gwadera et al., 2005b].

The anomaly score for a pattern p can be computed as the difference between the frequency of pattern p in test time series and the expected frequency in database D. [Keogh et al., 2002] define a soft match version of the problem where the frequency of pattern p in the database D is defined using the largest number l such that every subsequence of p of length l occurs at least once in D. Another form of soft matching is defined in [Atallah et al., 2004], where rather than an exact match of pattern p, any permutation of p is also considered a match. To make the computations efficient, the TARZAN algorithm was proposed which exploits suffix trees [Keogh et al., 2002], [Lin et al., 2003], [Lin et al., 2007]. Also, [Gwadera et al., 2005a], [Gwadera et al., 2005b] have proposed Interpolated Markov Models (IMM) to efficiently compute the match score of a pattern or its permutations within any time series.

2.1.4 OUTLIER POINTS ACROSS MULTIPLE TIME SERIES

Given: A database of time series D.
Find: An outlier point across a set of time series in D.
Examples: Faults in complex networks like memory leakage, missing file, busy loop [Guo et al., 2006], [Jiang et al., 2006]; anomalies in room temperature measurements of neighboring areas [Papadimitriou et al., 2005]; and volume anomalies in origin-destination flows in networks [Lakhina et al., 2004b].

This setting involves coupled detection of outlier time points. Rather than finding anomalous time points for each and every time series in the database separately, in this problem setting one is interested in broken correlations in multiple time series. In other words, one aims to capture the correlations among multiple time series and later find when and which time series break out of this correlation.

Jiang et al. [Jiang et al., 2006] use AutoRegressive models with eXogenous inputs (ARX model) to model the correlations between time series corresponding to flow intensities (e.g., HTTP requests to the front Web server) at multiple checkpoints in a complex distributed system. They track multiple such relationships and then based on the differences (residuals) between the actual observations and forecasted values they can predict time points when system faults have occurred. By linking residuals to particular system components, they further tackle the problem of fault isolation. Guo et al. [Guo et al., 2006] use Gaussian mixture models to characterize probabilistic correlation between time series corresponding to flow-intensities measured at multiple points in a complex network. A novel algorithm derived from Expectation-Maximization (EM) algorithm is proposed to learn the "likely" boundary of normal data relationship, which is further used as an oracle in anomaly detection. Papadimitriou et al. [Papadimitriou et al., 2005] use principal component analysis (PCA) for tracking correlation between multiple time series. AR models are used to track the stability of the weight of the principal components across time. An anomaly is detected when actual weights of principal components are very different from those predicted by the AR models across time. Lakhina et al. [Lakhina et al., 2004b] also use the PCA approach to track correlations and detect anomalies from origin-destination traffic data in backbone networks.

2.2 OUTLIERS WITHIN A GIVEN TIME SERIES

Given a single time series, one can find particular elements (or time points) within the time series as outliers, or one can also find subsequence outliers. In this section, we will discuss techniques for both of these cases.

2.2.1 POINTS AS OUTLIERS

Given: A time series t.
Find: Outlier points in t.
Examples: Anomalies in the altitude of the aircraft and roll angle [Basu and Meckesheimer, 2007]; measurement errors in a wind speed data stream [Hill and Minsker, 2010], [Hill et al., 2007].

Various methodologies have been proposed to find outlier points for a time series. A large number of prediction models and profile-based models have been proposed. An information-theoretic compression-based technique has also been proposed to find "deviants." We will discuss these techniques below. Apart from these, a number of clustering and classification approaches, described in Section 2.1.1, can also be used to detect point outliers.

Prediction Models The outlier score for a point in the time series is computed as its deviation from the predicted value by a summary *prediction model*. The primary variation across models, is in terms of the particular prediction model used.

Given a time series, one can predict the value at time t as a median of the values in the size-$2k$ window from $t-k$ to $t+k$ [Basu and Meckesheimer, 2007], or as an average of all the points in the cluster that the value at time t maps to [Hill and Minsker, 2010], or using regression models. Single-layer linear network predictor (or AR model) has been used in a large number of studies including [Hill and Minsker, 2010]. Other prediction models include Multi-layer perceptron (MLP) predictor [Hill and Minsker, 2010] and support vector regression [Ma and Perkins, 2003a]. Mixture transition distribution (MTD) have been proposed for outlier detection for general non-Gaussian time series [Le et al., 1996]. [Tsay et al., 2000] propose a vector ARIMA model to identify additive outliers, innovation outliers, level shifts, and temporary changes from multi-variate time series. Besides individual points, multiple outliers can also be discovered using prediction models, e.g., using re-weighted maximum likelihood estimates [Luceno, 1998] or using Gibbs sampling and block interpolation [Justel et al., 2001].

There are many variants of this technique. Outlier-aware variants of these prediction models estimate model parameters and outliers together [Bianco et al., 2001], [Chang et al., 1988], [Chen and Liu, 1993], [Tsay, 1986]. In multi-variate time series, prediction could be made for all constituent time series. In [Galeano et al., 2006], to compute outliers for multi-variate time series, testing for outliers is done only in some smartly selected projection directions rather than testing the multivariate series directly to compute outliers.

Profile Similarity-Based Approaches These approaches maintain a normal profile and then compare a new time point against this profile to decide whether it is an outlier. For example, for multiple OS performance metric time series, the Tiresias system [Williams et al., 2007] maintains a normal profile and also a variance vector. Any new data point is compared both with the normal profile and the variance vector to compute its anomaly score. Here the profile is the actual smoothed time series data from past data. In [Silvestri et al., 1994], a neural network is used to maintain the normal profile and an estimation is made for the next value in the sensor stream based on this profile. We will discuss more profile-based techniques in Chapter 3 in the context of data streams.

Deviants Deviants are outlier points in time series from a minimum description length (MDL) point of view [Jagadish et al., 1999]. If the removal of a point P from the time sequence results in a sequence that can be represented *significantly* more succinctly than the original one, then the point P is a deviant. These information-theoretic models explore the space-deviation tradeoff by fixing the deviation, rather than fixing the space, as in conventional models. Thus, the problem is to find points whose removal results in a histogram representation with a lower error bound than the original, even after the number of buckets has been reduced to account for the separate storage of these deviant points. [Jagadish et al., 1999] propose a dynamic programming mechanism to solve the problem. [Muthukrishnan et al., 2004] make the observation that for any bucket, the optimal set of k deviants within the bin always consists of the l highest and remaining $k - l$ lowest values for some $l \leq k$. Then, they propose an approximation to the dynamic programming-based solution that maintains a partial solution only for a few interspersed indexes of the time series rather than for each value.

2.2.2 SUBSEQUENCES AS OUTLIERS

Given: A time series t.
Find: Outlier subsequences in t.
Examples: Anomalous subsequences (discords) from electrocardiograms [Keogh et al., 2005]; an odd butterfly given a group of seemingly similar butterflies [Wei et al., 2006]; dacrocytes (indicative of several blood disorders) given a dataset of red blood cells [Keogh et al., 2005]; spores that have sprouted an "appendage" from a fungus dataset [Keogh et al., 2005]; and tool malfunctioning from Space telemetry data [Keogh et al., 2005].

In the previous section, we looked at techniques to identify *point* outliers. In this section, we will visit mechanisms to identify outlier *subsequences* in time series.

Given a time series T, the subsequence D of length n beginning at position l is said to be the discord (or outlier) of T if D has the largest distance to its nearest non-overlapping match [Keogh et al., 2006]. For example, in credit card fraud detection, an individual's credit card transactions are continuously monitored, and a discord, i.e., an anomalous sequence of actions (purchases), may indicate a case of identify theft/misuse. Solutions proposed for this problem all use a standard way of discretizing the time series: Symbolic ApproXimation (SAX) [Lin et al., 2007].

The brute force solution is to consider all possible subsequences $s \in S$ of length n in T and compute the distance of each such s with each other non-overlapping $s' \in S$. The subsequences with the anomaly scores above a user defined threshold are chosen as the top discords. But the problem with this solution is that it requires $O(l^2)$ comparisons of window pairs, where l is the length of the sequence T. Several faster variations have been proposed that can run in approximately linear time in the context of continuous time series and can be extended to discrete sequences.

One general technique for reducing complexity of the basic technique makes use of the following fact. Instead of scoring all windows, they score as many windows as required to get the top k anomalous windows. A top-K pruning approach can be used to make this computation efficient. In other words, a window can be pruned, if at any time its distance to its current m^{th} nearest neighbor is lower than the anomaly score of the current window with the k^{th} largest anomaly score. Because the distance of the window to its actual m^{th} nearest neighbor is upper bounded by the current distance, this window will never figure in the top k anomalous windows, and hence can be pruned. It should be noted that this pruning method guarantees the same result as the basic technique, but can result in lower execution time. Subsequence comparisons can be smartly ordered for effective pruning using various methods like heuristic reordering of candidate subsequences [Keogh et al., 2005], locality sensitive hashing [Wei et al., 2006], Haar wavelet and augmented tries [Bu et al., 2007, Fu et al., 2006], and SAX with augmented trie [Lin et al., 2005]. To compute the distance between subsequences, most methods use Euclidean distance while Compression-based Dissimilarity Measure (CDM) is used as a distance measure in [Keogh et al., 2004]. [Yankov et al., 2008] solve the problem for a large time series stored on the disk. In general, this broader principle was used for pruning even in the earliest distance-based methods for outlier detection in multidimensional data. An overview of such proximity-based methods may be found in Chapter 4 of [Aggarwal, 2013].

[Chen and Zhan, 2008] define the subsequence outlier detection problem for an unequal interval time series which is a time series with values sampled at unequal time intervals. For such a time series, a pattern is defined as a subsequence of two consecutive points. A pattern p is called an anomaly, if there are very few other patterns with the same slope and the same length. To identify anomalies at multiple scales, the Haar transform is used. Haar transform is also used in [Shahabi et al., 2000] to identify multi-level trends and anomalies.

Besides the aforementioned definitions of outlier subsequences, some more variants have also been discussed in the literature. For example, in [Wei et al., 2005] a lead and a lag window are defined as adjacent subsequences. The two windows could be of any length. The subsequence represented by the lead window is an anomaly, if its similarity with the lag window is very low. They measure the similarity using chaos bitmaps. As another example, [Zhu and Shasha, 2003] compute subsequences of length w with a very high aggregate value as outliers. They use a novel data structure called *Shifted Wavelet Tree* to solve the problem.

2.3 CONCLUSIONS AND SUMMARY

A significant amount of work has been performed in the area of time series outliers both in the statistics community and in the data mining community. We discussed two main types of outlier detection techniques for time series studied in the data mining community. The first part was about techniques to detect outliers over a database of time series, whereas the second part dealt with outliers within a single time series. In the former case, individual time series are outliers, whereas in the latter case, portions of a time series are outliers. Given a time series database, we discussed methods to identify a few time series sequences as outliers, or to identify a subsequence in a test sequence as an outlier. An outlier score for a time series was computed directly, or by first computing scores for overlapping fixed size windows and then aggregating them. Given a single time series, we discussed mechanisms to discover particular elements (or time points) within the time series as outliers, and also discussed mechanisms to find subsequence outliers.

Time series data could either be data with limited number of symbols (such as protein sequences), or it could be numeric data (such as the stock price of a company). While some outlier detection mechanisms are specific to one of these two types, most of the methods designed for the first type could be applied to numeric time series by first discretizing the numeric time series. In either case, time series analysis techniques assume that the data is limited in length (with respect to time). However, a large number of recent applications need to deal with an infinite flow of data (i.e., data streams). In the next chapter, we will discuss mechanisms for outlier detection for such data streams.

CHAPTER 3
Outlier Detection for Data Streams

In recent years, advances in hardware technology have facilitated new ways of collecting data continuously. In many applications such as network monitoring, the volume of such data is so large that it may be impossible to store the data on disk. Furthermore, even when the data can be stored, the volume of the incoming data may be so large that it may be impossible to process any particular record more than once. Therefore, many data mining and database operations such as classification, clustering, frequent pattern mining and indexing, and outlier detection have become significantly more challenging in this context. A detailed discussion of many different applications for the streaming scenario may be found in [Aggarwal, 2007].

In this chapter, we will discuss outlier detection mechanisms for data streams. Compared to static data, streaming data does not have a fixed length. In the context of streaming data, the problem of outlier detection is closely related to detecting abrupt changes from previous patterns of activity. These changes may be manifested either in the individual data points, or in the aggregate distributions between successive windows of data points.

The nature of the stream also has a significant impact on the outlier detection process. Streams can be a time-series or multi-dimensional. A multi-dimensional data stream consists of multiple aligned time series. Temporal dependencies for multi-dimensional streams, are used differently than in a single time-series. Rather than using time-series *forecasting* methods, outlier detection methods for multi-dimensional data streams are closer to conventional multi-dimensional models, but with a temporal component, which accounts for temporal drift and deviations.

3.1 EVOLVING PREDICTION MODELS

Given: A multi-dimensional data stream s.
Find: Outlier points in s.
Examples: Measurement errors in a wind speed data stream [Hill and Minsker, 2010], [Hill et al., 2007]; key points of change in Japanese economy like beginning of the bubble economy, black Monday, decay of the bubble economy, and Hanshin-Awaji earthquake [Yamanishi and Takeuchi, 2002]; abnormal disease outbreak from medical stream data.

Evolving prediction models are models in which the parameters or components of the model are updated as new data arrives in order to better capture the normal trends in the data.

Outliers are then detected as those data points which caused a drastic change in the models. In this section, we will discuss a few such evolving prediction models.

3.1.1 ONLINE SEQUENTIAL DISCOUNTING

[Yamanishi and Takeuchi, 2002] and [Yamanishi et al., 2004] present SmartSifter which employs an online discounting learning algorithm to incrementally learn the probabilistic mixture model, with a decay factor to account for drift. The outlier score for a data point is computed in terms of the probabilistic fit value to the learned mixture model. Such mixture models are used for conventional multi-dimensional data, but without the adjustments for incremental updates and temporal decay. Figure 3.1 shows an illustration of the method.

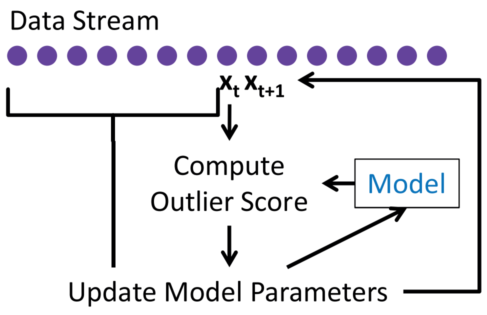

Figure 3.1: Stream anomaly detection.

At any time point, the next element in the time series x_t is processed to compute its outlier score by consulting the already learned model. Based on the outlier score, the model parameters are updated to reflect the knowledge learned at time t. Then, the system proceeds to process data at time $t + 1$.

The model is agnostic to data type (continuous or categorical), because both can be modeled by varying the probabilistic description of the mixture components. Even a combination of numerical and categorical data can be addressed. For categorical variables, they propose the Sequentially Discounting Laplace Estimation (SDLE) algorithm. In SDLE, cells are created by partitioning the space of all values of all categorical variables. The probability of a symbol in a cell is the number of occurrences of that symbol in that cell divided by the total data points with the Laplace smoothing. When a new data point arrives, the count for all the cells are adjusted with temporal discounting and appropriate Laplace smoothing is applied.

Here are a few details of the SDLE algorithm. Let the number of categorical variables in the multi-dimensional data stream be n. Let the range of the i^{th} categorical variable be $A^{(i)} = \{a_1^{(i)}, \ldots, a_{u_i}^{(i)}\}(i = 1, \ldots, n)$. Classify them to obtain a finite number of disjoint sets: $\{A_1^{(i)}, \ldots, A_{v_i}^{(i)}, (i = 1, \ldots, n)\}$, where $A_j^i \cap A_k^{(i)} = \phi(j \neq k)$ and $A^{(i)} = \cup_{j=1}^{v_i} A_j^{(i)}$. Let's call the cell $A_{j_1}^{(1)} \times \ldots \times A_{j_n}^{(n)}$ the $(J-1,\ldots,j_n)^{th}$ cell. We have $M = v_1 \times \ldots \times v_n$ cells in total. This induces a partitioning of the domain.

Given such a partitioning of the domain, a histogram density forms a probability distribution, which takes a constant value on each cell. The histogram density is specified by a parameter $\theta = (q_1,\ldots,q_k)$ where $\sum_{j=1}^{k} q_j = 1$, $q_j \geq 0$ and q_j denotes the probability value for the j^{th} cell. If there are L_j symbols in the j^{th} cell, the probability value of each symbol $\boldsymbol{x}$ in it is given by $p(x) = \frac{q_j}{L_j}$.

The SDLE algorithm can be viewed as a variant of the Laplace Law for smoothing the probability distributions over a discrete domain. As shown in Figure 3.1, the update is done every time a new data point $\boldsymbol{x}_t$ arrives at time t. Let the new data point $\boldsymbol{x}_t$ be represented by $\boldsymbol{x}_t = (x_1,\ldots,x_n)$. Now the following equations capture the updates. For each $(j_1,\ldots,j_n)^{th}$ cell,

$$T_t(j_1,\ldots,j_n) = (1-r_h)T_{t-1}(j_1,\ldots,j_n) + \delta_t(j_1,\ldots,j_n) \tag{3.1}$$

$$q^{(t)}(j_1,\ldots,j_n) = \frac{T_j(j_1,\ldots,j_n)+\beta}{(1-(1-r_h)^t)/r_h + k\beta}. \tag{3.2}$$

For each $\boldsymbol{x} \in A_{j_1}^{(1)} \times A_{j_2}^{(2)} \times \ldots \times A_{j_n}^{(n)}$,

$$p^{(t)}(x) = \frac{q^{(t)}(j_1,\ldots,j_n)}{|A_{j_1}^{(1)}|.|A_{j_2}^{(2)}|\ldots|A_{j_n}^{(n)}|}, \tag{3.3}$$

where $\delta_t(j_1,\ldots,j_n) = 1$ if the t^{th} datum falls into the $(j_1,\ldots,j_n)^{th}$ cell, and $\delta_t(j_1,\ldots,j_n) = 0$ otherwise. Note that r_h is the history discounting factor. A smaller value of r_h means that SDLE will have a larger influence of past examples. Also note that $T(j_1,\ldots,j_n)$ is initialized to 0 at time t=0 for all $1 \leq j_i \leq v_i (i = 1,\ldots,n)$.

For continuous variables, they propose two models: an independent model and a time series model. The independent model is a Gaussian mixture model in the parametric case and a kernel mixture model in the nonparametric case. For learning the Gaussian mixture model (GMM), they provide the Sequentially Discounting EM (SDEM) algorithm which is essentially an incremental EM algorithm with discounting of effect of past examples. SDEM presents two main improvements over the incremental EM algorithm to learn Gaussian mixture models from time series data. (1) Choose the data items in time order for the E-step. Make only one iteration of the E and the M steps for each time step. This updates the parameters each time a datum is input. (2) Introduce a discounting parameter $r(0 < r < 1)$ when computing sufficient statistics in the E-step. This makes the statistics exponentially decay with a factor of $1-r$ as the stage proceeds.

For learning the kernel mixture model, Yamanishi et al. provide Sequentially Discounting Prototype Updating (SDPU) algorithm where the coefficients of the mixture and the variance matrix are fixed and so it iteratively learns only the means of the kernels or the prototypes. For learning the time series model, they provide the Sequentially Discounting AR (SDAR) algorithm which learns the AR model parameters iteratively with time discounting.

All these self-updating models can now be used for anomaly detection. When a new data point $\boldsymbol{x}_t$ comes in, compute its probability of generation given the model at time t and the model at time $t-1$. If the distance between these two probabilities is high, i.e., the model has changed significantly after incorporating the learning from $\boldsymbol{x}_t$, the data point can be marked as an anomaly.

The online discounting behavior is also used to update the "normal" distribution in SRI's Emerald system [Javitz, 1994], by giving more weight to recent data. Online discounting has also been used for mixed attribute data streams, to maintain the frequency of an itemset (a set of attribute values) [Ghoting et al., 2004], where a point is called an outlier if it shares the attribute values with none or very few other points. Exponentially greater importance (another form of online discounting) is given to the recent points to learn a polynomial function in StreamEvent [Aggarwal, 2005a] which is an algorithm to efficiently compute outliers from multiple synchronous streams, though this model is based on the time-series scenario. This algorithm also requires supervision because examples of previous anomalous events need to be available, in order for the system to discover future examples of anomalous events.

3.1.2 DYNAMIC CLUSTER MAINTENANCE

Other than the online discounting methods, large number of methods use dynamically maintained cluster models for computing outliers from data streams. Normalized length of the longest common subsequence (LCS) has been used as the sequence similarity measure for dynamic clustering by [Sequeira and Zaki, 2002]. In the text domain, online clustering methods were proposed in [Aggarwal and Yu, 2010] to detect outliers.

Data streams exhibit considerable variations in the underlying patterns over the course of their progression. Old clusters may become inactive, and eventually get replaced by new clusters. Similarly, when newly arriving data points do not naturally fit in any particular cluster, these need to be initially classified as outliers. However, as time progresses, these new points may create a distinctive pattern of activity which can be recognized as a new cluster. The temporal locality of the data stream is manifested by these new clusters. For example, the first web page belonging to a particular category in a news stream of current events may be recognized as an outlier, but may later form a cluster of documents of its own. On the other hand, the new outliers may not necessarily result in the formation of new clusters. Such outliers are true short-term abnormalities in the data since they do not result in the emergence of sustainable patterns.

Consider the case where we want to find anomalous sequences from a stream of sequences. The k-means clustering method is often used, because it allows reallocation of samples even after assignment and it converges quickly. The problem with basic k-Means is that the random allocation of cluster centers reduces its accuracy. Also, k (number of clusters) and t (number of iterations) are hard to set to achieve a good clustering.

To counter this, [Sequeira and Zaki, 2002] propose a dynamic clustering approach to group a set of sequences, where clusters are grown when distance of a sequence from nearest cluster centroid is greater than the intra-cluster similarity r. Clustering is performed iteratively by changing the initial cluster centroids, until all sequences get clustered to some cluster.

Although this Dynamic Clustering algorithm counters all the basic k-Means disadvantages, setting the intra-cluster similarity r may require experimentation. Also, a cluster may have a lot in common with another, i.e., sequences assigned to it are as close to it as they are to another cluster. There may also be denser sub-clusters within the larger ones. To tackle these problems, the clustering is improved by refining clusters (merging and splitting clusters).

Once the clustering is done, each cluster represents a normal sequence pattern. When a new sequence comes in, it is marked as anomalous if the nearest matching cluster is consider too far to be considered statistically to be relevant to that cluster. Furthermore, this approach also uses the similarity between the new sequence and a few past sequences to distinguish between noise, concept drift or masquerader in a dataset of user command streams.

While the above method aims at just maintaining the overall clustering of the dataset, [Aggarwal and Yu, 2010] study detailed evolution of clusters with time discounting for points within the cluster. When a cluster is newly discovered during the arrival of the data stream, it is referred to as a trend-setter. From the point of view of the user, a trend-setter is an outlier, until the arrival of other points certify the fact that it is actually a cluster. If and when a sufficient number of new points have arrived in the cluster, it is referred to as a mature cluster. At a given moment in time, a mature cluster can either be active or inactive. A mature cluster is said to be active when it has continued to receive data points in the recent past. When a mature cluster is not active, it is said to be inactive. In some cases, a trend-setting cluster becomes inactive before it has had a chance to mature. Such a cluster typically contains a small number of transient data points. Since the stream clustering process should provide a greater level of importance to recent clusters, they provide a time-sensitive weightage to each data point. It is assumed that each data point has a time-dependent weight defined by the function $f(t)$. The function $f(t)$ is also referred to as the fading function.

For every cluster, they maintain a few statistics (together called as the cluster droplet). For example for categorical data, a cluster is represented using (1) $\overline{DF2}$, (2) $\overline{DF1}$, (3) n, (4) $w(t)$, and (5) l. Let the original categorical data have d dimensions. Let v_i be the number of possible categorical values of dimension i and v_j be the number of possible values of dimension j. The vector $\overline{DF2}$ contains $\sum_{i\in\{1\ldots d\},j\in\{1\ldots d\},i\neq j} v_i.v_j$ entries. For each of the $v_i.v_j$ categorical value pairs i and j, weighted counts of number of points for each value pair which are included in cluster C are maintained. The vector $\overline{DF1}$ contains $\sum_{i=1}^{d} v_i$ entries. For each i, we maintain a count of each of the v_i possible values of categorical attribute i occurring in the cluster. n is the number of data points. $w(t)$ contains the sum of the weights of the data points at time t. l contains the time stamp of the last time that a data point was added to the cluster. For text data, each dimension is essentially the frequency of occurrence of that word in a document.

Based on this definition of cluster droplets, they make two important observations which are important for dynamic cluster maintenance.

Observation 1: Consider the cluster droplets $D(t, C_1) = (\overline{DF2_1}, \overline{DF1_1}, n_1, w(t)_1, l_1)$ and $D(t, C_2) = (\overline{DF2_2}, \overline{DF1_2}, n_2, w(t)_2, l_2)$. Then the cluster droplet $D(t, C_1 \cup C_2)$ is defined by the tuple $(\overline{DF2_1} + \overline{DF2_2}, \overline{DF1_1} + \overline{DF1_2}, n1 + n2, w(t)_1 + w(t)_2, max\{l_1, l_2\})$.

Observation 2: Consider the cluster droplet $D(t, C) = (\overline{DF2}, \overline{DF1}, n, w(t), l)$. Then the entries of the same cluster droplet C at a time $t' > t$ (without the addition of new data points) are given by $D(t', C) = (\overline{DF2}.2^{-\lambda.(t'-t)}, \overline{DF1}.2^{-\lambda.(t'-t)}, n, w(t).2^{-\lambda.(t'-t)}, l)$.

These two observations play a vital role in dynamic maintenance of these cluster droplets. The maintenance algorithm continuously maintains the droplets $C_1 \ldots C_k$, which it updates as new data points arrive. For each cluster, the entire set of statistics in the droplet is maintained.

At the beginning of the maintenance algorithmic execution, we start with an empty set of clusters. As new data points arrive, unit clusters containing individual data points are created. Once a maximum number k of such clusters have been created, we can begin the process of online cluster maintenance. Thus, we initially start off with a trivial set of k clusters. These clusters are updated over time with the arrival of new data points. When a new data point $\overline{X}$ arrives, its similarity to each cluster droplet is computed. Similarity computation between the new data point and a cluster droplet can be computed using similarity between the $\overline{DF1}$ of the cluster droplet and the vector corresponding to the new data point. The new data point is allocated to the cluster with maximum similarity, if the similarity is greater than a threshold. If the maximum similarity between the new data point and any cluster droplet is less than the threshold, a new cluster is created for the new data point and the droplet statistics are computed.

When a new data point has to be added to a cluster, the statistics are updated to reflect the decay of the data points at the current moment in time. This update is performed using the computation discussed in Observation 2. Thus, the relevant updates are performed in a "lazy" fashion. In other words, the statistics for a cluster do not decay, until a new point is added to it. Such dynamically maintained clusters are further analyzed to discover outliers.

Besides these dynamic clustering algorithms, algorithms have also been proposed for dynamic (evolutionary) clustering of network data. Dynamic clusters help recognize anomalies in a much more effective way compared to static or batch clustering. For example, [Aggarwal et al., 2011] maintain dynamic partitions of a network based on the structural connectivity in the incoming graph stream seen so far. They then use this structural connectivity model-based partitions to define outliers in graph streams as those graph objects which contain unusual bridging edges. The work in [Aggarwal and Subbian, 2012] designs methods for maintaining clusters in a social stream. Significant deviations from these clusters are reported as interesting events. Social streams are very similar to text stream, except that they are more noisy and also contain network information about connections between social network participants. This enriches the event detection and knowledge discovery process.

3.1.3 DYNAMIC BAYESIAN NETWORKS (DBNS)

Sometimes, updating the parameters of a model may not be sufficient to account for the concept drift in the underlying data stream. The model itself may change significantly. An approach which modifies the model to account for concept drifts in the data stream is discussed in [Hill et al., 2007]. This work presents an approach that uses Dynamic Bayesian networks which are Bayesian networks with network topology that evolves over time, adding new state variables to represent the system state at the current time t. State variables can be categorized as either unknown (hidden) state variables that represent the true states of the system or measured (observed) state variables that represent imperfect measurements of one or more of the true states; state variables can be either discrete or continuous valued.

