Charu C. Aggarwal

Outlier Analysis

Outlier Analysis

Charu C. Aggarwal

Outlier Analysis

Springer

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

To my wife, Lata and
my daughter, Sayani

Contents

Preface

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

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

- P. Rousseeuw and A. Leroy. Robust Regression and Outlier Detection. *Wiley*, 2003.

- V. Barnett and T. Lewis. Outliers in Statistical Data, *Wiley*, 1994.

- D. Hawkins. Identification of Outliers, *Chapman and Hall*, 1980.

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

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

Acknowledgments

I would like to thank my wife and daughter for their love and support during the writing of this book. The writing of a book requires significant time which is taken away from family members. This book is the result of their patience with me during this time.

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

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

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

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

Chapter 1

AN INTRODUCTION TO OUTLIER ANALYSIS

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

1.   Introduction

An outlier is a data point which is significantly different from the remaining data. Hawkins formally defined [205] the concept of an outlier as follows:

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

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

- **Intrusion Detection Systems:** In many host-based or networked computer systems, different kinds of data are collected about the operating system calls, network traffic, or other activity in the system. This data may show unusual behavior because of malicious

activity. The detection of such activity is referred to as intrusion detection.

- **Credit Card Fraud:** Credit card fraud is quite prevalent, because of the ease with which sensitive information such as a credit card number may be compromised. This typically leads to unauthorized use of the credit card. In many cases, unauthorized use may show different patterns, such as a buying spree from geographically obscure locations. Such patterns can be used to detect outliers in credit card transaction data.

- **Interesting Sensor Events:** Sensors are often used to track various environmental and location parameters in many real applications. The sudden changes in the underlying patterns may represent events of interest. Event detection is one of the primary motivating applications in the field of sensor networks.

- **Medical Diagnosis:** In many medical applications the data is collected from a variety of devices such as MRI scans, PET scans or ECG time-series. Unusual patterns in such data typically reflect disease conditions.

- **Law Enforcement:** Outlier detection finds numerous applications to law enforcement, especially in cases, where unusual patterns can only be discovered over time through multiple actions of an entity. Determining fraud in financial transactions, trading activity, or insurance claims typically requires the determination of unusual patterns in the data generated by the actions of the criminal entity.

- **Earth Science:** A significant amount of spatiotemporal data about weather patterns, climate changes, or land cover patterns is collected through a variety of mechanisms such as satellites or remote sensing. Anomalies in such data provide significant insights about hidden human or environmental trends, which may have caused such anomalies.

In all these applications, the data has a "normal" model, and anomalies are recognized as deviations from this normal model. In many cases such as intrusion or fraud detection, the outliers can only be discovered as a sequence of multiple data points, rather than as an individual data point. For example, a fraud event may often reflect the actions of an individual in a particular sequence. The specificity of the sequence is relevant to identifying the anomalous event. Such anomalies are also referred to as *collective anomalies*, because they can only be inferred collectively from
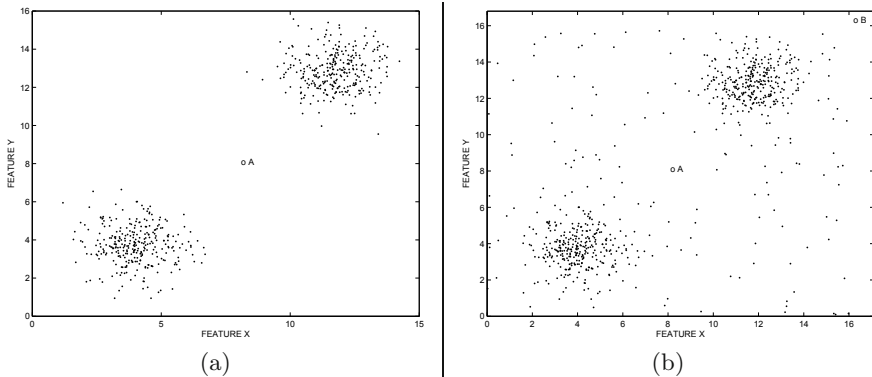
Figure 1.1. The difference between noise and anomalies

a *set or sequence* of data points. Such collective anomalies typically represent unusual *events*, which need to be discovered from the data. This book will address these different kinds of anomalies.

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

- Most outlier detection algorithm output a score about the level of "outlierness" of a data point. This can be used in order to determine a ranking of the data points in terms of their outlier tendency. This is a very general form of output, which retains all the information provided by a particular algorithm, but does not provide a concise summary of the small number of data points which should be considered outliers.

- A second kind of output is a binary label indicating whether a data point is an outlier or not. While some algorithms may directly return binary labels, the outlier scores can also be converted into binary labels. This is typically done by imposing thresholds on outlier scores, based on their statistical distribution. A binary labeling contains less information than a scoring mechanism, but it is the final result which is often needed for decision making in practical applications.