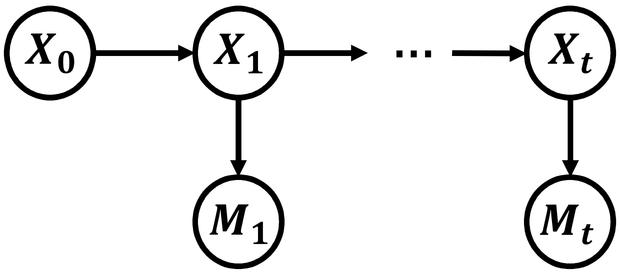

Figure 3.2: Graphical structure of DBN-1. Vector X represents the continuous valued, hidden system variables and vector M represents the continuous valued, observed system variables. Subscripts indicate time.

Because the network size increases over time, performing inference using the entire network would be intractable for all but trivial time durations. However, efficient recursive algorithms have been developed that perform exact inference on specific types of DBNs or approximate inference on more general types of DBNs. Two of these algorithms are Kalman filtering and Rao-Blackwellized particle filtering. Both algorithms perform filtering, or inference of the current hidden system states given all of the observed states to date. Kalman filtering employs the assumption that all state variables are linear Gaussian random processes to perform exact inference, while Rao-Blackwellized particle filtering uses a sample of the state distributions (the particles) to perform approximate inference and thus does not limit the type of state variables.

They present two strategies for detecting anomalous data: Bayesian credible interval (BCI) and maximum a posteriori measurement status (MAP-ms). The first method uses a simple DBN model, as shown in Figure 3.2, which tracks the multivariate distributions of linear Gaussian state variables corresponding to the hidden system states and their observed counterparts that are measured by the sensors and available as data stream of measurements. The hidden states are assumed to be first-order Markov processes, so the state at time t only depends on the state at time $t-1$. Kalman filtering is used to sequentially infer the posterior distributions of the hidden and the

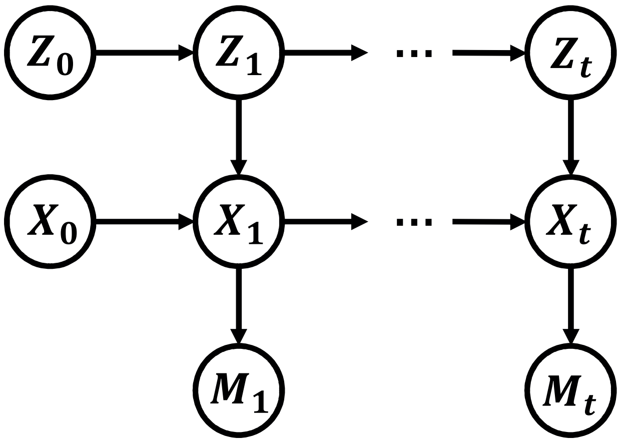
$$\begin{tikzcd}
Z_0 \arrow[r] & Z_1 \arrow[r] \arrow[d] & \cdots \arrow[r] & Z_t \arrow[d] \\
X_0 \arrow[r] & X_1 \arrow[r] \arrow[d] & \cdots \arrow[r] & X_t \arrow[d] \\
 & M_1 & & M_t
\end{tikzcd}$$

Figure 3.3: Graphical structure of DBN-2. Vectors X and Z represent the continuous valued and discrete valued hidden system variables, respectively, and vector M represents the continuously valued observed system variables. Subscripts indicate time.

observed states as new measurements become available from the data stream. The posterior distribution of the observed state variables can then be used to construct a Bayesian credible interval for the most recent set of measurements. The p% credible interval indicates that the posterior probability of the observed state variables falling within the interval is p; thus, the Bayesian credible interval delineates the range of plausible values for sensor measurements. For this reason, any measurements that fall outside of the p% Bayesian credible interval can be classified as anomalous. The network parameters (i.e., the probability distributions $P(X_0)$, $P(X_t|X_{t-1})$, $P(M_t|X_t)$) for DBN-1 were learned from sensor data using the expectation-maximization algorithm.

In the second method, a 2-layered DBN as shown in Figure 3.3 is used. It tracks the multivariate distributions of linear Gaussian state variables corresponding to hidden system states and their observed counterparts, which are measured by the environmental sensors, as well as the distribution of a discrete hidden state variable which indicates the status (e.g., normal/anomalous) of each sensor measurement. For example, if there are two measured states, then the measurement status variable will have four values: (normal, normal), (anomalous, normal), (normal, anomalous), and (anomalous, anomalous). Rao-Blackwellized particle filtering is used to sequentially infer the posterior distributions of the hidden and observed states as new measurements become available from the sensors. The maximum a posteriori estimate, (e.g., the most likely value given the posterior distribution) of the hidden state variable indicating the measurement status can then be used to classify the sensor measurements as normal or anomalous. DBN-2 requires (1) network parameters describing the time-evolution of the linear Gaussian states conditioned on each value of the discrete state and (2) parameters describing the time-evolution of the discrete state. For the case in which all sensor measurements were normal, the parameters of the linear Gaussian states were specified to be the same as those learned for DBN-1. For the cases in which one or more measurements was anomalous, the parameter specifying the measurement variance of the anomalous measurement was set to be a large number (e.g., 10,000), indicating that regardless of the true state of the system, the measurement could take any real value with approximately equal probability. This description of anomalous measurements was used because it indicates that an anomalous measurement is more likely to fall outside the range of plausible measurements than a normal measurement yet it does not require a priori knowledge of the types of anomalies that can occur. The discrete state distributions (i.e., $P(Z_0), P(Z_t|Z_{t-1})$) were set manually, using domain knowledge. Manually setting the parameters for the cases in which one or more measurements was anomalous was necessary because anomalous measurements are, by definition, infrequent; as such, insufficient information is available for learning these parameters from the data. Furthermore, learned parameters may define anomalies too narrowly to identify the range of anomalies that may be encountered.

3.2 DISTANCE-BASED OUTLIERS FOR SLIDING WINDOWS

Given: A data stream s.
Find: Distance-based outliers in any time window of s.
Examples: Distance outliers from the rain, sea surface temperature, relative humidity, precipitation time series [Angiulli and Fassetti, 2007]; surveillance video outliers like appearance of a new object, object zooming in a camera, and novel video content [Pokrajac et al., 2007]; anomalous behaviors such as people first walking right and then back left, or people walking very slowly [Pokrajac et al., 2007].

Besides defining outliers for data streams using prediction models, one can also compute distance-based outliers for data streams at any time instant. Given a set of points, a point o is called a $DB(k, R)$ distance outlier if there are less than k points within distance R from o [Knorr and Ng, 1998]. Given a data stream, at any time point, one can discover outliers from the set of points lying within the current sliding window. Distance-based outliers can be discovered both in a global as well as in a local sense on points within the current sliding window. Figure 3.4 shows a 1-dimensional stream dataset with two time windows (t_3 to t_{18} and t_7 to t_{22}). Consider the window at time t_{18}. If we consider a point with $k < 4$ points within distance R as outliers, $o9$ (with k=4 neighbors $o5$, $o10$, $o14$ and $o15$) and $o11$ (with k=4 neighbors $o3$, $o4$, $o6$ and $o13$) are examples of inliers, while $o8$ (with only 2 neighbors $o7$ and $o16$) and $o17$ (with only 1 neighbor $o18$) are examples of outliers.

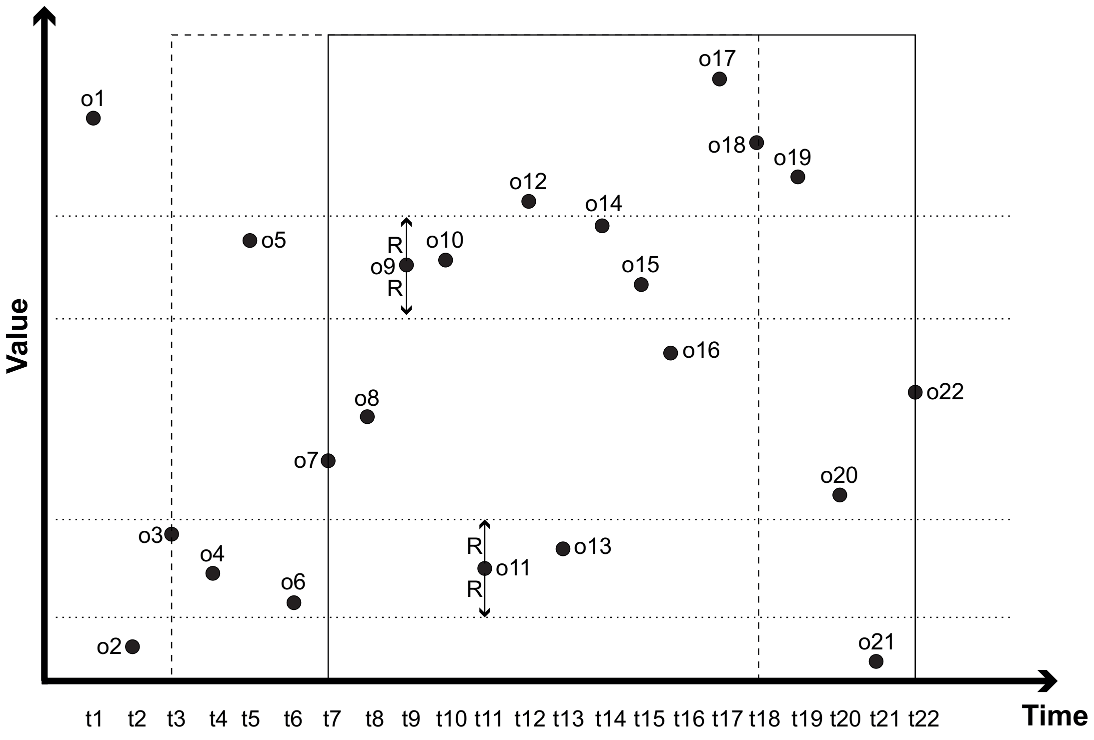

Figure 3.4: Distance-based outliers for sliding windows. Based on [Angiulli and Fassetti, 2007].

3.2.1 DISTANCE-BASED GLOBAL OUTLIERS

As the sliding window for a stream moves, old objects expire and new objects come in. Since objects expire over time, the number of preceding neighbors of any object decreases. Therefore, if the number of succeeding neighbors of an object is less than k, the object could become an outlier depending on the stream evolution. Conversely, since any object will expire before its succeeding neighbors, inliers having at least k succeeding neighbors will be inliers for any stream evolution. Such inliers are called safe inliers (e.g., in Figure 3.4, for k=3, $o9$ is a safe inlier as it has 3 succeeding neighbors ($o10$, $o14$, $o15$) while $o11$ is not, as it has only 1 succeeding neighbor ($o13$)).

[Angiulli and Fassetti, 2007] propose an exact algorithm to efficiently compute such distance outliers using a new data structure called Indexed Stream Buffer (ISB) which supports a range query. ISB maintains a summary of the current window by storing every data stream object as a different node. A node n consists the following summary about the corresponding object: identifier (arrival time) of the object, the number of succeeding neighbors, a list, having size at most k, containing the identifiers of the most recent preceding nearest neighbors. Further, they also propose an approximate algorithm which uses two heuristics. (a) It is sufficient to retain in ISB only a fraction of safe inliers; (b) rather than storing the list of k most recent preceding neighbors, it is enough to store only the fraction of preceding neighbors, which are safe inliers to the total number of safe inliers.

[Yang et al., 2009] propose that maintaining all neighbor relationships across time may be very expensive. Therefore, abstracted neighbor relationships can be maintained. However, maintaining such cluster abstractions is expensive too. Hence, they exploit an important characteristic of sliding windows, namely the "predictability" of the expiration of existing objects. In particular, given the objects in the current window, the pattern structures that will persist in subsequent windows can be predicted by considering the objects (in the current window) that will participate in each of these windows only. These predicted pattern structures can be abstracted into "predicted views" of each future window. They propose an efficient algorithm which makes use of the predicted views to compute distance-based outliers.

The problem of distance-based outlier detection for stream data can also be solved using dynamic cluster maintenance, as has been done for similar problems in [Bu et al., 2009] and [Cao et al., 2010]. Specifically, these two papers deal with the following interesting variants of the distance-based outlier detection problem. (1) Consider three adjacent windows: base window B, left window L and right window R of sizes w_b, w_l, and w_r in the order L, B, and R. B is an anomaly if there are $< k$ subsequences in L and R with distance from $B < d$ [Bu et al., 2009]. (2) Given sliding window with tuples $p_1, \ldots, p_W$, which are clustered into m clusters $C_1, \ldots, C_m$, each containing some tuples similar to each other, and a tuple p_i with timestamp t_i that falls into the current window and belongs to the cluster C_j, then p_i is called an outlier if there are no more than k tuples in C_j lying within distance R from p_i [Cao et al., 2010].

3.2.2 DISTANCE-BASED LOCAL OUTLIERS

Local Outlier Factor (LOF) is an algorithm for finding anomalous data points by measuring the local deviation of a given data point with respect to its neighbors [Breunig et al., 2000]. The LOF algorithm developed for static data can be adapted to the incremental LOF problem (i.e., the stream setting) in three ways: (a) periodic LOF, i.e., apply LOF on entire data set periodically; (b) supervised LOF, i.e., compute the k-distances, local reachability density (LRD) and LOF values using training data and use them to find outliers in test data; or as (c) iterated LOF where the static LOF algorithm can be re-applied every time a new data record is inserted into the data set. A better approach is proposed in [Pokrajac et al., 2007], where the incremental LOF algorithm computes LOF value for each data record inserted into the data set and instantly determines whether the inserted data record is an outlier. In addition, LOF values for existing data records are updated if needed. Thus, in the insertion part, the algorithm performs two steps: (a) insertion of new record, when it computes the reachability distance, LRD and LOF values of a new point; and (b) maintenance, when it updates k-distances, reachability distance, LRD, and LOF values for affected existing points.

3.3 OUTLIERS IN HIGH-DIMENSIONAL DATA STREAMS

The combination of the stream scenario and high-dimensional data is particularly challenging because of the complexity of high-dimensional outlier detection algorithms. It has been shown in [Aggarwal and Yu, 2001] that outlier detection in high-dimensional data is challenging because of the need to discover outliers in different subspaces of the data. Most of the dimensions are irrelevant, and the inclusion of the irrelevant dimensions leads to difficulty in detecting outliers with high-dimensional methods. In particular, the distances become increasing concentrated, and the inclusion of the irrelevant dimensions has a masking effect on the outlierness of the underlying data points. Generally, the process of determining the relevant subspaces is computationally expensive because an exponential number of combinations exist in which outliers may be found. A detailed discussion of these issues may be found in Chapter 5 of [Aggarwal, 2013]. The work in [Aggarwal and Yu, 2001] is, however, designed for static data, because it relies on exploring different subsets of dimensions with the use of a genetic algorithm. Such a method is difficult to generalize to the streaming case.

In the streaming scenario, the discovery of relevant subspaces is more difficult because of the exponential number of combinations of dimensions that need to be explored. The work in [Zhang et al., 2008] present Stream Projected Outlier deTector (SPOT), to deal with outlier detection problem in high-dimensional data streams. SPOT employs a window-based time model and decaying cell summaries to capture statistics like Relative Density and Inverse Relative Standard Deviation of data points in the cell from the data stream. These statistics are useful for detecting whether the cell is sufficiently sparse and hence highly likely to contain outliers. Indeed, SPOT constructs a set of top sparse subspaces by using unsupervised and/or supervised learning processes which are explored to find projected outliers effectively. They call these subspaces of interest as Sparse Subspace Template (SST).

Three main types of subspaces are included in SST. They are as follows.

- Fixed SST Subspaces (FS): Fixed SST Subspaces (FS) contains all the subspaces in the full lattice whose maximum dimension is *MaxDimension*, where *MaxDimension* is a user-specified parameter. In other words, FS contains all the subspaces with dimensions of $1, 2, \ldots, MaxDimension$.

- Clustering-based SST Subspaces (CS): SPOT takes in unlabeled training data from the data stream and automatically find the set of subspaces in which data exhibit a high level of overall sparsity, indicated by PCS with small RD and IRSD. Intuitively, these subspaces are where projected outliers are likely to exist. A multi-objective Genetic Algorithm (MOGA) is employed to search the lattice to find those top subspaces in which training data exhibit the highest sparsity. Clustering along with MOGA is used to detect such sparse subspaces as CS.

- Outlier-driven SST Subspaces (OS): A few outlier examples may be provided by domain experts. MOGA is applied on each of these outliers to find their top sparse subspaces. These subspaces are called Outlier-driven SST Subspaces (OS). Based on OS, example-based outlier detection can be performed that effectively detects more outliers that are similar to these outlier examples. Thus, this approach effectively uses supervision to improve the outlier detection process.

Finally, SPOT is also capable of online self-evolution, to cope with evolving dynamics of data streams. One advantage of such algorithms is that they provide information about the dimensions which are most relevant to the outliers [Aggarwal and Yu, 2001]. This is useful for describing outliers.

3.4 DETECTING AGGREGATE WINDOWS OF CHANGE

As discussed earlier, change point detection is closely related to outlier detection in temporal data. While the aforementioned methods for effective for determining individual data points of change, they are not particularly effective at determining windows of time in which the stream has changed very significantly.

A well-known method for finding significant windows of change is the technique of *velocity density estimation*. The idea in velocity density [Aggarwal, 2005b] 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, 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 $\overline{f}(\overline{X})$ based on n data points and kernel function $K'_h(\cdot)$ is defined as follows:

$$\overline{f}(\overline{X}) = (1/n) \cdot \sum_{i=1}^{n} K'_h(\overline{X} - \overline{X_i}).$$

Thus, each discrete point $\overline{X_i}$ in the data set is replaced by a continuous function $K'_h(\cdot)$ which peaks at X_i and has a variance which is determined by the smoothing parameter h. An example of such a distribution would be a Gaussian kernel with width h:

$$K'_h(\overline{X} - \overline{X_i}) = (1/\sqrt{2\pi} \cdot h) \cdot e^{-|\overline{X} - \overline{X_i}|^2/(2h^2)}.$$

The estimation error is defined by the kernel width h which is chosen in a data driven manner. The approach can also be easily generalized to the d-dimensional case by using multiple kernel functions.

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 in the estimation process.

Let t be the current instant and S be the set of data points which 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 which 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 ith data point in S is denoted by $(\overline{X_i}, t_i)$, 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 $\overline{X}$ and time t is given by:

$$F_{(h_s,h_t)}(\overline{X}, t) = C_f \cdot \sum_{i=1}^{|S|} K_{(h_s,h_t)}(\overline{X} - \overline{X_i}, t - t_i).$$

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 $K_{(h_s,h_t)}(\overline{X} - \overline{X_i}, t - t_i)$ 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:

$$\int_{\text{All } X} F_{(h_s,h_t)}(\overline{X}, t)\delta X = 1.$$

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 which 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 $R_{(h_s,h_t)}(\overline{X}, t)$ is defined as follows:

$$R_{(h_s,h_t)}(\overline{X}, t) = C_r \cdot \sum_{i=1}^{|U|} K_{(h_s,h_t)}(\overline{X} - \overline{X_i}, t_i - t).$$

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 $V_{(h_s,h_t)}(\overline{X}, T)$ at spatial location $\overline{X}$ and time T is defined as follows:

$$V_{(h_s,h_t)}(\overline{X}, T) = \frac{F_{(h_s,h_t)}(X, T) - R_{(h_s,h_t)}(\overline{X}, T - h_t)}{h_t}.$$

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. In general, it has been shown in [Aggarwal, 2005b] that when the spatiotemporal kernel function is defined as below, then the velocity density is directly proportional to a rate of change of the data density at a given point:

$$K_{(h_s,h_t)}(X,t) = (1 - t/h_t) \cdot K'_{h_s}(X).$$

This kernel function is only defined for values of t in the range $(0, h_t)$. The Gaussian spatial kernel function $K'_{h_s}(\cdot)$ was used because of its well-known effectiveness [Silverman, 1986]. Specifically, $K'_{h_s}(\cdot)$ is the product of d identical Gaussian kernel functions, and $h_s = (h_s^1, \ldots h_s^d)$, 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 $\overline{X}$ at time-instants t with unusually large absolute values of the* **local** *velocity density.* If desired, a normal distribution or student 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 multi-dimensional data distributions into a quantification, which can be used in conjunction with extreme-value analysis.

It is important to note that the data point $\overline{X}$ 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 [Aggarwal, 2005b]. Time instants with large values of the aggregate velocity density may be declared outliers.

A different way to characterizing aggregate changes in multi-dimensional 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 [Kifer et al., 2004] proposes a nonparametric 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 one-dimensional data streams, and the generalization to higher dimensions is not discussed.

The work in [Dasu et al., 2006] is a bit more general, in that it can address multi-dimensional 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.

3.5 SUPERVISED METHODS FOR STREAMING OUTLIER DETECTION

In many applications, labels are attached to individual data records. In such cases, it is much more effective to perform the outlier detection with the use of these labels. This is referred to as the *streaming rare class detection problem*. When labeled data is available, the quality of the determined outliers is much better with the use of the labels. The supervised outlier detection problem for temporal data is closely related to the streaming classification problem [Aggarwal, 2014]. The major distinctions from the streaming classification problem are as follows.

1. *Class Imbalance:* Typically, far fewer instances are present for the rare class, as compared to the normal class. This is referred to as the *class imbalance issue*. It is necessary to modify the classification algorithms in order to account for the class imbalance. It is much more important to classify the rare instances correctly, as compared to the normal instances.

2. *Novel Class:* In some cases, an entirely new class may appear, that has not been seen before. These are referred to as novel classes. If training data is not available for the novel class, then unsupervised methods need to be combined with the classification process in order to detect such data points.

3. *Recurring Rare class:* In some cases, a rare class may appear, then disappear for a while, before appearing again. Such classes need to be detected effectively by the modeling process.

In addition, the common problems of concept-drift are also present in these cases [Aggarwal, 2014]. A variety of streaming ensemble-based methods have been proposed in [Masud et al., 2010], [Masud et al., 2011], [Al-Khateeb et al., 2012], and [Masud et al., 2013], that can address these issues.

3.6 CONCLUSIONS AND SUMMARY

Streaming data is ubiquitous. Examples include Twitter streams, video surveillance data streams, sensor data streams, etc. The scale and the velocity of stream data require us to design highly efficient and single data scan methods. Besides these challenges, stream analysis models should be able to account for temporal drift and deviations.

In this chapter, we discussed three main types of outlier detection models for stream data. (1) Evolving prediction models are models in which the parameters or components of the model are updated as new data arrives in order to better capture the normal trends in the data. Outliers are then detected as those data points which caused a drastic change in the models. We discussed online sequential discounting models, dynamic cluster maintenance methods, and dynamic Bayesian network models. (2) Distance-based outliers have been studied in depth for static scenarios. We discussed mechanisms for fast incremental computation of both global and local outliers. (3) Outlier detection for high-dimensional data is further challenging because of the curse of dimensionality. We discussed a method which first identifies candidate outlying subspaces and then performs outlier detection in these subspaces.

CHAPTER 4

Outlier Detection for Distributed Data Streams

A large amount of data today is collected using distributed sensors. For example, environmental sensors which measure temperature, humidity, etc., are deployed at various geographic locations and then data is often transmitted to central stations. Similarly, Twitter data can also be considered as distributed data obtained from human sensors spread all across the world. [Zhang et al., 2008] provide a detailed survey for outlier detection techniques on wireless sensor networks. In this chapter, we focus on outlier detection from such distributed data streams.

4.1 EXAMPLES AND CHALLENGES

Here a few examples which demonstrate real-life scenarios where outlier detection from distributed data has been found to be useful.

- Environmental monitoring, in which sensors such as temperature and humidity are deployed in harsh and unattended regions to monitor the natural environment. Outlier detection can identify when and where an event occurs and trigger an alarm upon detection.
- Habitat monitoring, in which endangered species can be equipped with small non-intrusive sensors to monitor their behavior. Outlier detection can indicate abnormal behaviors of the species and provide a closer observation about behavior of individuals and groups.
- Health and medical monitoring, in which patients are equipped with small sensors on multiple different positions of their body to monitor their well-being. Outlier detection showing unusual records can indicate whether the patient has potential diseases and allow doctors to take effective medical care.
- Industrial monitoring, in which machines are equipped with temperature, pressure, or vibration amplitude sensors to monitor their operation. Outlier detection can quickly identify anomalous readings to indicate possible malfunction or any other abnormality in the machines and allow for their corrections. This has been found to be useful in monitoring nuclear reactors and monitoring Map-Reduce based distributed computer systems.
- Target tracking, in which sensors are embedded in moving targets to track them in real-time. Outlier detection can filter erroneous information to improve the estimation of the location of targets and also to make tracking more efficiently and accurately.

- Surveillance monitoring, in which multiple sensitive and unobtrusive sensors are deployed in restricted areas. Outlier detection identifying the position of the source of the anomaly can prevent unauthorized access and potential attacks by adversaries in order to enhance the security of these areas.
- Voting fairness, where voting data can be monitored for anomalous behavior. Outlier detection could identify individuals who voted significantly differently from his party members.

The main challenges for outlier detection in a distributed setting are as follows.

- Resource constraints: Energy, memory, computational capacity, and communication bandwidth are all scarce.
- High communication cost: Communication cost is orders of magnitude greater than the computation costs.
- Distributed streaming data: Processing data online coming at different rates from multiple distributed sensors is non-trivial.
- Dynamics: Dynamic network topology, frequent communication failures, mobility, and heterogeneity of nodes are a few dynamic features of such datasets which are challenging to handle.
- Large-scale deployment: Traditional outlier detection algorithms are not easily scalable.
- Identifying outlier sources: It is important to make distinctions between errors, events, and malicious attacks.

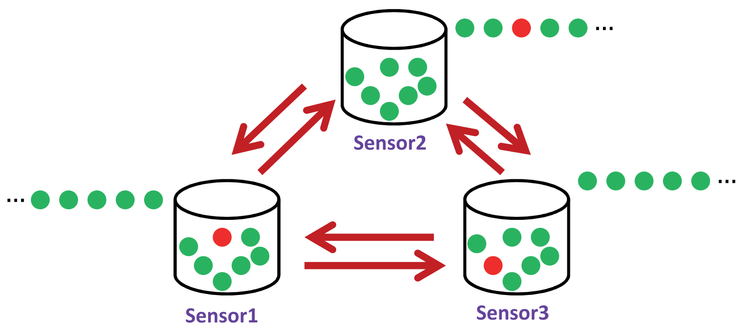

Figure 4.1: Distributed data streams scenario (horizontal data partitioning).

In a distributed stream setting, points are distributed across various nodes (sensors). Each sensor has an associated stream of incoming points and the aim is to find top few distance-based outliers based on the global data (Figure 4.1). In a distributed *spatio-temporal* setting, the position of sensors is also important when computing outliers. We discuss both the settings in this chapter. In a distributed setting, the sensors must either replicate the entire data set locally to some central location, or engage in voluminous queries to calculate the distances between data points residing on different sensors. However, this is not feasible due to the excessive storage and communication requirements. In this chapter, we will see methods to handle such problems.

Given: A set of sensors each with an associated data stream. Optionally, the position ((x,y) coordinates) of the sensors might also be available.

Find: Distance/density-based outlier data points based on global data or outlier sensors or outlier geographic regions.

Examples: Anomalous air temperature sensors [Bettencourt et al., 2007]; anomalous soil temperature readings [Bettencourt et al., 2007]; people with unusual combinations of demographic attribute values [Otey et al., 2006]; storms from wind dataset [Franke and Gertz, 2008].

4.2 SHARING DATA POINTS

The data appears in a distributed form at various sensor locations. The simplest approach is to collect data from all the locations at a single central location. Once all the data is collected at the central location in a streaming way, any of the outlier detection mechanisms for data streams (from among those that we studied in Chapter 3 can be used.

However, sharing all the data leads to huge communication cost. Hence, when the position of the sensors is known, one way is to share data only with the neighbors. [Elnahrawy and Nath, 2004] design a mechanism where data is shared only with the neighbors. A sensor s learns a Bayesian classifier using the following features based on its contextual information: (1) the current readings of the immediate neighbors (spatial); and (2) the last reading of the sensor (temporal). The sensor values are divided into various ranges (categories) and the classifier is used to predict a categorical label. If the predicted categorical label matches the actual range of the observed value at the sensor, the value is normal, else it is flagged as an anomaly.

Further, [Bettencourt et al., 2007] propose that sensors learn the distributions of differences among their own readings (over time), as well as the distributions of differences between their readings and the readings of their neighbors. Then, comparing the current readings to these distributions, allows sensors to identify local outliers using a significance test, and a user-specified threshold.

Sharing data with neighboring sensors can also be used for identifying outlier sensors and outlier regions. Consider neighbors of a sensor s as all other sensors within distance r from it. Given spatio-temporal data from a sensor network, a sensor s can be considered as an outlier for a time window w, if the value at sensor s at the end of the window has less than k measurements from all sensors within window w, within a distance r. Once outlier sensors are discovered, [Franke and Gertz, 2008] and [Franke and Gertz, 2009] propose models to compute polygonal outlier regions in the sensor network.

Sharing all data points with all neighbors still incurs a large communication cost. A better method is to share only local outliers and a few selected data points.

4.3 SHARING LOCAL OUTLIERS AND OTHER DATA POINTS

Let us first discuss an algorithm [Branch et al., 2006] for a distributed static setting, where each sensor holds a set of points and tries to compute the set of the global top-K outliers. An outlier detection algorithm A is available which outputs a ranking function R. R is anti-monotonic wrt data and R should be smooth, e.g., the distance to the k^{th} nearest neighbor, the average distance to the k nearest neighbors, etc. Consider a point x in dataset P. Consider a subset P_0 of P. If P_0 is the smallest subset such that $R(x, P) = R(x, P_0)$ then P_0 is called the support set of x over P.

Based on the above basic concepts, we discuss the algorithm flow as follows. Each sensor exchanges messages with its neighbors. Each sensor can apply A to compute top-K outliers for its knowledge where the *knowledge* of the sensor is its own points plus the points received in messages. Sensor p_i should send to p_j all of p_i's current outliers and their supports. If, for any of these points x, p_i cannot be certain that p_j has x (i.e., neither previous messages from p_i to p_j nor messages from p_j to p_i contain x), then x must be sent to p_j. Second, p_i may have points which would effect outliers previously sent by p_j , but these may not be accounted for in the first part. It suffices for p_i to send the support of all of the outliers in message from p_j to p_i which were not in that message. With new messages coming in, the knowledge and hence the local outliers at sensor p_i keep changing. Thus, the process is repeated iteratively. If there are potentially points at p_i that sensor p_j has not seen yet, these extra points are sent to p_j via broadcast.

In a streaming setting, when a new point is sampled, data changes at the local sensor itself. This requires that the same calculation is made as in the case of a change in knowledge of the sensor due to receiving a message. If the algorithm needs to only consider points which were sampled recently (i.e., employ a sliding window), this can be implemented by adding a time-stamp to each point when it is sampled. Each node can retire old points regardless of where they were sampled and at no communication cost at all.

Sometimes the (x, y) position of the sensor is also important for outlier detection. In such settings where data is available across space and time, one can find faulty sensors as follows.

In this setting, each sensor is associated with a data stream based on a part of a source signal received at its location. Each sensor thus receives data which is a combination of the source signal plus some noise where a part of the noise component captures the outlierness in the signal. The outlier detection process then consists of four phases: clustering, temporal outlier detection, removal of the spatial variation, and spatial outlier detection. In the beginning, sensors form a cluster and elect their cluster-head. After that, the cluster-head receives data from sensors, carries out most of the computation tasks, and broadcasts results to sensors. In the next phase, each sensor detects and recognizes any existing temporal outliers. This task can be realized by making all sensors operate simultaneously in parallel. In the third phase, it is important to remove any geometrical effect caused by the sensor distribution. The signal power reduces in proportion to the distance from the source. If this bias is not accounted for properly, sensors near the source that receive more energy might be falsely regarded as outliers, even though they are not. If the data has some bias, the distribution might be heavily tailed with outliers located near the tail of the distribution. In that case, the min and max values are removed as outliers iteratively using the property of α-stable distributions. The last phase is the spatial outlier detection using the variogram method [Cressie, 1993, pp. 69-83]. Since this method generally shows the spatial variance between sensors, outliers can be considered as those ones that deviate remarkably from the majority of the data. Note that this method [Jun et al., 2005] assumes that the outliers are uncorrelated in time and space, and can be modeled as an α-stable distribution.

4.4 SHARING MODEL PARAMETERS

[Rajasegarar et al., 2007] compare a centralized approach with the distributed approach where only the parameters of the local model are shared. In the centralized case, they collect data from all sensors at a central location. Some nonlinear mapping is then used to map the original feature space of the data to a higher-dimensional space. The mapped data feature vector in the new high-dimensional space is called the image vector. They then learn a 1-class quarter sphere SVM using the image vectors. The hypersphere is learned in the image space such that it has minimum radius R and centered at the origin and encompasses a majority of the mapped data points. Points outside the hypersphere are labeled as outliers.

In the distributed solution, [Rajasegarar et al., 2007] extend the quarter-sphere SVM scheme as follows.

- Each sensor node s_j runs the 1-class quarter sphere SVM anomaly detection algorithm on its local data and identifies the local anomalies and the local radius R_j of the quarter sphere SVM. It keeps the local radius R_j and the norms of the image vectors in memory.
- Each sensor node s_j sends its radius information R_j to its parent node s_p.
- The parent node s_p collects the radius information from its children and combines them with its own local radius. It then computes the global radius R_m, which is used for global anomaly detection. In order to identify the best strategy for global radius computation R_m that gives a comparable performance with that of centralised detection, they considered four strategies of global radius computations, namely, using the mean, median, maximum, or minimum of the combined radii from the parent node and its children.
- Parent node s_p sends the global radius R_m to its children.
- Children nodes compare the norms of their data vectors with the global radius R_m and classify them as globally anomalous or normal. A data vector x_i is identified as globally anomalous if its norm $> R_m^2$.

Their evaluation reveals that the distributed scheme achieves significant energy savings in terms of communication overhead in the network, while achieving a comparable performance to that of the centralized case.

[Otey et al., 2006] discuss the problem of identifying outliers from mixed attribute datasets. Since transmitting the entire data or even the local outliers to every node could be expensive, [Otey et al., 2006] propose that only the local model be exchanged between nodes. Their method first builds a local model with its parameters. These local models are transmitted to a central location which then combines these models to come with parameters for the global model. In the second pass the global model is broadcasted to all the nodes which then compute the outliers based on the global model. For the streaming scenario, however, a single pass algorithm is needed. Since their model is built in an incremental way, the single-pass approach can simply use the values of the model parameters computed from all previous points to compute the anomaly score of the current point. Each of the sensors can thus build and use an incremental local model. But building a global model in an incremental way is not easy.

Since the local models are continuously changing as more and more data flows in, the global model needs to be reconstructed regularly. However regularly updating the models is an expensive operation. A more efficient alternative is to exchange local outliers between sites. If all sites agree that a point is an outlier, then we can assume that the point is a global outlier. The sites only communicate when some user-specified event occurs. Examples of such events include a user's query for the global outliers, when a site finishes processing a fixed number of points, or when a site finds a fixed number of outliers. When such an event occurs, each site broadcasts all outliers it has found since the last event to all other sites. Upon receiving a broadcast, a site will check all the points it has received in the broadcast to see if they are flagged as outliers locally, and will return that information to the requesting site. If, for a given point, all sites agree that it is a local outlier, then it is flagged as a global outlier. Clearly, since there is no global computation, this is an approximate algorithm.

There are trade-offs to consider when choosing which type of event to use. For example, since outliers are rare by definition, the simplest event to use is that a site requests validation each time it receives a new local outlier. However, this approach requires a synchronization between sites for any outlier found at a local site. As an alternative, we can choose an event where a site accumulates k potential local outliers before broadcasting them for validation. In this case, there is less synchronization between the sites, but there is a greater communication cost. This increase in communication cost stems from the fact that the sites are receiving information about global outliers less often, which increase the number of local outliers that need to be broadcast at each event.

4.5 SHARING LOCAL OUTLIERS AND DATA DISTRIBUTIONS

In this approach, data distributions are also shared along with the local outliers. The goal is to identify, among all the sensor readings in a sliding window, those values that have very few near neighbors. [Subramaniam et al., 2006] propose an approach that uses kernel density estimators to approximate the sensor data distribution. Once the data distributions have been estimated, the density of the data space around each value can be computed, and therefore outlier values can be determined.

The sensor network on a 2D plane is organized in the form of a hierarchy as shown in Figure 4.2. The idea is to organize the network using overlapping virtual grids. Several tiers are defined for the grid with different levels of granularity, ranging from small local areas at the lowest tier, to the entire network area at the highest tier. At each cell at the lowest tier of the grid, there is one leader (or parent) node, that is responsible for processing the measurements of all the sensors in the cell. Moving up the hierarchy, the leader node of a cell collects values from the leader nodes of all its sub-cells in the lower level.

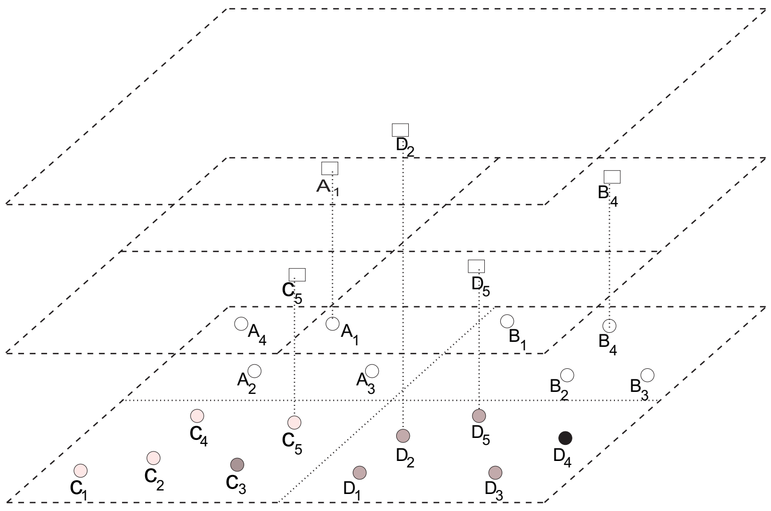

Figure 4.2: Hierarchical organization of a sensor network. Based on [Subramaniam et al., 2006].

In this work, they looked at the problem of finding the outlying values within a sliding window W that holds the last $|W|$ (d-dimensional) values in the union of the readings coming from multiple sensors. Let there be l different sensors each associated with its data stream. Each sensor can detect outliers from its own local stream. When we move up in the hierarchy, the aim is for each leader node to detect outliers within a sliding window that encompasses the measurements of all the sensors that belong to its cell. The problem of finding outliers can be solved efficiently if an accurate approximation of the data distribution can be found. In addition, the use of a data distribution approximation allows us to combine the information from many sensors efficiently, thus minimizing the communication costs required to find outliers among the values of different sensors. Hence, kernel density estimators are used to approximate the sensor data distribution. Once the data distributions are estimated, the density of the data space around each value can be computed, and thus outliers can be determined.

For kernel density estimate, the distribution for the parent node (the sample set and the bandwidth of the kernel function) can be obtained based on the parameters of the child node distributions. Thus, the data distributions can be learned for each of the nodes in the hierarchy in a distributed way. The computation starts from the child nodes and moves towards the top. An interesting question is how often a node should send its model to the leader of the cell it belongs to. The simplest approach is to have the children transmit updates to the parent as these updates take place in their own estimators. Assume that the parent has l children, each having a kernel estimator of size $|R|$, and that the kernel estimator of the parent has size $|R_p|$. Then, with probability $f = |R_p|/(l|R|)$, when a child updates its kernel estimator by adding a new kernel, it also propagates this update to its parent (i.e., it transmits the new kernel and the new standard deviation(s)). However, we can fine-tune this technique if we allow each sensor to monitor the distribution of the data it observes. When there are small changes, a sensor can reduce f, in effect reducing the update rate. When large changes are observed, the sensor can set f close to 1, thus essentially propagating its entire kernel estimator.

Based on this hierarchical computation of kernel density data distributions, they define two types of outliers: distance-based outliers and local metrics-based outliers. For distance-based outliers, the goal is to identify, among all the sensor readings in a sliding window, those values that have very few close neighbors. For density-based method, [Subramaniam et al., 2006] propose the Multi Granularity Deviation Factor (MDEF) metric. For any given value p, MDEF is a measure of how the neighborhood count of p (in its counting neighborhood) compares with that of the values in its sampling neighborhood. A value is flagged as an outlier, if its MDEF is statistically significantly different from that of the local averages.

The distributed detection of the distance-based outliers operates as follows. Given a new observation p, the sensor can use its current density distribution function $f(x)$, to estimate the number of values that are in the neighborhood of p. Specifically, the number of values, $N(p, r)$, in the time window that fall in the interval $[p - r, p + r]$. If this number is less than an application-specific threshold t then p is flagged as an outlier. It suffices for the parent node to examine only the values that have been marked as outliers by its children. All the other data values can be safely ignored, since they cannot possibly be outliers. Each time an outlying value is identified, it is transmitted to the corresponding parent node and is checked against the model of that parent node, in order to determine whether this value is also an outlier in that level of the hierarchy.

For a new observation p, a sensor can use its density estimator model to determine if p is an MDEF-based outlier with respect to its data stream as follows. MDEF-based outlier score is non-decomposable. That is, an observation detected as outlier in a parent sensor need not have been an outlier in its children sensors. Due to this reason, only the leaf sensors detect outliers. A leaf sensor can report outliers with respect to the rest of the values it is observing, as well as with respect to the values observed in an entire region in which it belongs. This is achieved by having a leader node in the higher levels of the hierarchy communicate its probability density function estimate, which is referred to as the global model, to the leaf sensor. MDEF-based outliers are then detected as follows. For every new observation that is added to the local estimator model at the lowest level sensors, the sensors transmit the observation to their leaders with probability f. The leaders receive the observations and transmit them in-turn to their parents with probability f. When a new observation is added to the kernel sample maintained at the leader node of the highest level, this update is communicated to the lowest level sensor nodes via the intermediate leaders. Thus, for every new observation a sensor at the lowest level sends, it receives $(f l)^n$ updates to its global estimator, where n is the number of levels in the hierarchy and l is the average number of children per a parent node.

The hierarchical architecture of a sensor network has also been proposed by [Palpanas et al., 2003]. However, here the hierarchy consists of only two levels comprising of high-capacity sensors and low-capacity sensors, respectively. The distribution of the sensed data is assumed to be the kernel density estimators. When the observed data points do not fit this model, they are considered as outliers.

4.6 VERTICALLY PARTITIONED DISTRIBUTED DATA

All the techniques that we discussed in this chapter until now assumed that the streaming data is partitioned horizontally, where each sensor node stores a subset of all observations. However, many applications have the observations vertically partitioned, i.e., each node stores different feature sets of the (same set of) observations. For instance, at the NASA's Distributed Active Archive Centers (DAAC), all precipitation data for all locations on the earth's surface can be at one data center while humidity observations for the same locations can be at another data center. Another example are oceanic water levels and weather conditions recorded by spatially distributed sensors over one day before a Tsunami.

[Das et al., 2011] present an algorithm that can perform outlier detection when the data is distributed across several sites, with only a subset of features at each site. They theoretically derive the true positive rate and show that the false positive rate of the algorithm is 0. For the proposed algorithm, the amount of communication required is less than 1% of that required for centralization, yet is 99% accurate compared to a centralized algorithm in finding the outliers. The accuracy is a function of the data percentage communicated and can be tuned based on the performance requirements and resources available to the users.

They assume that the vertically distributed data has a one-to-one mapping between the rows across the different distributed nodes. They use 1-class ν-SVMs for outlier detection with RBF kernels. 1-class ν-SVMs optimize the placement of the hyperplane in order to maximize the margin between the hyperplane and the origin, which is the lone representative of the outlier points.

The distributed outlier detection algorithm consists of two steps. In the first step, a local anomaly detection algorithm is executed at each node which identifies outliers based on the features present at these nodes only. Then, these local outliers from each of the nodes are collected at one central node (master node). Along with this, samples from the local nodes are also collected at the master node to build a global model. Then, all these local outliers are tested against the global model. Only those which are tagged as outliers by the global model are then output as the outliers from the distributed algorithm.

This approach cannot detect outliers which are global due to a combination of attributes. However, the algorithm shows good performance if global outliers are also local outliers. A major drawback of this approach is that the fixed-size sampling approach gives no guarantees or bounds on the correctness of the global model. For a user, it is therefore difficult to set the sampling size correctly, in advance. Moreover, during training, no other strategies than sampling are used for reducing communication costs. [Stolpe et al., 2013] refine the approach by using Vertically Distributed Core Vector Machine (VDCVM) with RBF kernels. They prove that the algorithm that uses VDCVM can have an order of magnitude lower communication cost compared to the one that uses 1-class ν-SVMs. Their methodology samples only as many points as needed, with known bounds for the correctness of the global model.

4.7 CONCLUSIONS AND SUMMARY

Outlier detection for distributed data streams has its own unique challenges: resource constraints, scale of the data, network dynamics. A large amount of data is collected today in distributed streaming form across a large number of application domains. In this chapter we discussed various mechanisms for the distributed stream setting. All the mechanisms try to address the tradeoff between the accuracy of the global outlier detection and the communication costs. We discussed various mechanisms which identify outliers accurately by sharing entire data on one extreme. On the other extreme, we discussed methods which share only model parameters infrequently and hence incur very low communication costs. We also discussed mechanisms where data is shared only with the neighbors, or local outliers are shared or data distributions are shared. Mechanisms with low communication overheads can afford to update the models more frequently and can therefore also detect outliers more quickly. Finally, we also discussed recent advances in the area of outlier detection for vertically partitioned data.

In this chapter, although we discussed outliers from a spatial as well as temporal sense, we were still interested in finding outlier points only. While we focused on the distributed nature of the data in this chapter, in the next chapter we will focus on the moving objects. In the next chapter, we will discuss outlier detection mechanisms for spatio-temporal objects whose behavioral (non-spatial and non-temporal) attributes are significantly different from those of the other objects in its spatial and temporal neighborhoods.

CHAPTER 5

Outlier Detection for Spatio-Temporal Data

In the previous chapters, we studied temporal outliers without any specific spatial continuity. In this chapter, we will study temporal outliers with spatial context. We study three main types of spatio-temporal outlier detection techniques. The first set of techniques focus on objects whose behavioral/thematic (non-spatial and non-temporal) attributes are significantly different from those of the other objects in their spatial and temporal neighborhoods. Such objects are referred to as spatio-temporal outliers or ST-Outliers. Next, we discuss detection of ST-Outlier solids which are regions of interest across time. Unlike ST-Outliers which are regions at a single time instance, ST-Outlier solids are regions across a time interval. Finally, we will discuss techniques for discovery of outlier trajectories.

5.1 SPATIO-TEMPORAL OUTLIERS (ST-OUTLIERS)

Given: A spatio-temporal dataset.
Find: ST-Outliers, i.e., spatio-temporal objects whose behavioral (non-spatial and non-temporal) attributes are significantly different from those of the other objects in their spatial and temporal neighborhoods.
Examples: Locations with significantly high wave height values on a particular date [Birant and Kut, 2006]; locations with anomalous water height [Cheng and Li, 2006]; floods and droughts from weather data (cloud cover percentage, diurnal temperature range, precipitation, temperature, vapor, and pressure) [Sun et al., 2005]; drought areas, intense fertility loss areas, hurricanes from rainfall dataset [Drosdowsky, 1993].

Some studies find outliers considering only their temporal neighbors as we have discussed in this book till now. Whereas a different class of techniques focus on outliers with respect to their spatial neighbors only. In this section, we will discuss approaches that combine these two notions. Combining the two ideas, a spatio-temporal outlier (ST-Outlier) is defined as a spatio-temporal object whose behavioral/thematic (non-spatial and non-temporal) attributes are significantly different from those of the other objects in its spatial and temporal neighborhoods. Figure 5.1 shows a typical ST-Outlier detection pipeline. Usually such methods first find spatial objects from the input data. These objects are analyzed to find spatial outliers. Further, these spatial outliers are then tracked across time to verify if they are also temporal outliers. Finally, objects which are outlier both in the spatial and temporal sense are output as spatio-temporal outliers.

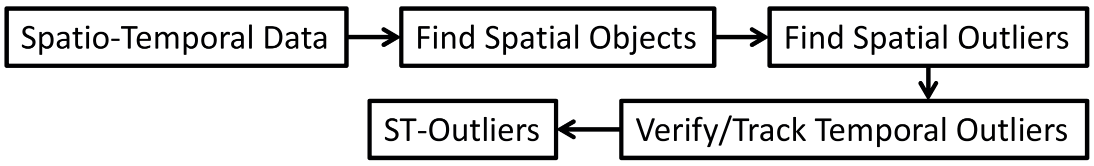

Figure 5.1: ST-outlier detection framework.

The techniques that we discuss in this section mainly differ in the way the spatial outliers are computed.

5.1.1 DENSITY-BASED OUTLIER DETECTION

[Birant and Kut, 2006] propose a density-based ST-Outlier detection mechanism. They use the DBSCAN [Ester et al., 1996] clustering algorithm to identify the spatial clusters and hence candidate spatial outliers. DBSCAN can identify clusters with arbitrary shapes and is very efficient even on large datasets. However, DBSCAN doesn't consider temporal aspects and it can't detect some outliers when clusters have different densities. Hence, they run a modified DBSCAN [Ester et al., 1996] clustering algorithm on the data with two main modifications: (1) To support the temporal aspect, a tree is traversed to find both spatial and temporal neighbors of any object within a given radius. (2) To find outliers, when clusters have different densities, the algorithm assigns a density factor to each cluster and compares the average value of a cluster with the new coming value. After clustering, potential spatial outliers are detected.

This is followed by checking the spatial neighbors to verify whether these objects are actually spatial outliers. Let o be the object that has been detected as a candidate spatial outlier using clustering. Let A be the average value of the spatial neighbors of the object o within a radius ϵ_1. Also let σ be the corresponding standard deviation. Then the object o is considered as a spatial outlier if its value does not lie in the range $[A - k_0\sigma, A + k_0\sigma]$ where $k_0 > 1$ is some pre-selected value.

The next step checks the temporal neighbors of the spatial outliers identified in the previous step. A tree is traversed to find the temporal neighbor objects of any object. In order to support temporal aspects, S-Outliers are compared to other objects of the same local area, but in different times. If the characteristic value of a spatial outlier does not have significant differences from its temporal neighbors, this is not an ST-Outlier. Otherwise, it is confirmed as an ST-Outlier.

5.1.2 OUTLIER DETECTION USING SPATIAL SCALING

[Cheng and Li, 2004] and [Cheng and Li, 2006] propose a four-step approach to address the semantic and dynamic properties of geographic phenomena for ST-Outlier detection.

- Finding Semantic Objects: The purpose of this step is to form some regions that have significant semantic meanings, i.e., the spatio-temporal objects of interest. The clustering method is designed based upon the prior knowledge and characteristics of the data. If the data are raster-based images, supervised classification might be applied or a classifier might be built based upon prior knowledge (i.e., a semantics-based approach). Other methods such as neural networks can also be applied if no prior knowledge about the data available.

- Aggregation: The main idea of this approach is to change the spatial scale of the data. If there are spatial outliers, they usually disappear if the scale of processing is reduced. It can be realized by changing the resolution of measurement so that the clustered results are different with different scales. With a decrease in scale, the difference between the objects and their spatial neighbors will decrease and the small regions which contain outliers will be removed. Therefore, aggregation here refers to the changing (decreasing) of spatial resolution (spatial scale) of the data for clustering. It is also called filtering since the outliers (noises) will be filtered after the spatial scale change.

- Comparison: In this step, the results obtained at two spatial scales are compared in order to detect the potential spatial outliers. Therefore, the results derived from Step 1 will be compared with the results derived from Step 2. The objects that are found in Step 1 and are missed (or filtered) in Step 2 are potential ST-Outliers. The comparison can be realized either by EVA (Exploratory Visualization Analysis) or VDM (Visual Data Mining). EVA methods tend to be highly interactive, providing interlinked and dynamic tools to explore the data, but without rigid control over the format that the scene might take. By contrast, VDM generally uses very specific algorithms so that uncovered objects or patterns will have a pre-defined visual appearance. In addition, EVA tends to be human-led and draws heavily from the perception literature, whereas VDM takes its cue from the numeric properties of the data and relies on statistical theory, pattern recognition and machine learning. The VDM methods are more structured and rigorous, but less flexible and perhaps less geographically intuitive.

- Verification: The outliers detected in the previous step can be suspected as ST-Outliers. According to the definition of ST-Outliers, they need to be verified since some of them might be noise and some are not, representing the real change of spatial objects. Since the natural change is usually smooth and continuous, what we see previously will be seen consecutively. Therefore, the verification checks the temporal neighbors of the suspected ST-Outliers detected in the previous step. If the attribute value of such an ST-Outlier is not significantly different from its temporal neighbors, this is not an ST-Outlier. Otherwise, it is confirmed as an ST-Outlier.

5.1.3 OUTLIER DETECTION USING VORONOI DIAGRAMS

[Adam et al., 2004] studied distance-based outlier detection for a spatio-temporal dataset. Voronoi diagrams are used to establish spatial clusters called micro-clusters. Based on the spatial and semantic similarity (computed using the corresponding time series) between two micro-clusters they can be merged to get macro-clusters. Spatial similarity for micro-cluster computation uses two measures: the Jaccard coefficient that represents similarity and the Silhouette coefficient that represents quality of clustering. Any datum that differs sufficiently from other points in the macro neighborhood is then marked as an ST-Outlier.

5.2 SPATIO-TEMPORAL OUTLIER SOLIDS

Given: A spatio-temporal dataset.
Find: ST-Outlier solids, i.e., a region across time.
Examples: Extreme precipitation events [Wu et al., 2010]; tracking hurricanes [Lu and Liang, 2004].

While previous techniques consider an ST-Outlier as a spatial region at a time instant, an ST-Outlier could be considered as a solid with its base in the XY space dimension and volume across the time dimension as shown in Figure 5.2.

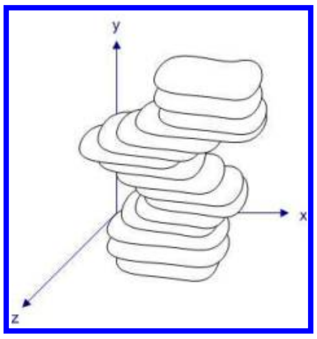

Figure 5.2: An ST-outlier solid (a moving region).

5.2.1 USING KULLDORFF SCAN STATISTIC

Before moving outliers can be discovered, however, we first need to find the spatial outliers in each time period. To identify the most significant outliers, some measure of discrepancy needs to be defined. Let the entire 2D space for the dataset be represented by U, e.g., the entire U.S. region. Let p represent a small cell and let R represent a region. A spatio-temporal dataset consists of measurement values. Thus, each p has a $m(p)$, e.g., precipitation value for a cell. This data can be deseasonalized to obtain smoothed value, $b(p)$. Let M and B represent the sum of the actual measurement and the baseline values across the entire space U. Thus, $M = \sum_{p \in U} m(p)$ and $B = \sum_{p \in U} b(p)$. Now for a region R, the measurement and the baseline values can be defined as $m_R = \sum_{p \in R} \frac{m(p)}{M}$ and $b_R = \sum_{p \in R} \frac{b(p)}{B}$. Using the m_R and b_R for the region R, one can measure the discrepancy for the region R using the Kulldorff scan statistic which is defined as follows:

$$d(m_R, b_R) = m_R \log(\frac{m_R}{b_R}) + (1 - m_R) \log(\frac{1 - m_R}{1 - b_R}) \tag{5.1}$$

if $m_R > b_R$ and 0 otherwise.

These discrepancy values can then be used to rank and compute top-K outlier regions using the Exact-Grid Top-K or the Approx-Grid Top-K algorithm. The Exact-Grid algorithm uses 4 sweep lines to find all possible different shaped regions that are located over a grid space of size $g \times g$ in $O(g^4)$ time. Kulldorff discrepancy value is computed for each region in constant time. When finding the top-K regions, we want the regions to not overlap too much. Hence, the algorithm takes a parameter which defines the maximum allowed overlap for any two regions in the top-K list of anomalous regions. The Exact-Grid Top-K algorithm is $O(g^4 k)$. The main difference between Exact-Grid Top-k and Approx-Grid Top-k is their approach to creating the test rectangles from the grid. The Approx-Grid algorithm is faster as expected with a runtime of $O(g^3 k)$.

Once the spatial outlier regions have been discovered across various time snapshots, the next step is to stitch these regions across time to obtain outlier spatio-temporal solids. [Wu et al., 2010] propose an ST-Outlier detection algorithm called Outstretch, which discovers the outlier movement patterns of the top-K spatial outliers over several time periods.

An ST-Outlier may exist over more than one time period. For example, if there is higher than average precipitation in Peru over the years 1998–2002, then the solid in three-dimensional (X, Y and $time$) space is an outlier. Using the top-K spatial outliers for each time period, they find all the sequences of outliers over time and store into a tree. The Outstretch algorithm takes as input the top-K values for each year period under analysis, and a variable r, the region stretch parameter, which is the number of grids to "stretch" by, on each side of an outlier. Outstretch then examines the top-K values of the second to last available year periods. For all the years, each of the outliers from the current year are examined to see if they are framed by any of the stretched regions from the previous year. If they are, the item is added to the end of the previous year's child list. All these lists are added to an outlier tree as soon as they are discovered. Figure 5.3 shows an example of an ST-Outlier solid region across three time points. Note the dark-blue region is the actual spatial outlier region discovered by the Exact-Grid Top-k algorithm across the three time points. The shaded region shows that the stretch parameter is set to 1. OutStretch links the three regions to form a single sequence and stores it in the outlier tree.

As a result, all possible sequences over all years get stored into the outlier tree and can be retrieved for analysis.

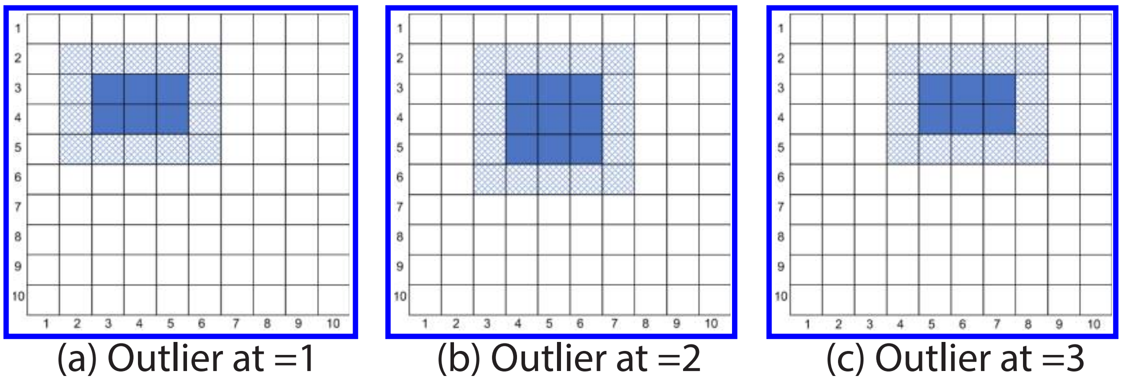

Figure 5.3: Example outstretch algorithm sequence.

In the above, a frequentist single stream approach is used to compute the Kulldorff scan statistic as a discrepancy measure. [Neill and Cooper, 2010] present the multivariate Bayesian scan statistic. The MBSS is a variant of the Kulldorff scan statistic, and has several advantages over the frequentist methods, including higher detection power, fast computation, easy interpretability of results, and ability to incorporate prior knowledge. It integrates prior information and observations from multiple data streams in a principled Bayesian framework, computing the posterior probability of each type of event in each space-time region. MBSS learns a multivariate Gamma-Poisson model from historical data, and models the effects of each event type on each stream using expert knowledge or labeled training examples.

It is important to not just look at all the streams together, but also to explore various subspaces. Recently, [Neill et al., 2013] proposed an approach for enabling computationally efficient detection of irregular space-time clusters even when the numbers of spatial locations and data streams are large. Using their algorithms, the scan statistic can be efficiently optimized over proximity-constrained subsets of locations and over all subsets of the monitored data streams, enabling timely detection of emerging events and accurate characterization of the affected locations and streams.

5.2.2 USING IMAGE PROCESSING

Similar to the above approach, [Lu and Liang, 2004] propose a wavelet fuzzy classification approach to detect and track region outliers in meteorological data. First, wavelet transform is applied to meteorological data to bring up distinct patterns that might be hidden within the original data. Then, a powerful image processing technique, edge detection with competitive fuzzy classifier, is extended to identify the boundary of region outlier. After that, to determine the center of the region outlier, the fuzzy-weighted average of the longitudes and latitudes of the boundary locations is computed. By linking the centers of the outlier regions within consecutive frames, the movement of a region outlier can be captured and traced.

5.3 TRAJECTORY OUTLIERS

In the previous section, we discussed about detecting outlier regions across time. In this section, we will discuss identification of outlier moving points (or trajectories) across time.
Given: A set of trajectories.
Find: Anomalous trajectories.
Examples: Anomalies such as vehicle moving to the left side of the road (wrong way), vehicle taking a wrong short cut, and people jaywalking [Ge et al., 2010]; traffic or road accident anomalies [Li et al., 2009].

Most of the techniques for finding trajectory outliers are based on the distance, direction and density of trajectories. However, classification and historical similarity-based techniques have also been proposed. We discuss these below.

5.3.1 DISTANCE BETWEEN TRAJECTORIES

Figure 5.4 shows five different trajectories. It is obvious that the thick portion of the trajectory TR_3 is quite different from the neighboring trajectories, though the overall behavior of TR_3 is similar to that of the neighboring trajectories.

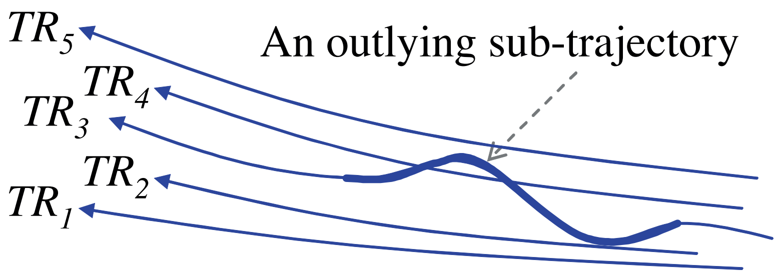

Figure 5.4: An example of any outlying sub-trajectory.

[Lee et al., 2008] propose TRAjectory Outlier Detection (TRAOD) algorithm which consists of two main phases: (1) Partitioning Phase and (2) Detection Phase. In the first phase, each trajectory is split into partitions using a 2-level partitioning. Distance between trajectory partitions is used for anomaly detection and is calculated as a weighted sum of three components: perpendicular distance ($d_\perp$), parallel distance ($d_\|$) and angle distance (d_θ). These components are intuitively illustrated in Figure 5.5.

The distance between two line segments L_i and L_j is defined as follows:

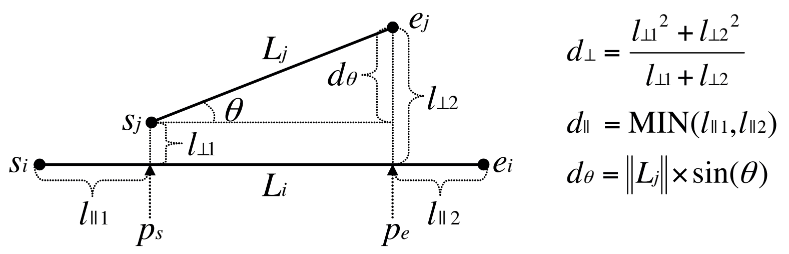
e_j
L_j
d_θ
$l_{\perp 2}$
s_j
θ
$l_{\perp 1}$
s_i
L_i
e_i
$l_{\parallel 1}$
p_s
p_e
$l_{\parallel 2}$
$d_\perp = \frac{l_{\perp 1}^2 + l_{\perp 2}^2}{l_{\perp 1} + l_{\perp 2}}$
$d_\parallel = \mathrm{MIN}(l_{\parallel 1}, l_{\parallel 2})$
$d_\theta = \|L_j\| \times \sin(\theta)$

Figure 5.5: Three components of the distance function for line segments.

$$dist(L_i, L_j) = w_\perp d_{perp}(L_i, L_j) + w_\parallel d_\parallel(L_i, L_j) + w_\theta d_\theta(L_i, L_j). \tag{5.2}$$

The weights above are application dependent. By appropriately fixing the weights, two types of trajectory outliers can be detected.

- Positional outlier: The location of a trajectory is different from those of neighboring trajectories.
- Angular outlier: The direction of a trajectory is different from those of neighboring trajectories.

The two-level partitioning is performed as follows. At the first level, each trajectory is partitioned into coarse t-partitions. At the second level, coarse t-partitions that are likely to be outlying are selected, and then, only selected ones are partitioned into fine t-partitions. In this way, we narrow the search space that needs to be inspected in fine granularity (i.e., by partitioning a trajectory at a base unit). As a result, many portions of trajectories can be pruned early on. The coarse partitioning is performed using the Minimum Description Length (MDL) principle.

In the second phase, a trajectory partition is considered outlying if it does not have sufficient number of similar neighbors (with appropriate considerations of local density). A trajectory is marked as an outlier if the sum of the length of its outlying partitions is at least F times the sum of lengths of all of its partitions.

5.3.2 DIRECTION AND DENSITY OF TRAJECTORIES

[Ge et al., 2010] consider two types of outlying trajectories: outliers in terms of direction and outliers in terms of density. The continuous space is discretized into small grids (as shown in Figure 5.6) and a probabilistic model is used to turn the direction information of trajectories in a grid into a vector with eight values to indicate the probabilities of moving towards eight directions within this grid. The direction trend is thus generated within a fixed area by summarizing the moving directions from large amounts of trajectories for a period of time. Then, once some objects move across this area along the completely different directions from the summarized directions, they can be labeled as outliers in a real-time fashion by measuring the similarity between the directions of the observed objects and the summarized directions.

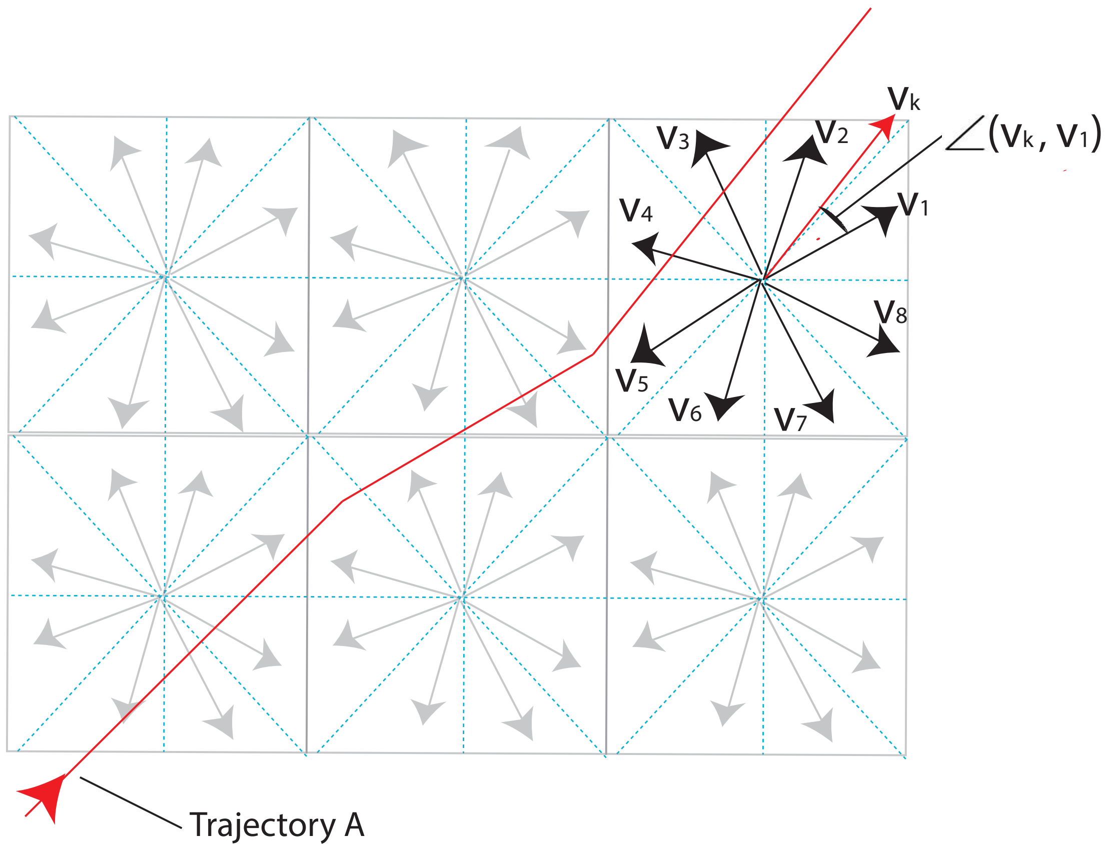

Figure 5.6: Grid and the direction vectors.

Also, with this discretized space, the density of trajectories can be computed in each grid conveniently. The trajectory density within each grid is estimated as the number of trajectories across this grid. The trajectory density distribution can be obtained with sufficient historical trajectory data. The outlying score of a new trajectory can then be measured based on the density of trajectories in the grids where this trajectory actually passes.

5.3.3 HISTORICAL SIMILARITY

While the above techniques stress on the spatial continuity to define outliers, [Li et al., 2009] propose a method for detecting temporal outliers with an emphasis on historical similarity trends between data points. They work with a dataset of road segments in San Francisco area. At each time step, each road segment checks its similarity with the other road segments, and the historical similarity values are recorded in a temporal neighborhood vector at each road segment. Outliers are calculated from drastic changes in these vectors. Given a road segment with a historically stable set of neighbors (in feature space, not physical space), an outlier is loosely defined as a drastic change in the membership of this set. A change is noted heavily for a pair of road segments if the two road segments are historically similar but dis-similar at this time instant, or if the two road segments are not historically similar but similar at this time instant. Based on this, each road segment is given an exponential reward or penalty each day. The outlier score of an road segment on a particular day is then equal to the sum of rewards and penalties across all other road segments. The power of this method compared with a method that measures only the singular road segment is that it is robust to population shifts.

5.3.4 TRAJECTORY MOTIFS

Instead of viewing the movement path of an object as a sequence of low-level spatio-temporal points, one can also view it as a sequence of movement motifs. A movement motif is a prototypical movement pattern of an object, often pre-defined by domain experts. Typical examples include straight line, right turn, u-turn, loop, near-an-island, etc. [Li et al., 2006] propose a motion-classifier for trajectory outlier detection, which uses such movement motifs and consists of the following three steps.

- Object movement features, called motifs, are extracted from the object paths. Each path consists of a sequence of motif expressions, associated with the values related to time and location. For example, a path could look as follows. $(Right-Turn, 3am, l_7), (U-Turn, 4pm, l_2), (U-Turn, 10pm, l_1)$ where l_1, l_2, and l_7 are locations. Besides time and location, the motif expressions could also be associated with other attributes like average speed, maximum speed, etc.

- To discover anomalies in object movements, motif-based generalization is performed to cluster similar object movement fragments and generalize the movements based on the associated motifs. This involves generalization of the attribute values. For example, for the time of the day feature, the above path can be generalized to $(Right-Turn, Night, l_7), (U-Turn, afternoon, l_2), (U-Turn, dusk, l_1)$.

- With motif-based generalization, objects are put into a multi-level feature space and are classified by a classifier that can handle high-dimensional feature spaces to discriminate anomalous trajectories from normal ones.

5.4 CONCLUSIONS AND SUMMARY

In this chapter, we studied three main kinds of techniques to identify temporal outliers in the spatial sense. Techniques to identify spatio-temporal (or ST) outliers differ mainly in the way in which spatial outlier detection is performed. We studied three ways of performing ST-Outlier Detection: density-based, using spatial scaling and using Voronoi diagrams. Next, we studied techniques for identification of ST-Outlier solids. Kulldorff Scan Statistic and image processing edge detection mechanisms are used for detecting outlier regions which are then stitched across time to obtain ST-Outlier solids. Finally, we briefly studied the literature on trajectory outlier detection. Trajectory outliers can be discovered by computing the distance, direction, and density of trajectories. Besides these we also visited the historical similarity-based and motif generalization-based trajectory outlier detection techniques in this chapter.

CHAPTER 6
Outlier Detection for Temporal Network Data

For a large number of applications, systems can be expressed using information networks. Compared to the traditional modeling as multi-dimensional datasets (often as relational databases), the networks (or graphs) can capture much more semantics using links between the tuples. Though effective outlier detection techniques exist for multi-dimensional data, new techniques need to be designed for networks which present a different set of challenges. Besides links, the networks may also have a temporal aspect. Indeed, a large number of graphs are inherently temporal. For example, in social networks, new users join, new friendships are created. In bibliographic networks, new authors publish more papers, more collaborations are done. In transportation or road networks, new roads are constructed. In ad hoc networks like the network of communication devices in a battlefield, devices keep changing their positions rapidly, new messages are transmitted. While some networks change slowly, other networks show rapid changes.

In this chapter, we will answer the following interesting questions on outlier detection for temporal graph data. Given a stream of graphs, how does one identify an outlier graph? Given a temporal network with community distributions for each of its nodes, how does one track community evolution, and thereby define community outliers?

6.1 OUTLIER GRAPHS FROM GRAPH TIME SERIES

Given: A series of graph snapshots.
Find: Outlier graph snapshots.
Examples: Given multiple crawls of the web graph, find an anomalous crawl graph [Papadimitriou et al., 2010], [Papadimitriou et al., 2008]; anomalous snapshots of a TCP/IP traffic network [Pincombe, 2005]; sudden changes in edge weights on a network indicating the beginning of a Denial of Service attack or malfunctioning of a device or service in the network [Kapsabelis et al., 2007]; anomalous behavior of email users like excessive chatter, aliasing from Enron email network [Priebe et al., 2005].

Various graph distance metrics can be used to create time series of network changes by sequentially comparing graphs from adjacent periods. These time series are individually modeled as univariate autoregressive moving average (ARMA) processes. Residuals exceeding a threshold are used to label particular changes as anomalous. Let us begin by introducing a few notations. Let V be the set of nodes when both the graph snapshots G and H contain the same set of nodes, else

V_G and V_H are used to denote the respective vertex sets of the two graphs. E_G and E_H are edges in graphs G and H. Let $w_G(u, v)$ and $w_H(u, v)$ denote the edge weights for edge (u, v) in the graphs G and H, respectively. In this section, we discuss various distance or similarity measures to compute distance between two graphs G and H.

6.1.1 WEIGHT INDEPENDENT METRICS

These are basic metrics related only to the edges and vertices in the graph without using the edge weights or the vertex weights.

- MCS Edge Distance [Pincombe, 2005]: The maximum common subgraph (MCS) F of G and H is the common subgraph with the most vertices. MCS edge distance is defined in terms of the number of edges in the MCS as follows:

$$d(G, H) = 1 - \frac{|mcs(E_G, E_H)|}{\max(|E_G|, |E_H|)}. \tag{6.1}$$

- MCS Vertex Distance [Pincombe, 2005]: It is defined in a similar way as in the MCS edge distance, but vertex sets of the two graphs rather than edge sets are used in the distance computation:

$$d(G, H) = 1 - \frac{|mcs(V_G, V_H)|}{\max(|V_G|, |V_H|)}. \tag{6.2}$$

- Graph Edit Distance [Pincombe, 2005], [Papadimitriou et al., 2008], [Shoubridge et al., 1999]: The simplest form of graph edit distance is given by

$$d(G, H) = |V_G| + |V_H| - 2|V_G \cap V_H| + |E_G| + |E_H| - 2|E_G \cap E_H|. \tag{6.3}$$

This captures the sequence of edit operations required to make graph G isomorphic to graph H. Graph edit distance can be expressed in other ways where the costs of inserting/deleting/substituting a node/edge are different [Kapsabelis et al., 2007].

$$d_1(G, H) = \sum_{n \in V_G \backslash (V_G \cap V_H)} C_{nd}(n) + \sum_{n \in V_H \backslash (V_G \cap V_H)} C_{ni}(n) + \sum_{e \in E_G \cap E_H} C_{es}(e) + \sum_{e \in E_G \backslash (E_G \cap E_H)} C_{ed}(e) + \sum_{e \in E_H \backslash (E_G \cap E_H)} C_{ei}(e) \tag{6.4}$$

where $C_{nd}(n)$ is the cost of deleting a node n, $C_{ni}(n)$ is the cost of inserting a node n, $C_{es}(e)$ is the cost of substituting an edge weight for an edge e, $C_{ed}(e)$ is the cost of deleting an edge e, $C_{ei}(e)$ is the cost of inserting an edge e. In the above a unity cost for any node edit operation is applied.

- Median Graph Edit Distance [Pincombe, 2005], [Dickinson et al., 2002]: Median graph can be computed for a fixed window of graphs using graph edit distance. Median graph edit distance is the edit distance of the graph from the median graph.

- Diameter Distance [Pincombe, 2005], [Gaston et al., 2006]: The diameter distance between graphs G and H is the difference in the diameters for each graph.

- Vertex/edge overlap (VEO) [Papadimitriou et al., 2010]: Two graphs are similar if they share many vertices and/or edges. The Jaccard index is one way to compute the overlap between two graphs. It is defined as the length of intersections of vertices/edges divided by that of the union:

$$sim_{VEO}(G, H) = \frac{|V_G \cap V_H| + |E_G \cap E_H|}{|V_G| + |V_H| + |E_G| + |E_H|}. \tag{6.5}$$

6.1.2 METRICS USING EDGE WEIGHTS

The following metrics also incorporate the edge weight information along with the edge sets and the vertex sets of the two graphs.

- Weight Distance [Pincombe, 2005]: It is a function of the difference between the edge weights of the graphs G and H normalized by the maximum of the edge weights in G or H.

$$d(G, H) = \frac{\sum_{u,v \in V} \frac{|w_G(u,v) - w_H(u,v)|}{\max(w_G(u,v), w_H(u,v))}}{|E_G \cup E_H|} \tag{6.6}$$

- MCS Weight Distance [Pincombe, 2005]: It computes weight distance based on the edges in MCS where the maximum common subgraph (MCS) F of G and H is the common subgraph with the most vertices.

- Edge-Weighted Graph Edit Distance [Pincombe, 2005], [Papadimitriou et al., 2008], [Shoubridge et al., 1999]: This is a variant of the graph edit distance which includes cost of change in edge weights. Parameter c helps to control the relative importance of node edits vs. edge edits as follows:

$$d_2(G,H) = c(|V_G| + |V_H| - 2|V_G \cap V_H|) + \sum_{e \in E_G \cap E_H} |w_G(e) - w_H(e)| + \sum_{e \in E_G \setminus (E_G \cap E_H)} w_G(e) + \sum_{e \in E_H \setminus (E_G \cap E_H)} w_H(e). \tag{6.7}$$

To accentuate relatively large changes in edge weights and hide small changes due to typical traffic variations, the following can be used with $\epsilon = 1$ to avoid division by 0:

$$d_3(G,H) = c(|V_G| + |V_H| - 2|V_G \cap V_H|) + \sum_{e \in E_G \cap E_H} \frac{|(w_G(e) + \epsilon) - (w_H(e) + \epsilon)|^2}{(\min(w_G(e), w_H(e)) + \epsilon)^2} + \sum_{e \in E_G \setminus (E_G \cap E_H)} (w_G(e) + \epsilon)^2 + \sum_{e \in E_H \setminus (E_G \cap E_H)} (w_H(e) + \epsilon)^2. \tag{6.8}$$

- Modality Distance [Pincombe, 2005]: The Modality distance between graphs G and H is the absolute value of the difference between the Perron vectors of these graphs. Perron vector is the eigen vector corresponding to the largest eigen value for the adjacency matrix of the corresponding graph.
- Entropy Distance [Pincombe, 2005], [Gaston et al., 2006]: The Entropy distance between graphs G and H is defined using entropy-like measures associated with edge weights for the corresponding graphs.

  Entropy of a graph G can be defined in terms of its edge weight distribution as follows:

$$E(G) = -\sum_{e \in E_G} (\hat{w}_G(e) - ln\hat{w}_G(e)) \tag{6.9}$$

where

$$\hat{w}_G(e) = \frac{w_G(e)}{\sum_{e \in E_G} w_G(e)}. \tag{6.10}$$

- Spectral Distance [Pincombe, 2005], [Gaston et al., 2006]: The Spectral distance between graphs G and H is defined in terms of the k largest eigenvalues of the Laplacian for the graphs G and H, respectively:

$$d(G,H) = \sqrt{\frac{\sum_{i=1}^{k} (\lambda_i - \mu_i)^2}{\min(\sum_{i=1}^{k} \lambda_i^2, \sum_{i=1}^{k} \mu_i^2)}}, \tag{6.11}$$

where λ and μ are the eigenvalues for graphs G and H, respectively, in the non-ascending order of their magnitudes.

- Umeyama graph distance [Dickinson and Kraetzl, 2003]: The Umeyama graph distance is defined as the sum of the squares of the differences between edge weights for the corresponding edges of the graphs G and H, respectively:

$$dist(G, H) = \sum_{u,v \in V} [w_G(u, v) - w_H(u, v)]^2. \tag{6.12}$$

For TCP/IP traffic data, it was observed that time series based on the MCS edge, MCS vertex, edit, median and entropy metrics had no false positives. However, the time series based on the spectral, modality and diameter distance metrics are considerably more computationally intensive and also did not lend themselves to accurate ARMA modelling [Pincombe, 2005].

6.1.3 METRICS USING VERTEX WEIGHTS

While the metrics in the previous subsection gave specific importance to the edge weights, the following metrics have been designed to incorporate the vertex weight information.

- Vector Similarity [Papadimitriou et al., 2008, 2010]: This family uses the rule that "two graphs are similar if their node/edge/graph weight vectors are close." In general, the weights represent quality. Let q_i be the quality score (e.g., Pagerank score) of vertex v_i in graph G. Then we can compare the quality vectors for both the graphs by computing the average difference between q_G and q_H for all i.

  Another way of defining this measure is by considering statistics of the graph to generate the vector. For example, by computing the Euclidean distance between the principal eigenvectors of the graph adjacency matrices of the two graphs.

  Another method described as follows could be used to use weight vectors for edges. For example, each edge can be given a weight which captures its importance relative to the local topology. As a base, for edge (u, v), compute weight $\#outlinks(u, v)$ as the number of outlinks from u to v. Then compute $\gamma(u, v)$ to capture the relative importance of this edge to other edges leaving node u as follows:

$$\gamma(u, v) = \frac{q_u \times \#outlinks(u, v)}{\sum_{\{v' : (u, v') \in E\}} \#outlinks(u, v')}. \tag{6.13}$$

  This approach assigns higher weights to edges that start from high quality vertices. Using these γ weights, the similarity between two graphs G and H can be calculated as follows:

$$sim_{VS}(G, H) = 1 - \frac{\sum_{(u,v) \in E_G \cup E_H} \frac{|\gamma_G(u,v) - \gamma_H(u,v)|}{\max(\gamma_G(u,v), \gamma_H(u,v))}}{|E_G \cup E_H|}. \tag{6.14}$$

- Vertex ranking (VR) [Papadimitriou et al., 2008, 2010]: Two graphs are similar if the rankings of their vertices are similar. The vertices can be ranked using their qualities, and the similarity of rankings can be computed using a rank correlation method such as the following modified version of the Spearman's correlation coefficient:

  $$sim_{VR}(G, H) = 1 - \frac{2 \sum_{v \in V_G \cup V_H} w_v \times (\pi_G(v) - \pi_H(v))^2}{D}, \tag{6.15}$$

  where $\pi_G(v)$ and $\pi_H(v)$ are the ranks of v in the sorted list for G and H, respectively, w_v is the quality of v, and D is a normalization factor that limits the maximum value of the fraction to 1.

  The modified version satisfies the constraints that (1) the rank correlation needs to be sensitive to quality and (2) it needs to work for partial lists, i.e., the lists that are not permutations of each other. If a vertex exists in both graphs, its final quality is the average of its qualities from both graphs; otherwise, its final quality is equal to its quality in whichever graph it exists. Moreover, if a vertex does not exist in one graph, say, H, its rank is equal to $|V_H| + 1$. Vertices not in G are handled analogously.

- Sequence similarity [Papadimitriou et al., 2010], [Papadimitriou et al., 2008]: It is the similarity of vertex sequences of the graphs that are obtained through a graph serialization algorithm. Such an algorithm creates a serial ordering of graph vertices which is maximally edge connected. That means that it maximizes the number of vertices that are consecutive in the ordering and are edge connected in the graph.

  One way to generate such a serial ordering from a graph is to use the following walk algorithm. The walk algorithm starts off by visiting the vertex associated with the highest quality, and repeats the following process: (i) among the unvisited neighbors reachable from the currently visited vertex, visit the neighbor with the highest quality; and (ii) if the currently visited vertex does not have any unvisited neighbors, jump to the vertex that has the highest quality among all the unvisited vertices.

  Once both the graphs have been converted to sequences using the walk algorithm, shingling and minhashing is used to compute similarity between the sequences.

- Signature similarity [Papadimitriou et al., 2010], [Papadimitriou et al., 2008]: Two objects are similar if their signatures are similar. Instead of converting a graph into a sequence, for signature similarity, the graph is converted into a set of features.

Vertices and edges of the graph are selected as features. The weights of these features are assigned as follows. Each vertex v is assigned its quality score and each edge (u, v) is assigned the quality of vertex u normalized by the number of its outlinks. This weighted feature set is then randomly projected into a smaller-dimensionality feature space (signatures) using the SimHash algorithm. This small dimensionality representation is essentially a b-bit vector. Finally, similarity between graphs G and H is computed as follows:

$$sim_{SimHash}(G, H) = 1 - \frac{Hamming(f_G, f_H)}{b}, \tag{6.16}$$

where f_G and f_H are the b-bit vectors corresponding to the graphs G and H, respectively. $Hamming(f_G, f_H)$ is the number of bit positions in f_G and f_H for which the corresponding bits are different.

Among vertex ranking, vector similarity, vertex/edge overlap, sequence similarity, and signature similarity, [Papadimitriou et al., 2008] observed that signature similarity and vector similarity were the best at detecting anomalies from web crawl graphs while not yielding false positives.

6.1.4 SCAN STATISTICS

Besides the network outliers discovered using the above measures, Priebe et al. [Priebe et al., 2005] propose a method based on scan statistics for outlier detection in graph streams. They first define locality statistic at a node v in a graph as any statistic derived from its k-hop neighborhood. For example, the number of edges, density, domination number, etc. Across all nodes one can find the maximum locality statistic. Further, one can vary k of k-hops and get the maximum value of the locality statistic across all nodes and across all hops. This is called as scan statistic. The approach is then to check whether this scan statistic is more than a threshold value for a particular snapshot. If there is a graph x for which $M(x)$ is greater than the threshold, then there is an anomaly according to the scan statistics method.

6.2 MULTI-LEVEL OUTLIER DETECTION FROM GRAPH SNAPSHOTS

In the previous section, we discussed a variety of metrics used to identify particular outlier graph snapshots given a series of graphs. In this section, we will focus on identify outlier parts of graphs given multiple graph snapshots. In the first part, we will discuss in brief the Metric Forensics system which identifies various interesting phenomena in graphs at various levels. In the second part, we will discuss a mechanism to identify top outlier node pairs with maximum shortest path distance changes.
Given: A series of graph snapshots.
Find: Outlier graphs or outlier parts of particular graphs in the series.

Examples: Snapshots with broken correlations and prolonged spikes of properties of networks [Henderson et al., 2010]; top shortest path distance change node pairs across snapshots [Gupta et al., 2011a]; outlier links from a network [Qi et al., 2012].

6.2.1 ELBOWS, BROKEN CORRELATIONS, PROLONGED SPIKES, AND LIGHTWEIGHT STARS

[Henderson et al., 2010] propose a system called MetricForensics that uses a collection of global, local, and community level graph metrics to find four novel phenomena in such graphs: elbows, broken correlations, prolonged spikes, and lightweight stars. MetricForensics combines a multi-level "drill down" approach, a collection of user-selected graph metrics, and a collection of analysis techniques. At each successive level, more sophisticated metrics are computed and the graph is viewed at finer temporal resolutions. In this way, MetricForensics scales to highly volatile graphs by only allocating resources for computationally expensive analysis when an interesting event is discovered at a coarser resolution first.

The system first creates a series of summary graphs using some "aggregation" (binary/sum/max) over edge weights of different snapshots in that time interval. At various levels, MetricForensics computes and monitors a suite of graph metrics as follows.

- Global metrics
  - Basic Metrics like number of active vertices, number of active edges, average vertex degree, average edge weight, and maximum vertex degree.
  - Connectivity Metrics like number of connected components, fraction of vertices in the largest component, number of articulation points, minimum spanning tree weight.
  - Spectral Metrics like top k eigenvalues of the adjacency matrix.
  - Stability Metrics like $Jaccard(V_T, V_{T-1})$, $Jaccard(E_T, E_{T-1})$.
- Community metrics: This includes static metrics like fraction of vertices in the largest community and number of communities. Dynamic community metrics include variation of information between successive community assignments, cross associations, etc.
- Local metrics: This includes centrality metrics, OddBall metrics [Akoglu et al., 2010], and impact metrics, e.g., leaving a single vertex out of the graph and recalculating other metrics to determine the impact of the vertex.

The MetricForensics system is equipped with a large collection of analysis techniques which use the above set of metrics to extract interesting graph phenomena. The list of techniques available in the system are as follows:

- Single metric analysis.
  - Autoregressive Moving Average (ARMA) Model to identify metric values that are abnormally large or small given recent values.
  - Fourier analysis can identify periodic behavior, such as daily trends in graph properties.
  - Lag plots.
  - Wavelet analysis to identify patterns and anomalies in metric values.
  - Outlier detection techniques such as Local Outlier Factor and fractal dimension analysis
- Coupled metric analysis
  - Pearson Correlation analysis.
  - Outlier detection or clustering on coupled metric data.
- Non-metric analysis
  - Visualization (3D display of summary graphs) tools: The size of a vertex can show its degree, while the color can depict the vertex between-ness centrality.
  - Attribute data inspection: Vertices and edges in volatile graphs can have attributes. For example, IP communication traces often have at least partial packet contents. Outliers can be discovered by inspecting such attribute values.

Based on the above metrics and the metric analysis techniques, the system is capable of identifying the following interesting phenomena at various scales in the graph snapshots.

- "Elbows": These are graph level outliers where the observed behavior changes while another phenomenon remains stable. For example, consider eigen analysis of the graph adjacency matrix. The large eigenvalues of a weighted graph typically correspond to either a single heavy edge, a vertex with high weighted degree, or a component with a large total weight. Thus, when we see a period of time when λ_1 is changing but λ_2 is steady, it is a result of the currently dominant phenomenon changing while the secondary phenomenon is stable (e.g., a single heavy edge changing weight while the structure of the giant component is steady). This is called as the "elbow" pattern because it appears as elbow-like structures when λ_1 is plotted against λ_2.
- Broken correlations: These are graph level outliers where previously observed strong correlations disappear. For example, one may observe that normally λ_1 is highly correlated with maximum edge weight. But if a certain time point, this no longer holds, an alarm can be raised.

- Prolonged spikes: These are again graph level outliers where there is low volume but prolonged activity-level. To quantify burstiness or spikes, they use a measure called fraction dimension. Fractal dimension is around 1 for a uniform distribution and 0 for a perfect spike, i.e., when all the points in the collection are at the same time-tick. They compute the fractal dimensions of disjoint fixed width windows of several graph metrics on the summary graphs. A sudden drop in the fractal dimension value indicates a prolonged spike.

- "Lightweight" stars: These are vertices that form very big star-like structures but have lower than expected total incident edge-weights. Such outlying vertices can be discovered by tracking the OddBall metrics [Akoglu et al., 2010] over the 1-hop neighborhood of every vertex.

6.2.2 OUTLIER NODE PAIRS

For the case of static data, outlier node pairs correspond to inconsistent pairs of nodes. For example, two connected nodes which contain inconsistent content can be viewed as outlier node pairs. The discovery of such node pairs also leads to the discovery of outlier links. For example, the work in [Qi et al., 2012] uses conditional random fields to discover such outlier links in static social media data. These techniques are however not designed for dynamic networks, in which the level of structural *change* is more relevant.

The evolution of information networks may result in changes in important structural properties such as pairwise shortest-path distances. One of the interesting changes is the sudden change in the shortest path distance in an evolutionary network. 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, 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 which connects a pair of nodes at a distance of 5 is an unusual event, and most likely reflects the sudden collaboration between a pair of authors in different topical areas. Therefore, it is interesting and useful to determine the top-k shortest path distance changes in an evolutionary network.

The problem of finding such shortest path change outlier node pairs can be solved directly by running well known all-pair shortest path algorithms and simply comparing the distance changes between all pairs. However, this is not a practical solution for very large graphs. For example, for a graph containing 10^8 nodes, the number of possible node pairs would be 10^{16}. The complexity of the all-pairs shortest path computation increases at least quadratically [Ahuja et al., 1993] with the number of nodes. Furthermore, the storage of such pairwise paths can be impractical. While this has not been a problem with the small memory-resident graphs which are frequently used with conventional algorithms, it is much more challenging for the large-scale networks which arise in social and information network scenarios.

One of the keys to determining the node pairs with the most change is to determine the critical edges which lie on the shortest paths between many pairs of nodes. Clearly, the addition of such edges can lead to tremendous changes in the shortest path distances. Therefore, edge importance is defined as the probability that an edge will belong to some shortest path tree. [Gupta et al., 2011a] first propose a randomized algorithm for edge importance estimation.

This notion of edge importance is then used in an algorithm called the Incidence Algorithm. The intuition for the Incidence Algorithm is that the maximum distance change node pairs will include at least one node on which new edges or edges with changed weights are incident, with high probability.

Let the active set represent the set of nodes with new edges or edges with changed weights. While nodes in the active set cover most of the top outlier node pairs with maximum shortest path distance change, the recall of such a method can be improved by selectively expanding the active set. The active set is expanded to include neighbors of the current nodes in the active set such that neighbors connected by highly important edges are included.

The nodes selected by the Incidence Algorithm are then used as the seed set to run Dijkstra's algorithm on both graph snapshots. The node pairs within maximum shortest path distance change are returned as outlier node pairs. If the snapshots of the graph are taken after long intervals, new edges would be incident on a large percentage of the nodes in the graph. The efficiency of our Incidence Algorithm is dependent upon the size of this set. Hence, when solving the problem over snapshots of a graph taken over longer time intervals, the Incidence Algorithm would be computationally expensive. Hence, strategies are needed to rank the nodes in the active node set so that the order helps to select the top few nodes which can be used to run single-source-based shortest path algorithms and capture the maximum distance change pairs. The trade-off is between the number of nodes selected and accuracy. The goal is to rank the nodes, so that by processing them one by one we obtain more accurate results by running shortest path algorithms from a very small number of source nodes. [Gupta et al., 2011a] propose various heuristics to rank nodes including edge weight-based ranking, edge weight change-based ranking, edge importance-based ranking, edge importance change-based ranking, and clustering-based ranking. The clustering-based ranking uses the intuition that a new inter-cluster edge with low weight would be more important in reducing the shortest path distance between a pair of nodes compared to an intra-cluster edge. Results on various synthetic datasets show that the clustering-based algorithm is the most effective algorithm in computing top outlier node pairs.

6.3 COMMUNITY-BASED OUTLIER DETECTION ALGORITHMS

Given: A series of graph snapshots.
Find: Outlier nodes with anomalous temporal community changes or outlier graphs with sudden changes.
Examples: Objects that do not follow the popular community distribution change trends [Gupta et al., 2012b]; anomalous community changes across multiple time snapshots [Gupta et al., 2012a]; change detection in bipartite graphs [Sun et al., 2007].

6.3.1 COMMUNITY OUTLIERS USING COMMUNITY CHANGE PATTERNS

Community-based outliers have been studied for static graphs, both for homogeneous networks [Gao et al., 2010] as well as for heterogeneous networks [Gupta et al., 2013d]. Recently, there has been work on community outliers for temporal data.

Given two snapshots of a network, the communities evolve across the two snapshots. While most objects follow the popular community distribution change trends, some objects do not. Such rare objects are called Evolutionary Community Outliers (ECOutliers) [Gupta et al., 2012b]. Figure 6.1 shows a typical framework to discover community outliers from network data. To begin with, communities can be discovered independently in both snapshots. To discover ECOutliers, it is important to first identify the usual transition trends of community evolution. A key problem in discovering the community transition trends is computing matching communities across both snapshots. ECOutlier detection can then be performed using these patterns. However, community matching itself suffers from ECOutliers and hence must be done in an outlier-aware way. The framework discussed in [Gupta et al., 2012b] integrates outlier detection and community matching together in an iterative manner. Such an integrated formulation performs much better than performing outlier detection after community matching separately.

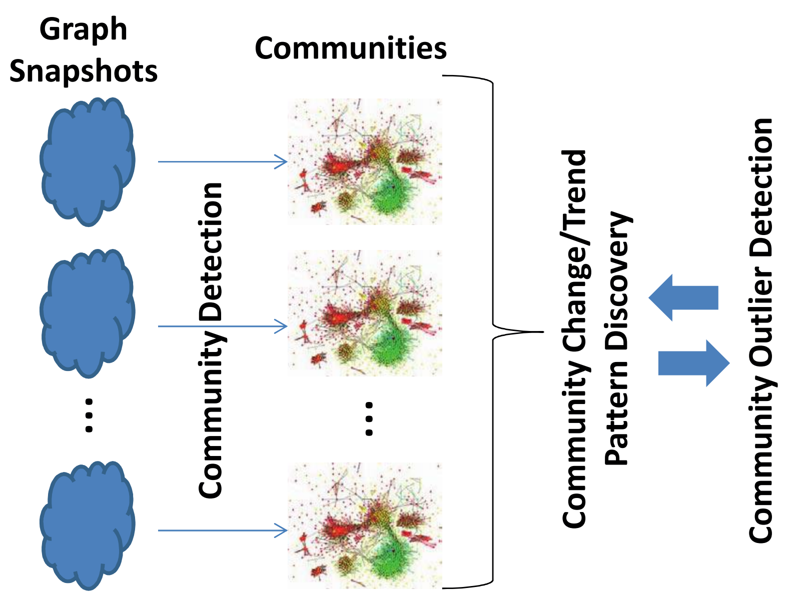
Graph Snapshots
Communities
Community Detection
Community Change/Trend Pattern Discovery
Community Outlier Detection

Figure 6.1: A framework for community outlier detection.

Moreover, the authors investigated the task of outlier detection in a general setting of multiple network snapshots [Gupta et al., 2012a]. Such outliers are called Community Trend

Outliers (CTOutliers). This study exploits the subtle temporal community evolution trends to perform outlier detection. While ECOutliers [Gupta et al., 2012b] capture the notion of anomalous community transitions, CTOutliers [Gupta et al., 2012a] capture anomalous community changes across multiple time snapshots, and in a more subtle way. Two main challenges were (1) computing patterns from a sequence of community membership probability distributions (soft sequences) associated with every object for a series of timestamps, and (2) computing the outlier score of individual soft sequences based on such patterns. The authors proposed an effective two-step procedure to detect community trend outliers. They first modeled the normal evolutionary behavior of communities across time using soft patterns discovered from the dataset. In the second step, they proposed effective measures to evaluate probability of an object deviating from the normal evolutionary patterns.

6.3.2 CHANGE DETECTION USING MINIMUM DESCRIPTION LENGTH

A method known as GraphScope [Sun et al., 2007] performs change detection in bipartite graphs. In bipartite graphs, it is assumed that all edges exist between a source set of nodes and a destination set of nodes. [Sun et al., 2007] use a combination of dynamic community detection and information theoretic methods like minimum description length (MDL) to determine the key change points in the underlying data. Intuitively, a change point is one which significantly increases the encoding cost in order to represent the series of graph snapshots. The approach groups similar sources together into source groups, and similar destinations together into destination groups. The most similar source-partitions for a given source node is the one that leads to small encoding cost. A similar definition is applied to the destination-partitions. 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. 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. This corresponds to a high level of change in the patterns of the underlying network. Such change points correspond to drastic discontinuities in the network, which can be regarded as temporal outliers.

6.3.3 COMMUNITY OUTLIERS USING EVOLUTIONARY CLUSTERING

Evolutionary clustering is a form of clustering on a series of temporal data such that snapshot clustering quality is maximized for each of the individual snapshots, and there exists some correspondence between the clusters in consecutive snapshots. This correspondence helps in analyzing clusters for various applications. [Gupta et al., 2011b] perform evolutionary clustering for networks. On each snapshot, a probabilistic assignment is learned with the use of the NetClus algorithm [Sun et al., 2009]. 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 measuring significant changes in the cluster behavior. Such changes can be used in order to quantify the outlier score of the underlying temporal patterns. The outlier score can be quantified at a variety of levels, and may correspond to node-specific values, cluster-specific values or global values over the whole data. This work suggests that temporal community structure analysis exposes several global and local structural properties of networks, which can be quantified in the form of temporal time series. Significant deviations in these values can be reported as anomalous changes in the network. A subset of the more important temporal change quantifications are as follows.

- Cluster Membership Consistency: Evolutionary clustering provides a soft clustering for each object for every snapshot. Use cosine similarity between cluster distribution for an object at time t and time $t+1$, one can identify change in its cluster membership. This can also be done at the entire graph level and sudden changes can be flagged as anomalies.

- Cluster Snapshot Quality: This is defined as the ratio of the intra-cluster similarity to inter-cluster similarity. Sudden change in cluster snapshot quality for a particular graph snapshot can be suspicious.

- Cluster Novelties, Merge, Split, and Disappearance: Suddenly changes in structure of multiple communities can indicate an important event in the network.

- Temporal Object Stability: Objects which move across different clusters over time are inherently unstable, and should therefore be considered outliers.

- Temporal Object Sociability: Objects which inherently belong to many different clusters are considered sociable. For example, in a bibliographic network, high object sociability 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.

6.4 ONLINE GRAPH OUTLIER DETECTION ALGORITHMS

Given: A stream of graphs.
Find: Outlier graph snapshots.
Examples: Graphs representing inter-disciplinary research papers from bibliographic networks [Aggarwal et al., 2011]; network snapshot with malfunctioning application running on some network node [Idé and Kashima, 2004]; inferring social events and festivals from anomalous activity in mobile telecommunication graphs [Akoglu and Faloutsos, 2010].

The techniques in the previous section are usually applied over a fixed length time series of graphs. Next, we will discuss an eigenvector-based approach and a structural outlier detection approach for graph streams.

6.4.1 SPECTRAL METHODS

[Idé and Kashima, 2004] proposed an eigenvector-based method for anomaly detection over graph streams. The principal eigenvector of the graph weight matrix at time t is called activity vector $u(t)$. Activity vectors over time interval h are stored in the matrix $U(t) = [u(t), u(t-1), \ldots, u(t-h+1)]$. Then the "typical pattern" is defined to be the left singular vector of $U(t)$. For any new data point at time t, this $U(t-1)$ matrix is constructed, the typical pattern is extracted and then the typical pattern vector is compared against the activity vector at time t. The angle between the vectors gives similarity between the activity vector and the typical pattern of the data in the last h time snapshots. The authors provide an online algorithm to calculate the threshold for the angle as well. A similar method is also proposed by [Akoglu and Faloutsos, 2010] to spot anomalous points in time at which many agents in an agent network change their behavior in a way such that it deviates from the norm.

6.4.2 STRUCTURAL OUTLIER DETECTION

While Ide et al. take a spectral approach, [Aggarwal et al., 2011] propose the problem of structural outlier detection in massive network streams. They use a structural connectivity model in order to define outliers in graph streams as those graph objects which contain unusual bridging edges. In order to handle the sparsity problem of massive networks, the network is dynamically partitioned in order to construct statistically robust models of the connectivity behavior. For robustness, multiple such partitionings are maintained. They propose a probabilistic algorithm for maintaining summary structural models about the graph stream. These models are maintained with the use of an innovative reservoir sampling approach for efficient structural compression of the underlying graph stream. Using these models, edge generation probability is defined and then graph object likelihood fit is defined as the geometric mean of the likelihood fits of its constituent edges. Those objects for which this fit is t standard deviations below the average of the likelihood probabilities of all objects received so far are reported as outliers. Further, in [Yu et al., 2013], the authors maintain the information about the changes in the different neighborhoods of the network using a localized principal component analysis algorithm. Localized regions of sudden change in the underlying network are then reported as outliers.

6.5 CONCLUSIONS AND SUMMARY

Many applications can be best modeled as temporal networks. In this chapter, we visited various methods for detecting outliers from temporal graphs. Using a large set of metrics, changes in such graphs can be tracked and an anomaly can be raised when a sudden change in these metrics is observed. Such metrics can indeed be monitored at various levels of a network. We visited mechanisms to detect outliers both from a fixed length series of graph snapshots as well as from graph streams. Finally, we also looked at mechanisms to extract outliers from graphs in the context of the community behavior.

Although community-based outlier detection has been studied for a static heterogeneous graph recently [Gupta et al., 2013d], there is no technique yet for temporal heterogeneous graphs. Also, query-based outlier detection is a recent trend in outlier detection for information networks [Gupta et al., 2013e], [Gupta et al., 2014b], [Gupta et al., 2014c]. However, these studies are for static networks only. Interesting rules (or patterns) for subgraph evolution can be learned, and query based outlier detection can be extended for dynamic scenarios in the future.

CHAPTER 7

Applications of Outlier Detection for Temporal Data

In this chapter, we briefly discuss different environmental, industrial, surveillance, computer network, biological, astronomy, web, information network, and economics applications, which are relevant to temporal outlier detection.

7.1 TEMPORAL OUTLIERS IN ENVIRONMENTAL SENSOR DATA

[Hill and Minsker, 2010] and [Hill et al., 2007] use dynamic Bayesian networks to identify measurement errors in a wind speed data stream from WATERS Network Corpus Christi Bay testbed. Figure 7.1 shows a two-min segment of the data stream from June 15, 2004, in which six suspicious events affecting nine data points can be easily identified. Subsequent investigation of these types of data anomalies revealed that events such as these were most likely caused by wireless transmission errors. Figure 7.2 shows a 35-min segment of the data stream from June 22, 2004, during which a suspicious long duration event occurs. The windspeed between minutes 8 and 26 in the plot appears to have been offset by a constant 7 m/s. This could be because of sensor fault (offset bias fault).

[Angiulli and Fassetti, 2007] extract distance outliers from the rain, sea surface temperature, relative humidity, precipitation time series obtained from the Pacific Marine Environmental Laboratory of the U.S. National Oceanic and Atmospheric Administration (NOAA). [Birant and Kut, 2006] identify spatio-temporal outliers from the wave height values of four seas: the Black Sea, the Marmara Sea, the Aegean Sea, and the east of the Mediterranean Sea. The outliers consist of locations with significantly high wave height values on a particular date compared to its spatio-temporal neighbors.

The region which is circled in dashed lines in Figure 7.3 had significantly high wave height values on January 24, 1998. The approximate wave height value in this region is 6 m. So it contains S-Outliers. Figure 7.4 shows the wave height values of the same region in different years. We detected that the region circled in dashed lines in Figure 7.3 had extreme wave height values in 1998. But, in other years, wave height values of this region were not too high. For this reason, the objects in this region are confirmed as ST-Outliers.

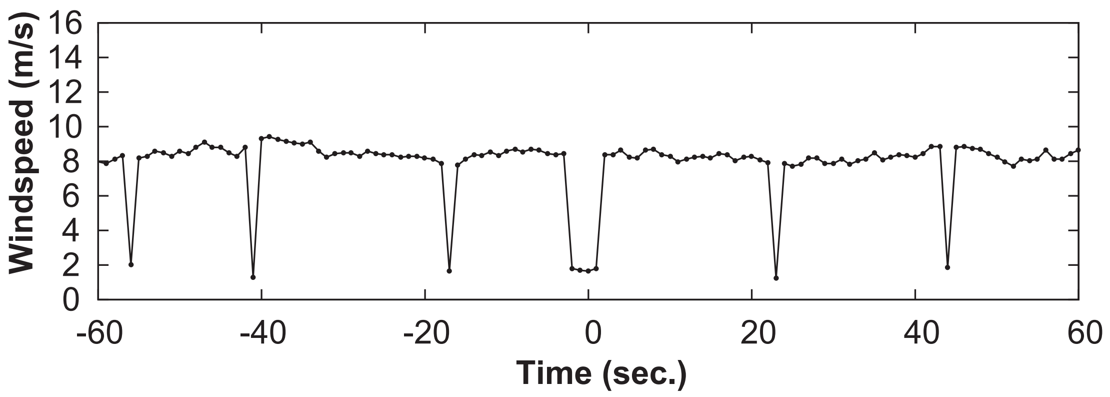

Figure 7.1: Data exhibiting errors resulting from short duration faults. Based on [Hill and Minsker, 2010].

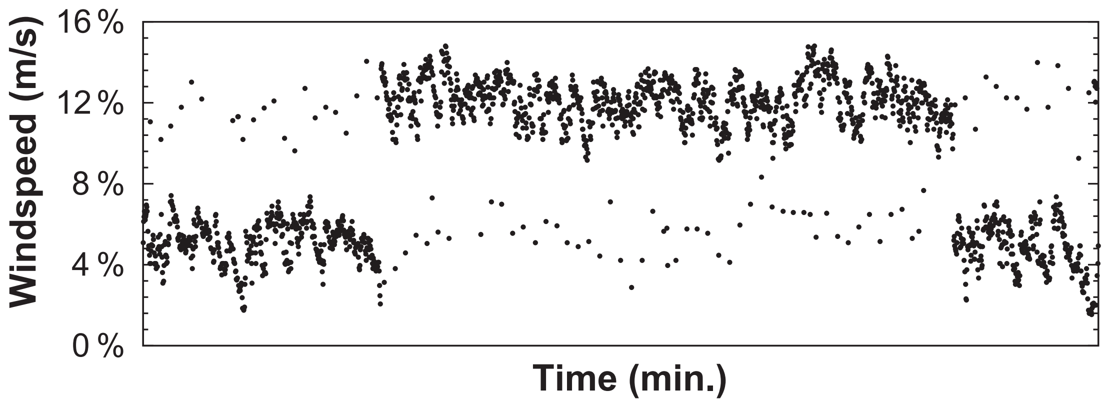

Figure 7.2: Data exhibiting errors resulting from long duration faults. Based on [Hill and Minsker, 2010].

[Cheng and Li, 2006] compute spatio-temporal outliers from the water height data obtained from Ameland, a barrier island in the north of the Netherlands. Figure 7.5 shows ST-Outliers marked on the digital elevation model for 6 consecutive years of the water height data.

[Sun et al., 2005] explore the South China area dataset from 1992–2002 with five variables: cloud cover percentage, diurnal temperature range, precipitation, temperature, vapor, and pressure. They answer outlier queries like "which locations always have different temperature from their neighborhoods in the recent 10 years." They can find droughts and flood outliers like flood in Yangzi River Valley in 1998. [Wu et al., 2010] find extreme precipitation events from South American precipitation data set obtained from the NOAA (Hydrology) for Peru over 1998–2002.

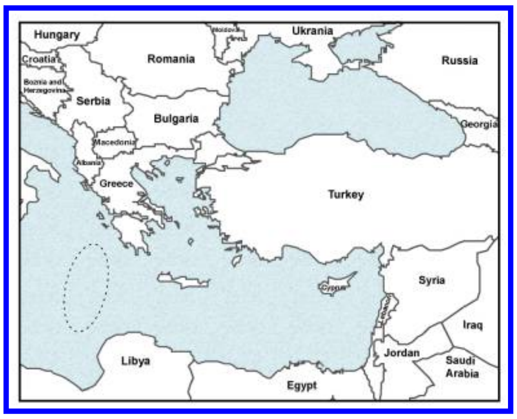

Figure 7.3: The region circled in dashed lines contains S-outliers (January 24, 1998). Based on [Birant and Kut, 2006].

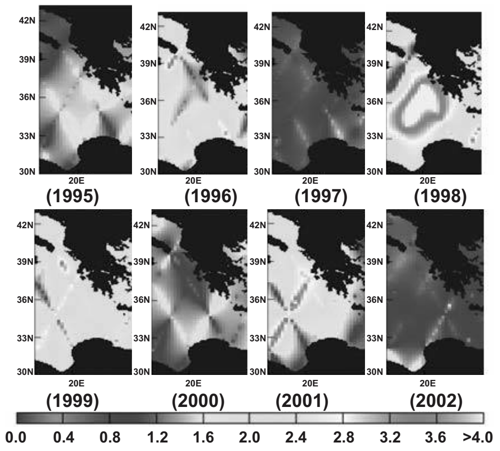

Figure 7.4: The wave height values of the same region on the same day, but in different years. Based on [Birant and Kut, 2006].

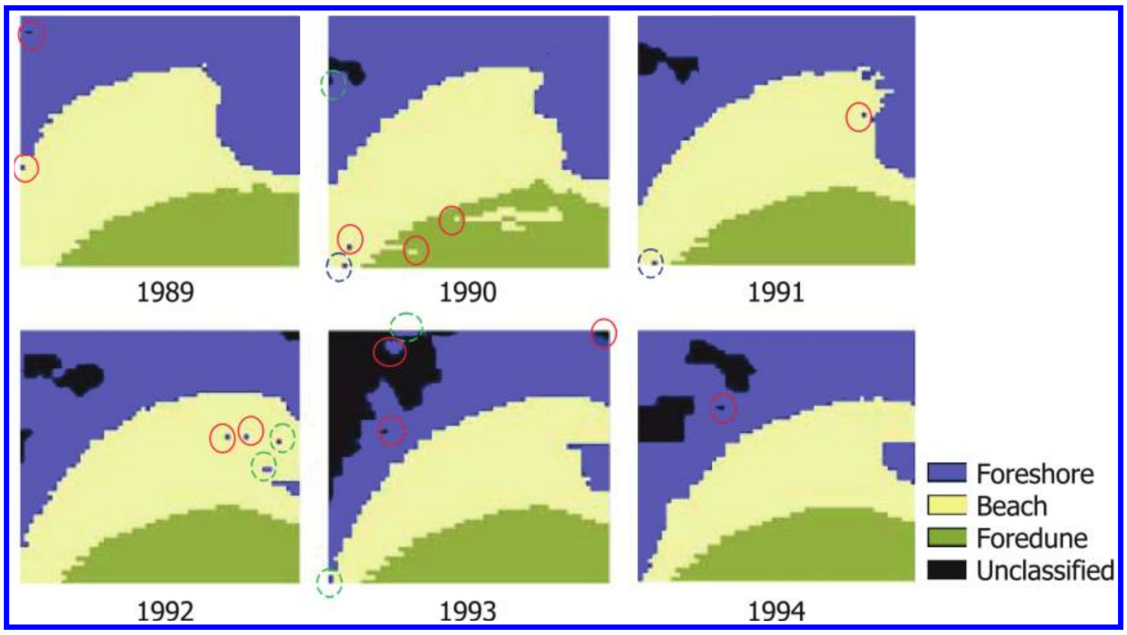

Figure 7.5: ST-outliers from digital elevation model data (red circles) [Cheng and Li, 2006].

Outlier detection using rotated PCA has also been used to discover drought areas, intense fertility loss areas, hurricanes from Australian district rainfall [Drosdowsky, 1993], Sicily Island yearly Maximum Value Composit of SPOT/VEGETATION NDVI [Lasaponara, 2006] and NOAA/NCEP (National Centers of Environmental Prediction) global reanalysis [Lu and Liang, 2004] data sets. Figure 7.6 shows areas that were affected by large fires on the Sicily Island. Figure 7.7 shows the detected boundary of Hurricane Isabel using Wavelet power distribution of Global water vapor.

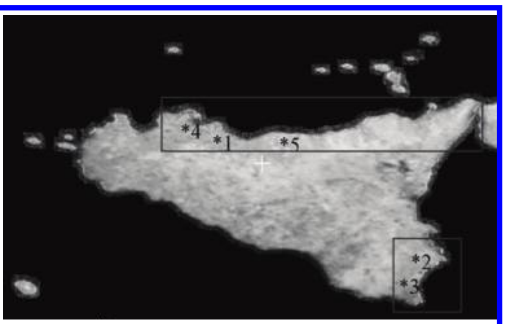

Figure 7.6: The areas indicated in the black boxes were affected by large fires that occurred during the 1998 and 1999 fire seasons (Sicily Island) [Lasaponara, 2006].

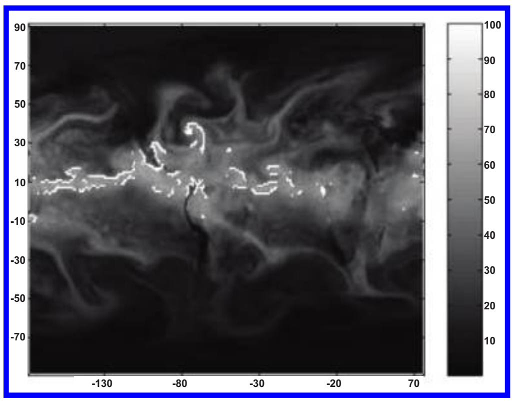

Figure 7.7: Detected boundary of hurricane Isabel using wavelet power distribution of global water vapor at 18 PM, Sep. 18, 2003 [Lu and Liang, 2004].

7.2 TEMPORAL OUTLIERS IN INDUSTRIAL SENSOR DATA

The work in [Basu and Meckesheimer, 2007] predicts anomalies based on two different signals obtained from a flight data recorder (FDR). These correspond to the altitude of the aircraft and roll angle. These have been used to find anomalous altitude changes and roll angle using deviation from median-based prediction. Anomalies in jet engines have been discovered by analyzing the high/low pressure shaft harmonic frequencies in jet engine vibration data [Nairac et al., 1999] using k-Means clustering. Similarly, anomalies such as tool breakage detection have been discovered using cutting force data [Dasgupta and Forrest, 1996] with a neural network. [Bu et al., 2007] discover discords from power consumption data for a Dutch research facility using Haar wavelets and augmented tries.

In the simulation shown in Figure 7.8, the tool was in normal cutting operation for 1500 time steps and then one tooth was broken, causing changes in cutting force signals at the corresponding tooth periods. The first 1000 data points were used as the self set for generating detectors, and rest of the data series was used for testing. Figure 7.9 shows a typical run and number of activated detectors. Note that three detectors are activated when the anomaly is present and no detectors are activated during normal operation.

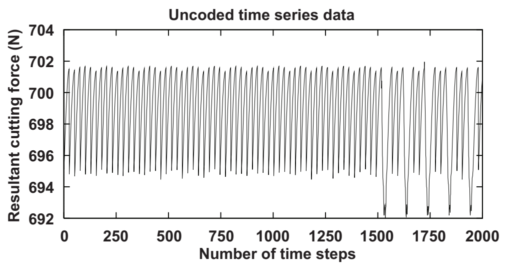

Figure 7.8: Simulated cutting force signals of normal behavior and with tool breakage in a milling operation. One tooth of the cutter is broken after 1500 time steps. Based on [Dasgupta and Forrest, 1996].

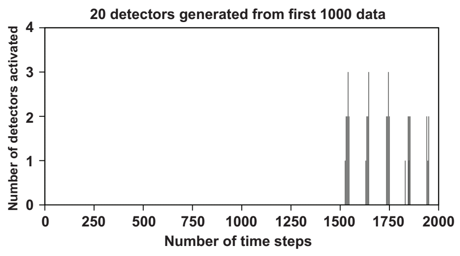

Figure 7.9: Number of detectors activated when novel patterns are found. Based on [Dasgupta and Forrest, 1996].

7.3 TEMPORAL OUTLIERS IN SURVEILLANCE AND TRAJECTORY DATA

[Li et al., 2009] discover anomalies from average daily speed and average daily load dataset for taxicabs in San Francisco during July 2006. [Ge et al., 2010] find anomalies such as vehicle moving to the left side of the road (wrong way), vehicle taking a wrong short cut, and people jaywalking. [Yankov et al., 2008] also find discords from trajectory data. Trajectory data can also be used for monitoring the movement of patients and senior citizens. For example, it can be used to discover events such as taking a wrong bus, having a bad fall, encountering a sudden slow-down and getting lost [Bu et al., 2009]. They use dynamic cluster maintenance to discover such outlier activities. In [Liu et al., 2012], the authors proposed a stochastic modeling approach to identify anomaly events of medical devices from their location traces. The application scenario is in a hospital environment, where each medical device has been attached by a sensor and location traces can be collected. Hotspots are first detected using a density-based clustering approach. The identified hotspots are then used as the context of nearby trajectories. The proposed stochastic model captures the movements of medical devices as the transitions in finite state machines. The transition patterns can be estimated from the historical indoor location traces, and abnormal patterns will be detected by comparing with the majority location traces.

Surveillance data can be useful in smart homes to discover anomaly situations. An example would be the situation where the resident has turned on the bathwater, but has not turned it off before going to bed [Jakkula and Cook, 2008]. Similarly, data from health and medical monitoring sensors attached to patients can be used to identify symptoms or diseases. Surveillance videos can be explored to discover outliers, such as the appearance of a new object, object zooming in a camera and novel video content [Pokrajac et al., 2007] using the distance-based local outliers for data streams. [Pokrajac et al., 2007] study trajectories from surveillance videos to identify

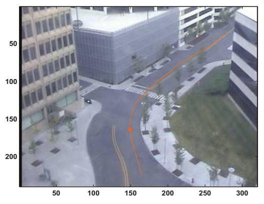
Figure 7.10: Vehicle moves to left side of road (wrong way) [Ge et al., 2010].

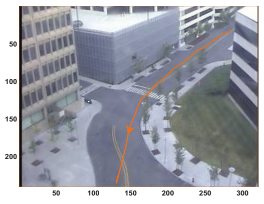
Figure 7.11: Vehicle takes short cut to drive to wrong way [Ge et al., 2010].

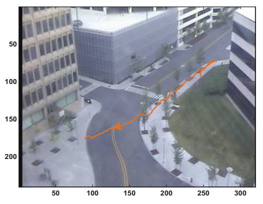
Figure 7.12: Person crossing the road illegally [Ge et al., 2010].

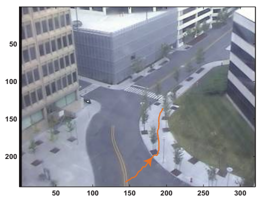
Figure 7.13: Person crossing the road illegally [Ge et al., 2010].

anomalous behaviors such as people first walking right and then back left, or people walking very slowly.

7.4 TEMPORAL OUTLIERS IN COMPUTER NETWORKS DATA

Techniques for outlier detection from temporal data have been widely used for intrusion detection [Sequeira and Zaki, 2002], [Hofmeyr et al., 1998], [Lane and Brodley, 1997], [Lane and Brodley, 1998], [Angiulli and Fassetti, 2007], [Warrender et al., 1999]. [Lakhina et al., 2004a] use multivariate timeseries of origin-destination flows measuring the number of bytes, packets, and IP-level flows to discover anomalies such as high rate point-to-point byte transfer, denial of service (DOS), distributed denial of service (DDOS) attacks, flash crowd (large demand for a resource/service), host scanning for a vulnerable port or network scanning for a target port, WORM[1], OUTAGE[2], etc. They find outliers using the subspace method which is borrowed from the multivariate statistical process control literature. Domain-specific features, such as duration, protocol type, number of bytes transferred for TCP connection with Euclidean distance are used with a variant of single-linkage clustering in [Portnoy et al., 2001]. [Ye, 2000] uses Markov models on the audit data of a Sun Solaris system from MIT Lincoln lab to detect intrusion scenarios, such as password guessing, root privileges access through symbolic links, remote unauthorized access attempt, etc. Sequences of Unix user commands are examined in [Szymanski and Zhang, 2004] using SVMs to discriminate between normal sequences and intrusion sequences. A variant of the online sequential discounting algorithm has been used in [Guo et al., 2006] to characterize probabilistic correlation between flow-intensities measured at multiple points in a computer network using GMMs. An AutoRegressive model with eXogenous inputs (ARX) has been proposed in [Jiang et al., 2006] to learn the relationship between flow intensities measured at multiple points in a computer network, and thereby detect correlation-based faults.

[1]A self propagating code that spreads across a network by exploiting security flaws
[2]Events causing decrease in traffic exchanged between a pair of nodes

7.5 TEMPORAL OUTLIERS IN BIOLOGICAL DATA

[Keogh et al., 2005] discover anomalous subsequences (discords) from electrocardiograms as shown in Figures 7.14 and 7.15. Shape discords (unusual shapes) have been found to be useful in zoology, anthropology, and medicine [Wei et al., 2006]. [Wei et al., 2006] use shape discords to discover an odd butterfly, given a group of seemingly similar butterflies as shown in Figures 7.16 and 7.17. Given a dataset of red blood cells, they use their techniques to discover teardrop shaped cells, or dacrocytes, which is indicative of several blood disorders. A fungus dataset has been studied to discover spores that have sprouted an "appendage" known as a germ tube. Endangered species equipped with sensors have been studied to discover abnormal migration behaviors of the species.

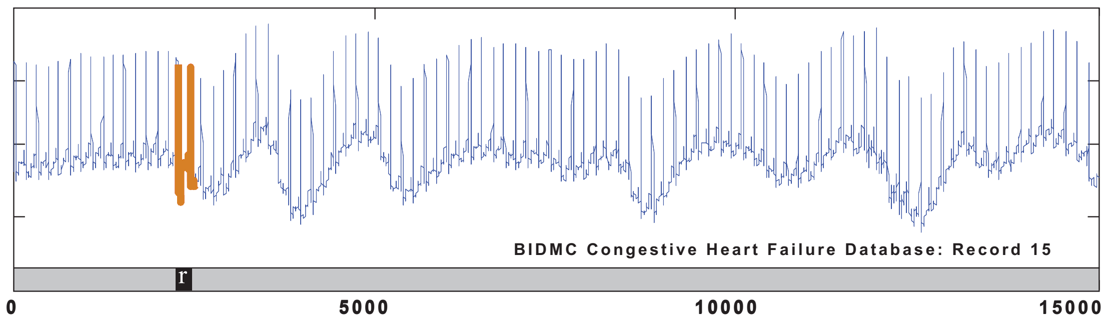

Figure 7.14: An ECG that has been annotated by a cardiologist as containing one premature ventricular contraction [Keogh et al., 2005].

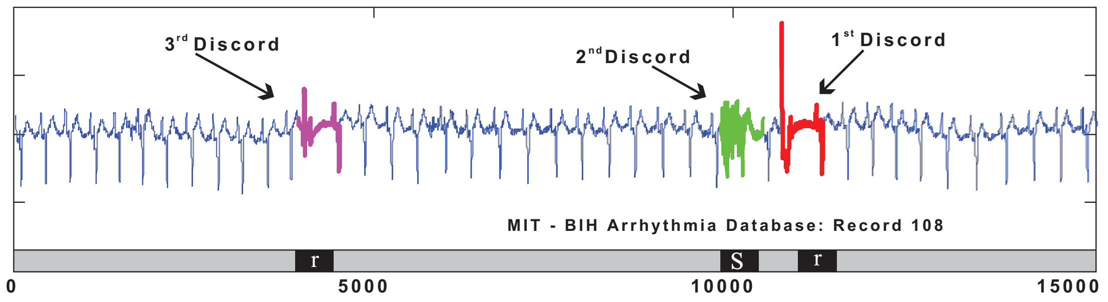

Figure 7.15: An excerpt of an ECG that has been annotated by a cardiologist as containing 3 anomalies [Keogh et al., 2005].

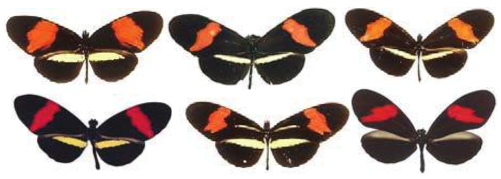

Figure 7.16: Six examples of butterflies, ostensibly all Heliconius erato (red passion flower) butterflies [Wei et al., 2006].

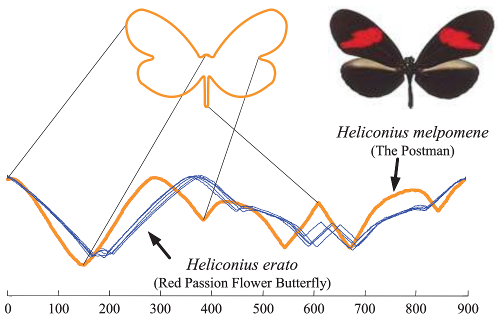

Figure 7.17: An example of Heliconius melpomene and not Heliconius erato [Wei et al., 2006].

7.6 TEMPORAL OUTLIERS IN ASTRONOMY DATA

[Keogh et al., 2005] discover discords from Space telemetry data. These discords correspond to manually noted errors, such as "Poppet pulled significantly out of the solenoid before energizing". [Yankov et al., 2008] find discords from star light-curve data. [Zhu and Shasha, 2003] detect high gamma ray bursts from Milagro Gamma Ray data stream.

7.7 TEMPORAL OUTLIERS IN WEB DATA

Given multiple crawls of the web graph, [Papadimitriou et al., 2010], [Papadimitriou et al., 2008] find a crawl graph with anomalies. These anomalies refer either to failures of the web hosts that do not allow the crawler to access their content or hardware/software problems in the search engine infrastructure that can corrupt parts of the crawled data. They use graph similarity measures like vertex/edge overlap, vertex ranking, vertex/edge vector similarity, sequence similarity and signature similarity to derive respective time series and then detect outliers. Figure 7.18 shows such anomalies for a tiny web graph. [Yankov et al., 2008] study the MSN query logs to discover both anticipated bursts (e.g., "Easter", "Hanukkah") and unanticipated bursts (e.g., unexpected events like death of a celebrity) of web queries as discords. [Lappas et al., 2012] use the notion of spatiotemporal term burstiness to compute highly spatiotemporally impactful events as outliers. [Mathioudakis et al., 2010] also propose efficient dynamic programming algorithms for the similar task of identifying notable geographically focused events from user generated content.

7.8 TEMPORAL OUTLIERS IN INFORMATION NETWORK DATA

[Aggarwal et al., 2011] discover graphs representing inter-disciplinary research papers as outliers from the DBLP dataset using structural outlier detection in massive network streams. They also

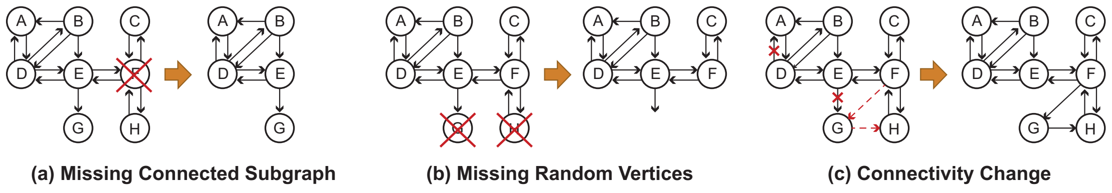

Figure 7.18: Anomalies in a tiny web graph.

discover movies with a cast from multiple countries as outliers from the IMDB dataset. [Gupta et al., 2012b] discover those authors from DBLP co-authorship network as outliers which show a significant deviation in the changes of their research areas compared with other authors. This work explores the difference across two network snapshots using temporal community outlier detection methods. Similarly, they discover actors from IMDB as outliers which show a very unusual change in their movie genre distributions. [Priebe et al., 2005] study the communication graph of the Enron data with respect to the maximum degree and digraph size scan statistics. Excessive communication outliers which can suggest insider trading scenarios are discovered.

7.9 TEMPORAL OUTLIERS IN ECONOMICS TIME SERIES DATA

Various temporal economic datasets have been studied with respect to outlier detection. [Gupta et al., 2012a] identify country outliers based on the unusual change in the constituents of the GDP (Consumption, Investment, Public Expenditure and Net Exports) across time, using temporal community outlier detection methods. They also find US states as outliers from the Budget dataset with respect to anomalous change in the distribution of spending with respect to different components. The *Budget* dataset consists of (Pensions, Health Care, Education, Defense, Welfare, Protection, Transportation, General Government, Other Spending) components for 50 U.S. states for 2001–2010 (10 snapshots)[3]. For states with 6% pension, 16% healthcare, 32% education, 16% other spending in 2006, it has been observed that healthcare increased by 4–5% in 2009–2010 while other spending decreased by 4%. However, in the case of **Arkansas (AK)** which followed a similar distribution in 2006, healthcare decreased by 3% and other spending increased by 5% in 2009–2010. More details can be found in Figure 7.19. The right figure shows the distribution of expenses for AK for the 10 years, while the left part shows similar distribution averaged over 5 states which have a distribution very similar to AK for the years 2004–2009. One can see that Arkansas follows quite a different distribution compared to the five states for other years especially for 2002.

[Otey et al., 2006] study the U.S. Census Bureau's Income data set to detect outliers, on the basis of unusual combinations of demographic attribute values. They also studied the U.S.

[3]`http://www.usgovernmentspending.com/`

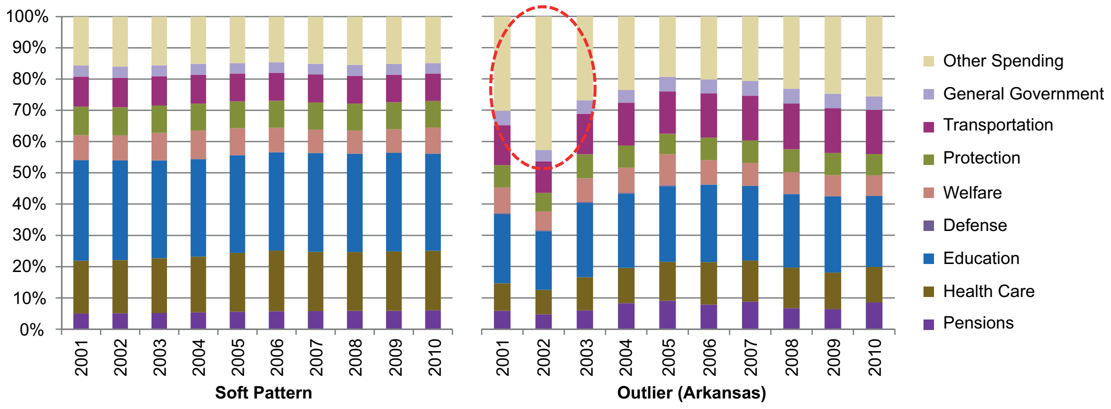

Figure 7.19: Average trend of 5 states with distributions close to that of AK for 2004-2009 (left-pattern), distributions of budget spending for AK (right-outlier) [Gupta et al., 2012a].

Congressional Voting Data to find outliers. An example is a Republican congressman who voted significantly differently from his party on four bills. They use distance-based outlier detection for distributed temporal data to obtain such outliers. [Zhu and Shasha, 2003] perform outlier subsequence detection from the NYSE IBM stock time series data using Shifted Wavelet Trees.

7.10 CONCLUSIONS AND SUMMARY

Outlier detection in general is a very applied field. In this chapter, we discussed a wide variety of applications of outlier detection for temporal data. Temporal data is available in almost all domains where computing is an essential component. Using various examples from published research we discussed applications of outlier detection for environmental, industrial, surveillance, computer network, biological, astronomy, web, information network, and economics datasets. Recently, new forms of temporal data are becoming available and in huge quantities. New kind of datasets open up new challenges and demand design of new outlier detection techniques.

CHAPTER 8
Conclusions and Research Directions

In this book, we presented an organized overview of the various techniques proposed for outlier detection on temporal data. Modeling temporal data is a challenging task due to the dynamic nature and complex evolutionary patterns in the data. A wide variety of models have been developed to capture different facets in temporal data outlier detection. This book organized the discussion along different data types, presented various outlier definitions, and briefly introduced the corresponding techniques. Finally, we also discussed various applications for which these techniques have been successfully used. This book provides a number of insights and lessons as follows.

- The methods for different data types are not easy to generalize to one another, though some of them may have similarity in the framework at the broader level. For example, change detection in continuous time series and discrete time series both require forecasting methods. However, the specific kind of forecasting method is extremely different in the two scenarios (regression models in one vs. Markov Models in the other).
- Most window-based models are currently offline, whereas online methods do exist for point-based models. Therefore, there is significant opportunity for research in the former.
- While the number of formulations of the temporal outlier detection problem are diverse, they are generally motivated by the most common applications which are encountered in the literature. Many recent applications, especially those corresponding to novel data types in the context of web-based applications, may result in combinations of facets, which have not been explored before.
- There are numerous formulations of temporal outlier detection, which have not been sufficiently explored. This is because of the many different combinations of facets, which can be used for defining temporal outlier detection problems. Complex data types such as social streams in which two different data types are present in combination (text and structure) have also not been studied.

Here are a few interesting research directions that can be pursued for outlier detection in temporal data.

- **Further Explorations in Community-Based Outlier Detection**: The problem of community-based outlier detection can be further studied with respect to the following dimensions. (1) In previous studies, integration of community matching and outlier detection has been studied. How to integrate the three tasks of community detection, community matching and outlier detection together? (2) Community-based outliers are sensitive to the criteria used for clustering. How to incorporate user constraints to assist clustering and then use such guidance for community matching and outlier detection?

- **Further Explorations in Query-Based Outlier Detection**: Until now, outlier detection has been studied from a global perspective. Instead of taking a general global perspective, one may want to build an outlier detection system that aims at giving the user a flexibility to find outliers following a particular schema and predicates encoded in the form of a query. For example, given a temporal network, one may want to find size 3 cliques containing a data mining author and two theory authors, which are outliers given a temporal DBLP network of computer science research publications, authors, conferences and title keywords. Thus, rather than finding all outlier subgraphs from a temporal series of graphs, the user wants to restrict the outliers to follow the 3-clique schema. How to support such outlier detection queries in an efficient way is an interesting area of research. For different datasets, queries could be of different forms. For networks, a query could simply be a typed subgraph query. For categorical sequence, a query could simply be a motif.

- **Context-Sensitive Outlier Detection**: We discussed a variety of methods which work for particular types of data. However, often times multiple signals from different sources could be integrated to discover more robust outliers. For example, to find faulty nodes in a MapReduce system, one can leverage both task tracker logs as well as the computer operating system performance logs. Sudden increase in CPU usage may look like an anomaly if only performance logs are leveraged. But if a new mapper job started on the machine at the same time (as obtained from task tracker logs), there is enough context to claim that this sudden increase in CPU usage is not an anomaly.

- **Better Understanding and Description of Anomalies**: Current outlier detection systems have a very good precision and recall for outlier events which is very helpful for addressing the alarms. However, most of the times, the exact cause for the alarm is unclear. More research work needs to be done to identify the root cause of the alarms, and to be able to distinguish outliers into error, noise or interesting data points. Also, given that an outlier corresponds to an interesting event, one may want to further classify popular interesting events for a domain into a few domain-specific categories. For example, for network intrusion datasets, an outlier could be classified as a distributed DOS attack, or some virus attack, etc. Essentially outlier detection systems may also have capabilities of effective outlier description and anomaly type prediction.

- **Outlier Detection under the Influence of External Events**: Until now, there have studies on datasets by observing the temporal changes in the dataset under normal conditions.

However, say the system under observation is exposed to an external event. It is expected that different parts of the system will react to the external event in a particular way. Most of the parts of the system follow a few fixed reaction patterns. However, some parts do not. Such parts of the system can be considered as outliers for the system under influence of the external event. For example, a social network reacts under the influence of various events like elections, earthquakes, volcanoes, arrests, movements, etc. Usually, the nodes and subgraphs react to such events in a usual way which corresponds to certain patterns. The nodes near the event location get affected the most. The influence then spreads from such nodes to other nodes in the network with a damping effect. But certain nodes may not react as expected. Besides the nodes, there could be interesting links or subgraphs in the network which respond to events in a way much different than expected, e.g., a faulty sensor may not show any impact even when there is a cyclone in the neighborhood.

- **Outlier Detection for Multiple Networks with Shared Objects**: Many users have their profile data scattered across multiple social networks. It could be interesting to observe the behavior of users (e.g., temporal community changes) and discover surprising outliers from such networks. Across multiple networks, one expects connectivity structure and community structures to be somewhat similar and to follow similar change trends. However, users who are spamming one network, may have very different structures across networks. A special case of the multiple networks are Location-Based Social Networks (LBSNs) and Event-Based Social Networks (EBSNs) which essentially consist of two parts—an online and an offline network. It could be very interesting to identify patterns of user connectivity and reactions to various events, across these two networks and identify ways in which users or communities deviate from such patterns.

- **Fake Account Detection from Social Media**: There exist a large number of fake accounts on various social media websites like Facebook and Twitter. How can we analyze the temporal behavior of various user accounts on such websites and identify features based on which we could identify fake accounts with a high precision and recall?

- **Impact of Outliers**: Current research outlier detection has focused only on detection of outliers. Usually outliers with a high anomaly score are ranked higher. However, outliers can also be ranked based on expected damage that they can cause. Damage or risk prediction models can be combined with outlier detection systems to rank outliers. Also, damage could be long term or short term. Is it possible to estimate amount of damage and duration up to which the loss will be incurred if the outlier is not removed from the data?

- **Semi-Supervised Outlier Detection**: Most of the outlier detection systems are unsupervised. But for temporal data analysis, one can use the system to find outliers, which can then be validated manually. Can these validated outliers be used to help the system detect outliers better in the future? There is currently not much research on outlier detection for temporal data where past outliers are used to improve the accuracy of the system in the future.

- **Outlier Ensemble Analysis:** Time-series data is particularly challenging task because of the significant amount of noise in the underlying data representation. Different models may have different effectiveness over different data sets. Ensemble analysis provides a way to achieve these goals. Recently, a position paper on outlier ensemble analysis [Aggarwal, 2012] discusses many of the challenges in this important area.

This book provides a good insight into various techniques that have been applied on multiple forms of temporal data and can be used to identify problem settings that can be worked on in the future. Finally, for further reading, we direct the reader to a recent book on outlier analysis [Aggarwal, 2013] and to [Gupta et al., 2013b], [Gupta et al., 2013a], and [Gupta et al., 2013c] for tutorial version (slides) of the content covered in this book. Also, [Chandola et al., 2012] present a nice overview of outlier detection techniques for time series data. We did not discuss much about event detection in text data. [Markou and Singh, 2003a], and [Markou and Singh, 2003b] discuss such novelty detection techniques in detail.

Bibliography

Adam, N. R., Janeja, V. P., and Atluri, V. (2004). Neighborhood Based Detection of Anomalies in High Dimensional Spatio-temporal Sensor Datasets. In *Proc. of the 2004 ACM Symposium on Applied Computing (SAC)*, pages 576–583. DOI: 10.1145/967900.968020. 52

Aggarwal, C. and Subbian, K. (2012). Event Detection in Social Streams. In *Proc. of the* 12^{th} *SIAM Intl. Conf. on Data Mining (SDM)*, pages 624–635. 1, 26

Aggarwal, C. C. (2005a). On Abnormality Detection in Spuriously Populated Data Streams. In *Proc. of the 2005 SIAM Intl. Conf. on Data Mining (SDM)*, pages 80–91. DOI: 10.1137/1.9781611972757.8. 1, 24

Aggarwal, C. C. (2005b). On Change Diagnosis in Evolving Data Streams. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 17(5):587–600. DOI: 10.1109/TKDE.2005.78. 33, 35

Aggarwal, C. C. (2007). *Data Streams: Models and Algorithms*, volume 31. Springer. 21

Aggarwal, C. C. (2012). Outlier Ensembles: Position Paper. *ACM SIGKDD Explorations*, 14(2):49–58. DOI: 10.1145/2481244.2481252. 90

Aggarwal, C. C. (2013). *Outlier Analysis*. Springer. DOI: 10.1007/978-1-4614-6396-2. 1, 19, 32, 90

Aggarwal, C. C. (2014). A Survey of Stream Classification Algorithms. In *Data Classification: Algorithms and Applications*. Chapman and Hall/CRC. 36

Aggarwal, C. C. and Yu, P. S. (2001). Outlier Detection for High Dimensional Data. *SIGMOD Records*, 30:37–46. DOI: 10.1145/376284.375668. 1, 32, 33

Aggarwal, C. C. and Yu, P. S. (2008). Outlier Detection with Uncertain Data. In *Proc. of the 2008 SIAM Intl. Conf. on Data Mining (SDM)*, pages 483–493. DOI: 10.1137/1.9781611972788.57. 1

Aggarwal, C. C. and Yu, P. S. (2010). On Clustering Massive Text and Categorical Data Streams. *Knowledge and Information Systems (KAIS)*, 24(2):171–196. DOI: 10.1007/s10115-009-0241-z. 24, 25

Aggarwal, C. C., Zhao, Y., and Yu, P. S. (2011). Outlier Detection in Graph Streams. In *Proc. of the* 27th *Intl. Conf. on Data Engineering (ICDE)*, pages 399–409. IEEE Computer Society. DOI: 10.1109/ICDE.2011.5767885. 1, 26, 74, 75, 84

Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. (1993). *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall. 70

Akoglu, L. and Faloutsos, C. (2010). Event Detection in Time Series of Mobile Communication Graphs. In *Proc. of the Army Science Conf.* 74, 75

Akoglu, L., McGlohon, M., and Faloutsos, C. (2010). Oddball: Spotting Anomalies in Weighted Graphs. In *Proc. of the* 14th *Pacific-Asia Conf. on Advances in Knowledge Discovery and Data Mining (PAKDD)*, pages 410–421. DOI: 10.1007/978-3-642-13672-6_40. 68, 70

Al-Khateeb, T., Masud, M. M., Khan, L., Aggarwal, C., Han, J., and Thuraisingham, B. (2012). Stream Classification with Recurring and Novel Class Detection Using Class-Based Ensemble. In *Proc. of the 2012 IEEE* 12th *Intl. Conf. on Data Mining (ICDM)*, pages 31–40. DOI: 10.1109/ICDM.2012.125. 36

Angiulli, F. and Fassetti, F. (2007). Detecting Distance-based Outliers in Streams of Data. In *Proc. of the* 16th *ACM Conf. on Information and Knowledge Management (CIKM)*, pages 811–820. DOI: 10.1145/1321440.1321552. 29, 30, 77, 82

Atallah, M., Gwadera, R., and Szpankowski, W. (2004). Detection of Significant Sets of Episodes in Event Sequences. In *Proc. of the* 4th *IEEE Intl. Conf. on Data Mining (ICDM)*, pages 3–10. DOI: 10.1109/ICDM.2004.10090. 16

Barnett, V. and Lewis, T. (1978). *Outliers in Statistical Data*. John Wiley & Sons Ltd. 7

Basu, S. and Meckesheimer, M. (2007). Automatic Outlier Detection for Time Series: An Application to Sensor Data. *Knowledge and Information Systems - Special Issue on Mining Low-Quality Data*, 11(2):137–154. DOI: 10.1007/s10115-006-0026-6. 17, 80

Bettencourt, L. M., Hagberg, A. A., and Larkey, L. B. (2007). Separating the Wheat from the Chaff: Practical Anomaly Detection Schemes in Ecological Applications of Distributed Sensor Networks. In *Distributed Computing in Sensor Systems (DCOSS)*, pages 223–239. DOI: 10.1007/978-3-540-73090-3_15. 41

Bianco, A. M., García Ben, M., Martínez, E. J., and Yohai, V. J. (2001). Outlier Detection in Regression Models with ARIMA Errors Using Robust Estimates. *Journal of Forecasting*, 20(8):565–579. DOI: 10.1002/for.768. 18

Birant, D. and Kut, A. (2006). Spatio-Temporal Outlier Detection in Large Databases. *Journal of Computing and Information Technology (CIT)*, 14(4):291–297. DOI: 10.1109/ITI.2006.1708474. 49, 50, 77, 79

Branch, J., Szymanski, B., Giannella, C., Wolff, R., and Kargupta, H. (2006). In-Network Outlier Detection in Wireless Sensor Networks. In *Proc. of the 26th IEEE Intl. Conf. on Distributed Computing Systems (ICDCS)*, pages 51–81. DOI: 10.1007/s10115-011-0474-5. 42

Breunig, M. M., Kriegel, H.-P., Ng, R. T., and Sander, J. (2000). LOF: Identifying Density-based Local Outliers. In *Proc. of the 2000 ACM SIGMOD Intl. Conf. on Management of Data (SIGMOD)*, pages 93–104. DOI: 10.1145/342009.335388. 31

Bu, Y., Chen, L., Fu, A. W.-C., and Liu, D. (2009). Efficient Anomaly Monitoring over Moving Object Trajectory Streams. In *Proc. of the 15th ACM Intl. Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 159–168. DOI: 10.1145/1557019.1557043. 31, 81

Bu, Y., Leung, O. T.-W., Fu, A. W.-C., Keogh, E. J., Pei, J., and Meshkin, S. (2007). WAT: Finding Top-K Discords in Time Series Database. In *Proc. of the 7th SIAM Intl. Conf. on Data Mining (SDM)*, pages 449–454. DOI: 10.1137/1.9781611972771.43. 19, 80

Budalakoti, S., Srivastava, A., Akella, R., and Turkov, E. (2006). Anomaly Detection in Large Sets of High-dimensional Symbol Sequences. *NASA Ames Research Center, Tech. Rep. NASA TM-2006-214553*. 8, 10

Budalakoti, S., Srivastava, A. N., and Otey, M. E. (2009). Anomaly Detection and Diagnosis Algorithms for Discrete Symbol Sequences with Applications to Airline Safety. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications*, 39(1):101–113. DOI: 10.1109/TSMCC.2008.2007248. 8, 10

Burman, J. P. and Otto, M. C. (1988). Census Bureau Research Project: Outliers in Time Series. 1

Cabrera, J. a. B. D., Lewis, L., and Mehra, R. K. (2001). Detection and Classification of Intrusions and Faults using Sequences of System Calls. *SIGMOD Records*, 30(4):25–34. DOI: 10.1145/604264.604269. 14

Cao, H., Zhou, Y., Shou, L., and Chen, G. (2010). Attribute Outlier Detection over Data Streams. In *Proc. of the 15th Intl. Conf. on Database Systems for Advanced Applications - Volume Part II (DASFAA)*, pages 216–230. DOI: 10.1007/978-3-642-12098-5_17. 31

Chandola, V., Banerjee, A., and Kumar, V. (2009). Anomaly Detection: A Survey. *ACM Computing Surveys*, 41(3):15:1–15:58. DOI: 10.1145/1541880.1541882. 1

Chandola, V., Banerjee, A., and Kumar, V. (2012). Anomaly Detection for Discrete Sequences: A Survey. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 24(5):823–839. DOI: 10.1109/TKDE.2010.235. 90

Chandola, V., Mithal, V., and Kumar, V. (2008). A Comparative Evaluation of Anomaly Detection Techniques for Sequence Data. In *Proc. of the 2008 8th IEEE Intl. Conf. on Data Mining (ICDM)*, pages 743–748. DOI: 10.1109/ICDM.2008.151. 8, 12, 13

Chang, I., Tiao, G. C., and Chen, C. (1988). Estimation of Time Series Parameters in the Presence of Outliers. *Technometrics*, 30(2):193–204. DOI: 10.1080/00401706.1988.10488367. 18

Chen, C. and Liu, L.-M. (1993). Joint Estimation of Model Parameters and Outlier Effects in Time Series. *Journal of the American Statistical Association*, 88(421):284–297. DOI: 10.2307/2290724. 18

Chen, X.-y. and Zhan, Y.-y. (2008). Multi-scale Anomaly Detection Algorithm based on Infrequent Pattern of Time Series. *Journal of Computational and Applied Mathematics*, 214(1):227–237. DOI: 10.1016/j.cam.2007.02.027. 19

Cheng, T. and Li, Z. (2004). A Hybrid Approach to Detect Spatial-Temporal Outliers. In *Proc. of the* 12th *Intl. Conf. on Geoinformatics Geospatial Information Research*, pages 173–178. 50

Cheng, T. and Li, Z. (2006). A Multiscale Approach for Spatio-Temporal Outlier Detection. *Transactions in Geographic Information Systems (T. GIS)*, 10(2):253–263. DOI: 10.1111/j.1467-9671.2006.00256.x. 49, 50, 78, 79

Cho, H., jin Kim, Y., Jung, H. J., Lee, S.-W., and Lee, J. W. (2008). OutlierD: An R Package for Outlier Detection using Quantile Regression on Mass Spectrometry Data. *Bioinformatics*, 24(6):882–884. DOI: 10.1093/bioinformatics/btn012. 1

Cressie, N. A. C. (1993). *Statistics for Spatial Data*. Wiley. 43

Das, K., Bhaduri, K., and Votava, P. (2011). Distributed Anomaly Detection using 1-class SVM for Vertically Partitioned Data. *Statistical Analysis and Data Mining*, 4(4):393–406. DOI: 10.1002/sam.10125. 47

Dasgupta, D. and Forrest, S. (1996). Novelty Detection in Time Series Data using Ideas from Immunology. In *Proc. of the* 5th *Intl. Conf. on Intelligent Systems*. 80, 81

Dasgupta, D. and Majumdar, N. (2002). Anomaly Detection in Multidimensional Data using Negative Selection Algorithm. In *Proc. of the 2002 Congress on Evolutionary Computation (CEC)*, volume 2, pages 1039–1044. DOI: 10.1109/CEC.2002.1004386. 15

Dasgupta, D. and Nino, F. (2000). A Comparison of Negative and Positive Selection Algorithms in Novel Pattern Detection. In *Proc. of the 2000 IEEE Intl. Conf. on Systems, Man, and Cybernetics*, volume 1, pages 125–130. DOI: 10.1109/ICSMC.2000.884976. 14, 15

Dasu, T., Krishnan, S., Venkatasubramanian, S., and Yi, K. (2006). An Information-Theoretic Approach to Detecting Changes in Multi-dimensional Data Streams. In *In Proc. of the Symp. on the Interface of Statistics, Computing Science, and Applications*. 36

D'haeseleer, P., Forrest, S., and Helman, P. (1996). An Immunological Approach to Change Detection: Algorithms, Analysis and Implications. In *Proc. of the 1996 IEEE Symposium on Security and Privacy*, pages 110–119. DOI: 10.1109/SECPRI.1996.502674. 15

Dickinson, P., Bunke, H., Dadej, A., and Kraetzl, M. (2002). Median Graphs and Anomalous Change Detection in Communication Networks. In *Proc. of the Intl. Conf. on Information, Decision and Control*, pages 59–64. DOI: 10.1109/IDC.2002.995366. 63

Dickinson, P. and Kraetzl, M. (2003). Novel Approaches in Modelling Dynamics of Networked Surveillance Environment. In *Proc. of the* 6^{th} *Intl. Conf. of Information Fusion*, volume 1, pages 302–309. DOI: 10.1109/ICIF.2003.177461. 65

Drosdowsky, W. (1993). An Analysis of Australian Seasonal Rainfall Anomalies: 1950–1987. *Intl. Journal of Climatology*, 13(1):1–30. DOI: 10.1002/joc.3370130102. 49, 79

Elnahrawy, E. and Nath, B. (2004). Context-aware Sensors. In *Wireless Sensor Networks*, pages 77–93. DOI: 10.1007/978-3-540-24606-0_6. 41

Endler, D. (1998). Intrusion Detection Applying Machine Learning to Solaris Audit Data. In *Proc. of the* 14^{th} *Annual Computer Security Applications Conf. (ACSAC)*, pages 268–279. DOI: 10.1109/CSAC.1998.738647. 14

Eskin, E., Arnold, A., Prerau, M., Portnoy, L., and Stolfo, S. (2002). A Geometric Framework for Unsupervised Anomaly Detection: Detecting Intrusions in Unlabeled Data. In *Proc. of Applications of Data Mining in Computer Security*. DOI: 10.1007/978-1-4615-0953-0_4. 10, 11

Eskin, E., Lee, W., and Stolfo, S. (2001). Modeling System Calls for Intrusion Detection with Dynamic Window Sizes. In *Proc. of the DARPA Information Survivability Conf & Exposition II (DISCEX)*, volume 1, pages 165–175. DOI: 10.1109/DISCEX.2001.932213. 13

Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proc. of the* 2^{nd} *ACM Intl. Conf on Knowledge Discovery and Data Mining (KDD)*, pages 226–231. 50

Evangelista, P., Bonnisone, P., Embrechts, M., and Szymanski, B. (2005). Fuzzy ROC Curves for the 1 Class SVM: Application to Intrusion Detection. In *Proc. of the* 13^{th} *European Symposium on Artificial Neural Networks*, pages 345–350. 10, 11

Florez-Larrahondo, G., Bridges, S. M., and Vaughn, R. (2005). Efficient Modeling of Discrete Events for Anomaly Detection using Hidden Markov Models. In *Proc. of the* 8th *Intl. Conf. on Information Security (ISC)*, pages 506–514. DOI: 10.1007/11556992_38. 13

Forrest, S., Hofmeyr, S. A., Somayaji, A., and Longstaff, T. A. (1996). A Sense of Self for Unix Processes. In *Proc. of the 1996 IEEE Symposium on Security and Privacy*, pages 120–128. DOI: 10.1109/SECPRI.1996.502675. 14, 15

Forrest, S., Perelson, A. S., Allen, L., and Cherukuri, R. (1994). Self-Nonself Discrimination in a Computer. In *Proc. of the 1994 IEEE Symposium on Security and Privacy*, pages 202–212. DOI: 10.1109/RISP.1994.296580. 15

Fox, A. J. (1972). Outliers in Time Series. *Journal of the Royal Statistical Society. Series B (Methodological)*, 34(3):350–363. 1, 7

Franke, C. and Gertz, M. (2008). Detection and Exploration of Outlier Regions in Sensor Data Streams. In *Proc. of the 2008 IEEE Intl. Conf. on Data Mining Workshops (ICDMW)*, pages 375–384. DOI: 10.1109/ICDMW.2008.21. 41

Franke, C. and Gertz, M. (2009). ORDEN: Outlier Region Detection and Exploration in Sensor Networks. In *Proc. of the 2009 ACM SIGMOD Intl. Conf. on Management of data (SIGMOD)*, pages 1075–1078. DOI: 10.1145/1559845.1559985. 41

Fu, A. W.-c., Leung, O. T.-W., Keogh, E., and Lin, J. (2006). Finding Time Series Discords based on Haar Transform. In *Proc. of the* 2nd *Intl. Conf. on Advanced Data Mining and Applications (ADMA)*, pages 31–41. DOI: 10.1007/11811305_3. 19

Galeano, P., Peña, D., and Tsay, R. S. (2006). Outlier Detection in Multivariate Time Series by Projection Pursuit. *Journal of the American Statistical Association*, 101(474):654–669. DOI: 10.1198/016214505000001131. 18

Gao, B., Ma, H.-Y., and Yang, Y.-H. (2002). HMMs (Hidden Markov Models) based on Anomaly Intrusion Detection Method. In *Proc. of the 2002 Intl. Conf. on Machine Learning and Cybernetics*, volume 1, pages 381–385. DOI: 10.1109/ICMLC.2002.1176779. 13, 14

Gao, J., Liang, F., Fan, W., Wang, C., Sun, Y., and Han, J. (2010). On Community Outliers and their Efficient Detection in Information Networks. In *Proc. of the* 16th *ACM Intl. Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 813–822. DOI: 10.1145/1835804.1835907. 1, 72

Gaston, M., Kraetzl, M., and Wallis, W. (2006). Graph Diameter as a Pseudo-metric for Change Detection in Dynamic Networks. *Australasian Journal of Combinatorics*, 35:299–312. 63, 64

Ge, Y., Xiong, H., Zhou, Z.-h., Ozdemir, H., Yu, J., and Lee, K. C. (2010). Top-Eye: Top-K Evolving Trajectory Outlier Detection. In *Proc. of the* 19th *ACM Intl. Conf. on Information and Knowledge Management (CIKM)*, pages 1733–1736. DOI: 10.1145/1871437.1871716. 55, 56, 81, 82

Ghosh, A., Schwartzbard, A., and Schatz, M. (1999a). Learning Program Behavior Profiles for Intrusion Detection. In *Proc. of the* 1st *USENIX Workshop on Intrusion Detection and Network Monitoring*, pages 51–62. 14

Ghosh, A., Schwartzbard, A., Schatz, M., et al. (1999b). Using Program Behavior Profiles for Intrusion Detection. In *Proc. of the SANS Intrusion Detection Workshop*. 14

Ghosh, A. K. and Schwartzbard, A. (1999). A Study in using Neural Networks for Anomaly and Misuse Detection. In *Proc. of the* 8th *Conf. on USENIX Security Symposium (SSYM)*, pages 12–23. 14

Ghoting, A., Otey, M. E., and Parthasarathy, S. (2004). LOADED: Link-Based Outlier and Anomaly Detection in Evolving Data Sets. In *Proc. of the* 4th *IEEE Intl. Conf. on Data Mining (ICDM)*, pages 387–390. DOI: 10.1109/ICDM.2004.10011. 1, 24

González, F. A. and Dasgupta, D. (2003). Anomaly Detection Using Real-Valued Negative Selection. *Genetic Programming and Evolvable Machines*, 4(4):383–403. DOI: 10.1023/A:1026195112518. 10, 11, 14, 15

Gosh, A. K., Wanken, J., and Charron, F. (1998). Detecting Anomalous and Unknown Intrusions Against Programs. In *Proc. of the* 14th *Annual Computer Security Applications Conf. (ACSAC)*, pages 259–267. DOI: 10.1109/CSAC.1998.738646. 14

Guo, Z., Jiang, G., Chen, H., and Yoshihira, K. (2006). Tracking Probabilistic Correlation of Monitoring Data for Fault Detection in Complex Systems. In *Proc. of the Intl. Conf. on Dependable Systems and Networks (ICDSN)*, pages 259–268. DOI: 10.1109/DSN.2006.70. 16, 82

Gupta, M., Aggarwal, C. C., and Han, J. (2011a). Finding Top-K Shortest Path Distance Changes in an Evolutionary Network. In *Proc. of the* 12th *Intl. Conf. on Advances in Spatial and Temporal Databases (SSTD)*, pages 130–148. DOI: 10.1007/978-3-642-22922-0_9. 68, 71

Gupta, M., Aggarwal, C. C., Han, J., and Sun, Y. (2011b). Evolutionary Clustering and Analysis of Bibliographic Networks. In *Proc. of the 2011 Intl. Conf. on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 63–70. DOI: 10.1109/ASONAM.2011.12. 73

Gupta, M., Gao, J., Aggarwal, C., and Han, J. (2014a). Outlier Detection for Temporal Data. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*. 2

Gupta, M., Gao, J., Aggarwal, C. C., and Han, J. (2013a). Tutorial: Outlier Detection for Graph Data. In *Proc. of the 2013 Intl. Conf. on Advances in Social Networks Analysis and Mining (ASONAM)*. 90

Gupta, M., Gao, J., Aggarwal, C. C., and Han, J. (2013b). Tutorial: Outlier Detection for Temporal Data. In *Proc. of the* 13th *SIAM Intl. Conf. on Data Mining (SDM)*. 90

Gupta, M., Gao, J., Aggarwal, C. C., and Han, J. (2013c). Tutorial: Outlier Detection for Temporal Data. In *Proc. of the 2013 ACM Intl. Conf. of Information and Knowledge Management (CIKM)*. 90

Gupta, M., Gao, J., and Han, J. (2013d). Community Distribution Outlier Detection in Heterogeneous Information Networks. In *Proc. of the 2013 European Conf. on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, pages 557–573. DOI: 10.1007/978-3-642-40988-2_36. 72, 76

Gupta, M., Gao, J., Sun, Y., and Han, J. (2012a). Community Trend Outlier Detection using Soft Temporal Pattern Mining. In *Proc. of the 2012 European Conf. on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, pages 692–708. DOI: 10.1007/978-3-642-33486-3_44. 1, 71, 72, 73, 85, 86

Gupta, M., Gao, J., Sun, Y., and Han, J. (2012b). Integrating Community Matching and Outlier Detection for Mining Evolutionary Community Outliers. In *Proc. of the* 18th *ACM Intl. Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 859–867. DOI: 10.1145/2339530.2339667. 1, 71, 72, 73, 85

Gupta, M., Gao, J., Yan, X., Cam, H., and Han, J. (2013e). On Detecting Association-Based Clique Outliers in Heterogeneous Information Networks. In *Proc. of the 2013 Intl. Conf. on Advances in Social Networks Analysis and Mining (ASONAM)*. DOI: 10.1145/2492517.2492526. 76

Gupta, M., Gao, J., Yan, X., Cam, H., and Han, J. (2014b). Top-K Interesting Subgraph Discovery in Information Networks. In *Proc. of the* 30th *Intl. Conf. on Data Engineering (ICDE)*. 76

Gupta, M., Mallya, A., Roy, S., Cho, J. H. D., and Han, J. (2014c). Local Learning for Mining Outlier Subgraphs from Network Datasets. In *Proc. of the* 14th *SIAM Intl. Conf. on Data Mining (SDM)*. 76

Gupta, M., Sharma, A. B., Chen, H., and Jiang, G. (2013f). Context-Aware Time Series Anomaly Detection for Complex Systems. In *Proc. of the SDM Workshop on Data Mining for Service and Maintenance*. 9, 10

Gwadera, R., Atallah, M. J., and Szpankowski, W. (2005a). Markov Models for Identification of Significant Episodes. In *Proc. of the* 5th *SIAM Intl. Conf. on Data Mining (SDM)*, pages 404–414. DOI: 10.1137/1.9781611972757.36. 16

Gwadera, R., Atallah, M. J., and Szpankowski, W. (2005b). Reliable Detection of Episodes in Event Sequences. *Knowledge and Information Systems (KAIS)*, 7(4):415–437. DOI: 10.1109/ICDM.2003.1250904. 16

Harold S. Javitz, A. V. (1994). The NIDES Statistical Component Description and Justification. Technical Report A010, SRI International. 24

Hawkins, D. M. (1980). *Identification of Outliers*. Chapman and Hall. DOI: 10.1007/978-94-015-3994-4. 7

Henderson, K., Eliassi-Rad, T., Faloutsos, C., Akoglu, L., Li, L., Maruhashi, K., Prakash, B. A., and Tong, H. (2010). Metric Forensics: A Multi-Level Approach for Mining Volatile Graphs. In *Proc. of the* 16th *ACM Intl. COnf. on Knowledge Discovery and Data Mining (KDD)*, pages 163–172. DOI: 10.1145/1835804.1835828. 68

Hill, D. J. and Minsker, B. S. (2010). Anomaly Detection in Streaming Environmental Sensor Data: A Data-driven Modeling Approach. *Environmental Modelling and Software*, 25(9):1014–1022. DOI: 10.1016/j.envsoft.2009.08.010. 17, 21, 77, 78

Hill, D. J., Minsker, B. S., and Amir, E. (2007). Real-time Bayesian Anomaly Detection for Environmental Sensor Data. In *Proc. of the* 32nd *Conf. of Intl. Association of Hydraulic Engineering and Research (IAHR)*. DOI: 10.1029/2008WR006956. 17, 21, 27, 77

Hodge, V. J. and Austin, J. (2004). A Survey of Outlier Detection Methodologies. *Artificial Intelligence Review*, 22(2):85–126. DOI: 10.1023/B:AIRE.0000045502.10941.a9. 1

Hofmeyr, S. A., Forrest, S., and Somayaji, A. (1998). Intrusion Detection using Sequences of System Calls. *Journal of Computer Security*, 6(3):151–180. 14, 15, 82

Idé, T. and Kashima, H. (2004). Eigenspace-based Anomaly Detection in Computer Systems. In *Proc. of the* 10th *ACM Intl. Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 440–449. DOI: 10.1145/1014052.1014102. 74, 75

Jagadish, H. V., Koudas, N., and Muthukrishnan, S. (1999). Mining Deviants in a Time Series Database. In *Proc. of the* 25th *Intl. Conf. on Very Large Data Bases (VLDB)*, pages 102–113. DOI: 10.1109/SSDM.2004.1311192. 18

Jakkula, V. and Cook, D. J. (2008). Anomaly Detection using Temporal Data Mining in a Smart Home Environment. *Methods of Information in Medicine*, 47(1):70–75. 81

Jiang, G., Chen, H., and Yoshihira, K. (2006). Modeling and Tracking of Transaction Flow Dynamics for Fault Detection in Complex Systems. *IEEE Transactions on Dependable and Secure Computing*, 3:312–326. DOI: 10.1109/TDSC.2006.52. 16, 82

Jun, M., Jeong, H., and Kuo, C. (2005). Distributed Spatio-temporal Outlier Detection in Sensor Networks. In *Society of Photo-Optical Instrumentation Engineers (SPIE)*, pages 760–763. DOI: 10.1117/12.604764. 43

Justel, A., Pena, D., and Tsay, R. S. (2001). Detection of Outlier Patches in Autoregressive Time Series. *Statistica Sinica*, 11(3):651–674. 17

Kang, D.-K., Fuller, D., and Honavar, V. (2005). Learning Classifiers for Misuse Detection using a Bag of System Calls Representation. In *Proc. of the* 3^{rd} *IEEE Intl. Conf. on Intelligence and Security Informatics (ISI)*, pages 511–516. DOI: 10.1007/11427995_51. 14

Kapsabelis, K. M., Dickinson, P. J., and Dogancay, K. (2007). Investigation of Graph Edit Distance Cost Functions for Detection of Network Anomalies. In *Proc. of the* 13^{th} *Biennial Computational Techniques and Applications Conf. (CTAC)*, volume 48, pages C436–C449. 61, 62

Kaufman, L. and Rousseeuw, P. (1987). Clustering by means of Medoids. In *Proc. of the* 1^{st} *Intl. Conf. on Statistical Data Analysis Based on the* L_1*-Norm and Related Methods*, pages 405–416. 10

Keogh, E., Lin, J., and Fu, A. (2005). HOT SAX: Efficiently Finding the Most Unusual Time Series Subsequence. In *Proc. of the* 5^{th} *IEEE Intl. Conf. on Data Mining (ICDM)*, pages 226–233. DOI: 10.1109/ICDM.2005.79. 18, 19, 83, 84

Keogh, E., Lin, J., Lee, S.-H., and Van Herle, H. (2006). Finding the Most Unusual Time Series Subsequence: Algorithms and Applications. *Knowledge and Information Systems (KAIS)*, 11(1):1–27. DOI: 10.1007/s10115-006-0034-6. 19

Keogh, E., Lonardi, S., and Chiu, B. Y.-c. (2002). Finding Surprising Patterns in a Time Series Database in Linear Time and Space. In *Proc. of the* 8^{th} *ACM Intl. Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 550–556. DOI: 10.1145/775047.775128. 16

Keogh, E., Lonardi, S., and Ratanamahatana, C. A. (2004). Towards Parameter-free Data Mining. In *Proc. of the* 10^{th} *ACM Intl. Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 206–215. DOI: 10.1145/1014052.1014077. 19

Kifer, D., Ben-David, S., and Gehrke, J. (2004). Detecting Change in Data Streams. In *Proc. of the* 30^{th} *Intl. Conf. on Very Large Data Bases - Volume 30*, pages 180–191. 35

Knorr, E. M. and Ng, R. T. (1998). Algorithms for Mining Distance-Based Outliers in Large Datasets. In *Proc. of the 24th Intl. Conf. on Very Large Data Bases (VLDB)*, pages 392–403. Morgan Kaufmann. 29

Lakhina, A., Crovella, M., and Diot, C. (2004a). Characterization of Network-wide Anomalies in Traffic Flows. In *Proc. of the 4th ACM SIGCOMM Conf. on Internet Measurement (IMC)*, pages 201–206. DOI: 10.1145/1028788.1028813. 82

Lakhina, A., Crovella, M., and Diot, C. (2004b). Diagnosing Network-wide Traffic Anomalies. In *ACM SIGCOMM Computer Communication Review*, volume 34, pages 219–230. DOI: 10.1145/1015467.1015492. 16, 17

Lane, T., Brodley, C., et al. (1997). Sequence Matching and Learning in Anomaly Detection for Computer Security. In *AAAI Workshop: AI Approaches to Fraud Detection and Risk Management*, pages 43–49. 8, 15

Lane, T. and Brodley, C. E. (1997). An Application of Machine Learning to Anomaly Detection. In *Proc. of the 20th National Information Systems Security Conf. (NISSC)*, pages 366–380. 15, 82

Lane, T. and Brodley, C. E. (1998). Temporal Sequence Learning and Data Reduction for Anomaly Detection. In *Proc. of the 5th ACM Conf. on Computer and Communications Security (CCS)*, pages 150–158. DOI: 10.1145/322510.322526. 8, 15, 82

Lappas, T., Vieira, M. R., Gunopulos, D., and Tsotras, V. J. (2012). On The Spatiotemporal Burstiness of Terms. *Proc. of the Very Large Databases (PVLDB)*, 5(9):836–847. 84

Lasaponara, R. (2006). On the Use of Principal Component Analysis (PCA) for Evaluating Interannual Vegetation Anomalies from SPOT/VEGETATION NDVI Temporal Series. *Ecological Modelling*, 194(4):429–434. DOI: 10.1016/j.ecolmodel.2005.10.035. 80

Le, N. D., Martin, R. D., and Raftery, A. E. (1996). Modeling Flat Stretches, Bursts, and Outliers in Time Series Using Mixture Transition Distribution Models. *Journal of the American Statistical Association*, 91(436):1504–1515. DOI: 10.1080/01621459.1996.10476718. 17

Lee, J.-G., Han, J., and Li, X. (2008). Trajectory Outlier Detection: A Partition-and-Detect Framework. In *Proc. of the 2008 IEEE 24th Intl. Conf. on Data Engineering (ICDE)*, pages 140–149. DOI: 10.1109/ICDE.2008.4497422. 55

Lee, W. and Stolfo, S. J. (1998). Data Mining Approaches for Intrusion Detection. In *Proc. of the 7th Conf. on USENIX Security Symposium (SSYM)*, pages 6–20. 14

Lee, W., Stolfo, S. J., and Chan, P. K. (1997). Learning Patterns from Unix Process Execution Traces for Intrusion Detection. In *In AAAI Workshop on AI Approaches to Fraud Detection and Risk Management*, pages 50–56. 13

Li, X. and Han, J. (2007). Mining Approximate Top-K Subspace Anomalies in Multi-dimensional Time-series Data. In *Proc. of the* 33rd *Intl. Conf. on Very Large Data Bases (VLDB)*, pages 447–458. 14

Li, X., Han, J., and Kim, S. (2006). Motion-Alert: Automatic Anomaly Detection in Massive Moving Objects. In *Proc. of the* 4th *IEEE Intl. Conf. on Intelligence and Security Informatics (ISI)*, pages 166–177. DOI: 10.1007/11760146_15. 14, 58

Li, X., Han, J., Kim, S., and Gonzalez, H. (2007). ROAM: Rule- and Motif-Based Anomaly Detection in Massive Moving Object Data Sets. In *Proc. of the* 7th *SIAM Intl. Conf. on Data Mining (SDM)*, pages 273–284. DOI: 10.1137/1.9781611972771.25. 14

Li, X., Li, Z., Han, J., and Lee, J.-G. (2009). Temporal Outlier Detection in Vehicle Traffic Data. In *Proc. of the 2009 IEEE Intl. Conf. on Data Engineering (ICDE)*, pages 1319–1322. DOI: 10.1109/ICDE.2009.230. 55, 57, 81

Lin, J., Keogh, E., Fu, A., and Van Herle, H. (2005). Approximations to Magic: Finding Unusual Medical Time Series. In *Proc. of the* 18th *IEEE Symposium on Computer-Based Medical Systems (CBMS)*, pages 329–334. DOI: 10.1109/CBMS.2005.34. 19

Lin, J., Keogh, E., Lonardi, S., and Chiu, B. (2003). A Symbolic Representation of Time Series, with Implications for Streaming Algorithms. In *Proc. of the* 8th *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 2–11. DOI: 10.1145/882082.882086. 16

Lin, J., Keogh, E., Wei, L., and Lonardi, S. (2007). Experiencing SAX: A Novel Symbolic Representation of Time Series. *Data Mining and Knowledge Discovery (DMKD)*, 15(2):107–144. DOI: 10.1007/s10618-007-0064-z. 16, 19

Liu, C., Xiong, H., Ge, Y., Geng, W., and Perkins, M. (2012). A Stochastic Model for Context-Aware Anomaly Detection in Indoor Location Traces. In *Proc. of the* 12th *IEEE Intl. Conf. on Data Mining (ICDM)*, pages 449–458. DOI: 10.1109/ICDM.2012.69. 81

Lu, C.-T. and Liang, L. R. (2004). Wavelet Fuzzy Classification for Detecting and Tracking Region Outliers in Meteorological Data. In *Proc. of the* 12th *Annual ACM Intl. Workshop on Geographic Information Systems (GIS)*, pages 258–265. DOI: 10.1145/1032222.1032260. 52, 54, 80

Luceno, A. (1998). Detecting Possibly Non-Consecutive Outliers in Industrial Time Series. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 60(2):295–310. DOI: 10.1111/1467-9868.00126. 17

Ma, J. and Perkins, S. (2003a). Online Novelty Detection on Temporal Sequences. In *Proc. of the* 9th *ACM Intl. Conf on Knowledge Discovery and Data Mining (KDD)*, pages 613–618. DOI: 10.1145/956750.956828. 17

Ma, J. and Perkins, S. (2003b). Time-series Novelty Detection using One-class Support Vector Machines. In *Proc. of the Intl. Joint Conf. on Neural Networks (IJCNN)*, volume 3, pages 1741–1745. DOI: 10.1109/IJCNN.2003.1223670. 10, 11

Marceau, C. (2000). Characterizing the Behavior of a Program using Multiple-length N-grams. In *Proc. of the 2000 Workshop on New Security Paradigms (NSPW)*, pages 101–110. DOI: 10.1145/366173.366197. 12

Markou, M. and Singh, S. (2003a). Novelty Detection: A Review - Part 1: Statistical Approaches. *Signal Processing*, 83(12):2481–2497. DOI: 10.1016/j.sigpro.2003.07.018. 90

Markou, M. and Singh, S. (2003b). Novelty Detection: A Review - Part 2: Neural Network Based Approaches. *Signal Processing*, 83(12):2499–2521. DOI: 10.1016/j.sigpro.2003.07.019. 90

Masud, M. M., Al-Khateeb, T., Khan, L., Aggarwal, C. C., Gao, J., Han, J., and Thuraisingham, B. M. (2011). Detecting Recurring and Novel Classes in Concept-Drifting Data Streams. In *Proc. of the* 11^{th} *IEEE Intl. Conf. on Data Mining (ICDM)*, pages 1176–1181. DOI: 10.1109/ICDM.2011.49. 36

Masud, M. M., Chen, Q., Khan, L., Aggarwal, C. C., Gao, J., Han, J., Srivastava, A., and Oza, N. C. (2013). Classification and Adaptive Novel Class Detection of Feature-Evolving Data Streams. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 25(7):1484–1497. DOI: 10.1109/TKDE.2012.109. 36

Masud, M. M., Chen, Q., Khan, L., Aggarwal, C. C., Gao, J., Han, J., and Thuraisingham, B. M. (2010). Addressing Concept-Evolution in Concept-Drifting Data Streams. In *Proc. of the* 10^{th} *IEEE Intl. Conf. on Data Mining (ICDM)*, pages 929–934. DOI: 10.1109/ICDM.2010.160. 36

Mathioudakis, M., Bansal, N., and Koudas, N. (2010). Identifying, Attributing and Describing Spatial Bursts. *Proc. of the Very Large Databases (PVLDB)*, 3(1):1091–1102. DOI: 10.1109/ICDM.2010.160. 84

Michael, C. C. and Ghosh, A. (2000). Two State-based Approaches to Program-based Anomaly Detection. In *Proc. of the* 16^{th} *Annual Computer Security Applications Conf. (ACSAC)*, pages 21–30. DOI: 10.1109/ACSAC.2000.898854. 12

Mitsa, T. (2010). *Temporal Data Mining*. Chapman and Hall/CRC. DOI: 10.1201/9781420089776. 2

Muthukrishnan, S., Shah, R., and Vitter, J. (2004). Mining Deviants in Time Series Data Streams. In *Proc. of the* 16^{th} *Intl. Conf. on Scientific and Statistical Database Management (SSDBM)*, pages 41–50. DOI: 10.1109/SSDBM.2004.51. 18

Nairac, A., Townsend, N., Carr, R., King, S., Cowley, P., and Tarassenko, L. (1999). A System for the Analysis of Jet Engine Vibration Data. *Integrated Computer-Aided Engineering*, 6(1):53–66. 7, 10, 80

Neill, D. B. and Cooper, G. F. (2010). A Multivariate Bayesian Scan Statistic for Early Event Detection and Characterization. *Machine Learning*, 79(3):261–282. DOI: 10.1007/s10994-009-5144-4. 54

Neill, D. B., McFowland, E., and Zheng, H. (2013). Fast Subset Scan for Multivariate Event Detection. *Statistics in Medicine*, 32(13):2185–2208. DOI: 10.1002/sim.5675. 54

Otey, M. E., Ghoting, A., and Parthasarathy, S. (2006). Fast Distributed Outlier Detection in Mixed-Attribute Data Sets. *Data Mining and Knowledge Discovery (DMKD)*, 12(2-3):203–228. DOI: 10.1007/s10618-005-0014-6. 41, 44, 85

Palpanas, T., Papadopoulos, D., Kalogeraki, V., and Gunopulos, D. (2003). Distributed Deviation Detection in Sensor Networks. *SIGMOD Records*, 32(4):77–82. DOI: 10.1145/959060.959074. 47

Pan, X., Tan, J., Kavulya, S., Gandhi, R., and Narasimhan, P. (2010). Ganesha: BlackBox Diagnosis of MapReduce Systems. *SIGMETRICS Performance Evaluation Review*, 37(3):8–13. DOI: 10.1145/1710115.1710118. 10

Papadimitriou, P., Dasdan, A., and Garcia-Molina, H. (2008). Web Graph Similarity for Anomaly Detection. In *Proc. of the* 17th *Intl. Conf. on World Wide Web (WWW)*, pages 1167–1168. DOI: 10.1007/s13174-010-0003-x. 61, 62, 63, 65, 66, 67, 84

Papadimitriou, P., Dasdan, A., and Garcia-Molina, H. (2010). Web Graph Similarity for Anomaly Detection. *Journal of Internet Services and Applications*, 1(1):19–30. DOI: 10.1007/s13174-010-0003-x. 61, 63, 65, 66, 84

Papadimitriou, S., Sun, J., and Faloutsos, C. (2005). Streaming Pattern Discovery in Multiple Time-series. In *Proc. of the* 31st *Intl. Conf. on Very Large Data Bases (VLDB)*, pages 697–708. 16

Pincombe, B. (2005). Anomaly Detection in Time Series of Graphs using ARMA Processes. *ASOR Bulletin*, 24(4):2–10. 61, 62, 63, 64, 65

Pokrajac, D., Lazarevic, A., and Latecki, L. J. (2007). Incremental Local Outlier Detection for Data Streams. In *Proc. of the IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, pages 504–515. IEEE. DOI: 10.1109/CIDM.2007.368917. 29, 31, 81

Portnoy, L., Eskin, E., and Stolfo, S. (2001). Intrusion Detection with Unlabeled Data using Clustering. In *Proc. of the ACM CSS Workshop on Data Mining Applied to Security (DMSA)*, pages 5–8. 7, 10, 82

Priebe, C. E., Conroy, J. M., Marchette, D. J., and Park, Y. (2005). Scan Statistics on Enron Graphs. *Computational & Mathematical Organization Theory*, 11(3):229–247. DOI: 10.1007/s10588-005-5378-z. 61, 67, 85

Qi, G.-J., Aggarwal, C. C., and Huang, T. S. (2012). On Clustering Heterogeneous Social Media Objects with Outlier Links. In *Proc. of the* 5^{th} *ACM Intl. Conf. on Web Search and Data Mining (WSDM)*, pages 553–562. DOI: 10.1145/2124295.2124363. 68, 70

Qiao, Y., Xin, X., Bin, Y., and Ge, S. (2002). Anomaly Intrusion Detection Method based on HMM. *Electronics Letters*, 38(13):663–664. DOI: 10.1049/el:20020467. 13

Rajasegarar, S., Leckie, C., Palaniswami, M., and Bezdek, J. C. (2007). Quarter Sphere based Distributed Anomaly Detection in Wireless Sensor Networks. In *IEEE Intl. Conf. on Communications (ICC)*, pages 3864–3869. DOI: 10.1109/ICC.2007.637. 43

Rebbapragada, U., Protopapas, P., Brodley, C. E., and Alcock, C. (2009). Finding Anomalous Periodic Time Series. *Journal of Machine Learning*, 74(3):281–313. DOI: 10.1007/s10994-008-5093-3. 10

Rousseeuw, P. J. and Leroy, A. M. (1987). *Robust Regression and Outlier Detection*. John Wiley & Sons, Inc. DOI: 10.1002/0471725382. 7

Salvador, S. and Chan, P. (2005). Learning States and Rules for Detecting Anomalies in Time Series. *Applied Intelligence*, 23(3):241–255. DOI: 10.1007/s10489-005-4610-3. 12, 13

Sequeira, K. and Zaki, M. (2002). ADMIT: Anomaly-based Data Mining for Intrusions. In *Proc. of the* 8^{th} *ACM Intl. Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 386–395. DOI: 10.1145/775047.775103. 8, 10, 24, 25, 82

Shahabi, C., Tian, X., and Zhao, W. (2000). TSA-Tree: A Wavelet-Based Approach to Improve the Efficiency of Multi-Level Surprise and Trend Queries on Time-Series Data. In *Proc. of the* 12^{th} *Intl. Conf. on Scientific and Statistical Database Management (SSDBM)*, pages 55–68. DOI: 10.1109/SSDM.2000.869778. 19

Shoubridge, P., Kraetzl, M., and Ray, D. (1999). Detection of Abnormal Change in Dynamic Networks. In *Proc. of the Intl. Conf. on Information, Decision and Control*, pages 557–562. DOI: 10.1109/IDC.1999.754216. 62, 63

Silverman, B. W. (1986). *Density Estimation for Statistics and Data Analysis*, volume 26. CRC Press. DOI: 10.1007/978-1-4899-3324-9. 35

Silvestri, G., Verona, F., Innocenti, M., and Napolitano, M. (1994). Fault Detection using Neural Networks. In *Proc. of the 1994 IEEE Intl. Conf. on Neural Networks*, volume 6, pages 3796–3799. DOI: 10.1109/ICNN.1994.374815. 18

Stolpe, M., Bhaduri, K., Das, K., and Morik, K. (2013). Anomaly Detection in Vertically Partitioned Data by Distributed Core Vector Machines. In *Proc. of the 2013 European Conf. on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, pages 321–336. DOI: 10.1007/978-3-642-40994-3_21. 48

Subramaniam, S., Palpanas, T., Papadopoulos, D., Kalogeraki, V., and Gunopulos, D. (2006). Online Outlier Detection in Sensor Data using Non-parametric Models. In *Proc. of the* 32nd *Intl. Conf. on Very Large Data Bases (VLDB)*, pages 187–198. 45, 46

Sun, J., Faloutsos, C., Papadimitriou, S., and Yu, P. S. (2007). GraphScope: Parameter-free Mining of Large Time-evolving Graphs. In *Proc. of the* 13th *ACM Intl. Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 687–696. DOI: 10.1145/1281192.1281266. 71, 73

Sun, P., Chawla, S., and Arunasalam, B. (2006). Mining for Outliers in Sequential Databases. In *Proc. of the* 6th *SIAM Intl. Conf. on Data Mining (SDM)*, pages 94–105. 13

Sun, Y., Xie, K., Ma, X., Jin, X., Pu, W., and Gao, X. (2005). Detecting Spatio-Temporal Outliers in Climate Dataset: A Method Study. In *Proc. of the 2005 IEEE Intl. Geoscience and Remote Sensing Symposium (IGARSS)*, pages 760–763. DOI: 10.1109/IGARSS.2005.1525218. 49, 78

Sun, Y., Yu, Y., and Han, J. (2009). Ranking-based Clustering of Heterogeneous Information Networks with Star Network Schema. In *Proc. of the* 15th *ACM Intl. Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 797–806. DOI: 10.1145/1557019.1557107. 73

Szymanski, B. and Zhang, Y. (2004). Recursive Data Mining for Masquerade Detection and Author Identification. In *Proc. of the* 5th *Annual IEEE Systems, Man, and Cybernetics - Information Assurance Workshop*, pages 424–431. DOI: 10.1109/IAW.2004.1437848. 7, 10, 11, 82

Tian, S., Mu, S., and Yin, C. (2007). Sequence-similarity Kernels for SVMs to Detect Anomalies in System Calls. *Neurocomputing*, 70(4-6):859–866. DOI: 10.1016/j.neucom.2006.10.017. 14

Tsay, R. S. (1986). Time Series Model Specification in the Presence of Outliers. *Journal of the American Statistical Association*, 81(393):132–141. DOI: 10.1080/01621459.1986.10478250. 18

Tsay, R. S., Pena, D., and Pankratz, A. E. (2000). Outliers in Multivariate Time Series. *Biometrika*, 87(4):789–804. DOI: 10.1093/biomet/87.4.789. 17

Wang, M., Zhang, C., and Yu, J. (2006). Native API Based Windows Anomaly Intrusion Detection Method Using SVM. In *Proc. of the IEEE Intl. Conf. on Sensor Networks, Ubiquitous, and Trustworthy Computing - Vol 1 (SUTC)*, pages 514–519. DOI: 10.1109/SUTC.2006.95. 14

Warrender, C., Forrest, S., and Pearlmutter, B. (1999). Detecting Intrusions using System Calls: Alternative Data Models. In *Proc. of the 1999 IEEE Symposium on Security and Privacy*, pages 133–145. DOI: 10.1109/SECPRI.1999.766910. 82

Wei, L., Keogh, E., and Xi, X. (2006). SAXually Explicit Images: Finding Unusual Shapes. In *Proc. of the* 6th *Intl. Conf. on Data Mining (ICDM)*, pages 711–720. DOI: 10.1109/ICDM.2006.138. 18, 19, 83, 84

Wei, L., Kumar, N., Lolla, V., Keogh, E. J., Lonardi, S., and Ratanamahatana, C. (2005). Assumption-free Anomaly Detection in Time Series. In *Proc. of the* 17th *Intl. Conf. on Scientific and Statistical Database Management (SSDBM)*, pages 237–240. 20

Williams, A. W., Pertet, S. M., and Narasimhan, P. (2007). Tiresias: Black-box Failure Prediction in Distributed Systems. In *Proc. of the* 21st *Intl. Parallel and Distributed Processing Symposium (IPDPS)*, pages 1–8. DOI: 10.1109/IPDPS.2007.370345. 18

Wu, E., Liu, W., and Chawla, S. (2010). Spatio-Temporal Outlier Detection in Precipitation Data. In *Proc. of the* 2nd *Intl. Conf. on Knowledge Discovery from Sensor Data (Sensor-KDD)*, pages 115–133. DOI: 10.1007/978-3-642-12519-5_7. 52, 53, 78

Yamanishi, K. and Takeuchi, J.-i. (2002). A Unifying Framework for Detecting Outliers and Change Points from Non-stationary Time Series Data. In *Proc. of the* 8th *ACM Intl. Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 676–681. DOI: 10.1145/775047.775148. 21, 22

Yamanishi, K., Takeuchi, J.-i., Williams, G., and Milne, P. (2004). On-Line Unsupervised Outlier Detection Using Finite Mixtures with Discounting Learning Algorithms. *Data Mining and Knowledge Discovery*, 8(3):275–300. DOI: 10.1023/B:DAMI.0000023676.72185.7c. 22

Yang, D., Rundensteiner, E. A., and Ward, M. O. (2009). Neighbor-based Pattern Detection for Windows over Streaming Data. In *Proc. of the* 12th *Intl. Conf. on Extending Database Technology: Advances in Database Technology (EDBT)*, pages 529–540. DOI: 10.1145/1516360.1516422. 31

Yang, J. and Wang, W. (2003). CLUSEQ: Efficient and Effective Sequence Clustering. In *Proc. of the* 19th *Intl. Conf. on Data Engineering (ICDE)*, pages 101–112. DOI: 10.1109/ICDE.2003.1260785. 13

Yankov, D., Keogh, E., and Rebbapragada, U. (2008). Disk Aware Discord Discovery: Finding Unusual Time Series in Terabyte Sized Datasets. *Knowledge and Information Systems (KAIS)*, 17(2):241–262. DOI: 10.1007/s10115-008-0131-9. 19, 81, 84

Ye, N. (2000). A Markov Chain Model of Temporal Behavior for Anomaly Detection. In *Proc. of the 2000 IEEE Systems, Man, and Cybernetics - Information Assurance and Security Workshop*, volume 166, pages 171–174. 8, 13, 82

Yu, W., Aggarwal, C. C., Ma, S., and Wang, H. (2013). On Anomalous Hotspot Discovery in Graph Streams. In *Proc. of the* 13*th* *IEEE Intl. Conf. on Data Mining (ICDM)*. IEEE. DOI: 10.1109/ICDM.2013.32. 75

Zhang, J., Gao, Q., and Wang, H. (2008). SPOT: A System for Detecting Projected Outliers From High-dimensional Data Streams. In *Proc. of the 2008 IEEE* 24*th* *Intl. Conf. on Data Engineering (ICDE)*, pages 1628–1631. DOI: 10.1109/ICDE.2008.4497638. 32

Zhang, X., Fan, P., and Zhu, Z. (2003). A New Anomaly Detection Method based on Hierarchical HMM. In *Proc. of the* 4*th* *Intl. Conf. on Parallel and Distributed Computing, Applications and Technologies (PDCAT)*, pages 249–252. DOI: 10.1109/PDCAT.2003.1236299. 13

Zhang, Y., Meratnia, N., and Havinga, P. J. M. (2008). Outlier Detection Techniques For Wireless Sensor Networks: A Survey. Technical Report TR-CTIT-08-59, Centre for Telematics and Information Technology University of Twente. DOI: 10.1109/SURV.2010.021510.00088. 1, 39

Zhu, Y. and Shasha, D. (2003). Efficient Elastic Burst Detection in Data Streams. In *Proc. of the* 9*th* *ACM Intl. Conf on Knowledge Discovery and Data Mining (KDD)*, pages 336–345. DOI: 10.1145/956750.956789. 20, 84, 86

Authors' Biographies

MANISH GUPTA

Manish Gupta is an applied researcher at Microsoft Bing, India. He is also an adjunct faculty at the International Institute of Information Technology, Hyderabad (IIIT-H), India. He received his Masters in Computer Science from IIT Bombay in 2007 and his Ph.D. in Computer Science from University of Illinois at Urbana Champaign in 2013. He worked for Yahoo! Bangalore from 2007 to 2009. His research interests are in the areas of data mining, information retrieval, and web mining.

JING GAO

Jing Gao received her Ph.D. from University of Illinois at Urbana Champaign in 2011. She is currently an assistant professor in the Computer Science and Engineering Department of the State University of New York at Buffalo. She was a recipient of an IBM Ph.D. fellowship and is broadly interested in data and information analysis with a focus on information integration, ensemble methods, transfer learning, anomaly detection, and mining data streams. She is a member of the IEEE.

CHARU C. AGGARWAL

Charu C. Aggarwal is a Research Scientist at the IBM T. J. Watson Research Center in Yorktown Heights, New York. He completed his Ph.D. from Massachusetts Institute of Technology in 1996. He has since worked in the field of performance analysis, databases, and data mining. He has published over 200 papers in refereed conferences and journals, and has applied for, or been granted, over 80 patents. He has received the IBM Corporate Award (2003), IBM Outstanding Innovation Award (2008), IBM Research Division Award (2008), and Master Inventor at IBM three times. He is a fellow of the ACM and IEEE.

JIAWEI HAN

Jiawei Han is the Abel Bliss Professor in the Department of Computer Science at the University of Illinois. He has been researching data mining, information network analysis, and database systems, with more than 500 publications. He received the ACM SIGKDD Innovation Award (2004), IEEE Computer Society Technical Achievement Award (2005), and IEEE W. Wallace McDowell Award (2009). His book *Data Mining: Concepts and Techniques* (Morgan Kaufmann) has been used worldwide as a textbook. He is a fellow of the ACM and IEEE.