It is often a subjective judgement, as to what constitutes a "sufficient" deviation for a point to be considered an outlier. In real applications, the data may be embedded in a significant amount of noise, and such noise may not be of any interest to the analyst. It is usually the *significantly interesting deviations* which are of interest. In order to illustrate this point, consider the examples illustrated in Figures 1.1(a) and (b). It is evident that the main patterns (or clusters) in the data are identical in both cases, though there are significant differences outside these main
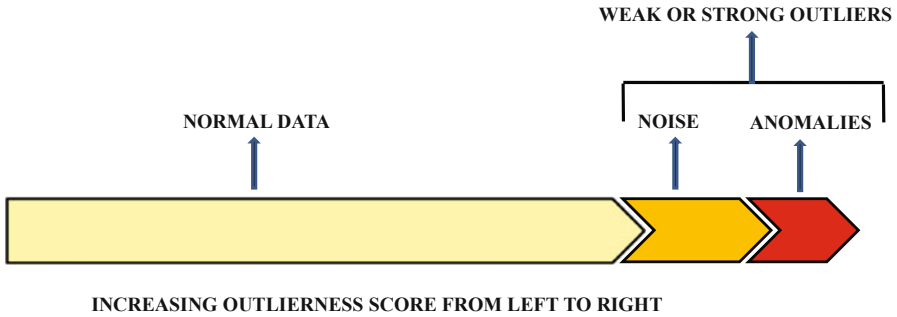
Figure 1.2.   The spectrum from normal data to outliers

clusters. In the case of Figure 1.1(a), a single data point (marked by 'A') seems to be very different from the remaining data, and is therefore very obviously an anomaly. The situation in Figure 1.1(b) is much more subjective. While the corresponding data point 'A' in Figure 1.1(b) is also in a sparse region of the data, it is much harder to state confidently that it represents a true deviation from the remaining data set. It is quite likely that this data point represents randomly distributed noise in the data. This is because the point 'A' seems to fit a pattern represented by other randomly distributed points. Therefore, throughout this book the term "outlier" refers to a data point, which could either be considered an abnormality or noise, whereas an "anomaly" refers to a special kind of outlier, which is of interest to an analyst.

In the *unsupervised scenario*, where previous examples of interesting anomalies are not available, the noise represents the semantic boundary between normal data and true anomalies– noise is often modeled as a weak form of outliers, which does not always meet the strong criteria necessary for a data point to be considered interesting or anomalous enough. For example, data points at the boundaries of clusters may often be considered noise. Typically, most outlier detection algorithms use some quantified measure of the *outlierness* of a data point, such as the sparsity of the underlying region, nearest neighbor based distance, or the fit to the underlying data distribution. Every data point lies on a continuous spectrum from normal data to noise, and finally to anomalies, as illustrated in Figure 1.2. The separation of the different regions of this spectrum is often not precisely defined, and is chosen on an ad-hoc basis according to application-specific criteria. Furthermore, the separation between noise and anomalies is not pure, and many data points created by a noisy generative process may be deviant enough to be interpreted

as anomalies on the basis of the outlier score. Thus, anomalies will *typically* have a much higher outlier score than noise, but this is not a distinguishing factor between the two as a matter of *definition*. Rather, it is the interest of the analyst, which regulates the distinction between noise and an anomaly.

Some authors use the terms *weak outliers* and *strong outliers* in order to distinguish between noise and anomalies [4, 262]. The detection of noise in the data has numerous applications of its own. For example, the removal of noise creates a much cleaner data set, which can be utilized for other data mining algorithms. While noise may not be interesting in its own right, its *removal and identification* continues to be an important problem for mining purposes. Therefore, both noise and anomaly detection problems are important enough to be addressed in this book. Throughout this book, methods specifically relevant to *either* anomaly detection or noise removal will be identified. However, the bulk of the outlier detection algorithms could be used for either problem, since the difference between them is really one of semantics.

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

Several levels of supervision are possible in practical scenarios. In the fully supervised scenario, examples of both normal and abnormal data are available, and can be clearly distinguished. In some cases, examples

of outliers are available, but the examples of "normal" data may also contain outliers in some (unknown) proportion. This is referred to as classification with positive and unlabeled data. In other semi-supervised scenarios, only examples of normal data or only examples of anomalous data may be available. Thus, the number of variations of the problem are rather large, each of which requires a related but dedicated set of techniques.

Finally, the data representation may vary widely across applications. For example, the data may be purely multidimensional with no relationships among points, or the data may be sequential with temporal ordering, or may be defined in the form of a network with arbitrary relationships among data points. Furthermore, the attributes in the data may be numerical, categorical or may be mixed. Clearly, the outlier detection process needs to be sensitive to the nature of the attributes and relationships in the underlying data. In fact, the relationships themselves may often provide a criterion for outlier detection, in the form of connections between entities which do not usually occur together. Such outliers are referred to as *contextual* outliers. A classical example of this is the concept of *linkage outliers* in social network analysis [15]. In this case, entities (nodes) in the graph, which are normally not connected together may show *anomalous* connections with each other. Thus, the impact of data types on the anomaly detection process is significant, and will be carefully addressed in this book.

This chapter is organized as follows. In section 2, the importance of data modeling in outlier analysis is discussed. In section 3, the basic outlier models for outlier detection are introduced. Meta-algorithms for outlier analysis are addressed in section 4. Section 5 discusses the basic data types used for analysis. Section 6 introduces the concept of supervised modeling of outliers for data analysis. Methods for evaluating outlier detection algorithms are discussed in section 7. The conclusions are presented in section 8.

2.   The Data Model is Everything

Virtually all outlier detection algorithms create a model of the normal patterns in the data, and then compute an outlier score of a given data point on the basis of the deviations from these patterns. For example, this data model may be a generative model such as a gaussian mixture model, a regression-based model, or a proximity-based model. All these models make different assumptions about the "normal" behavior of the data. The outlier score of a data point is then computed by evaluating the quality of the fit between the data point and the model. In many

(a) Normal Distribution   (b) Zipf Distribution
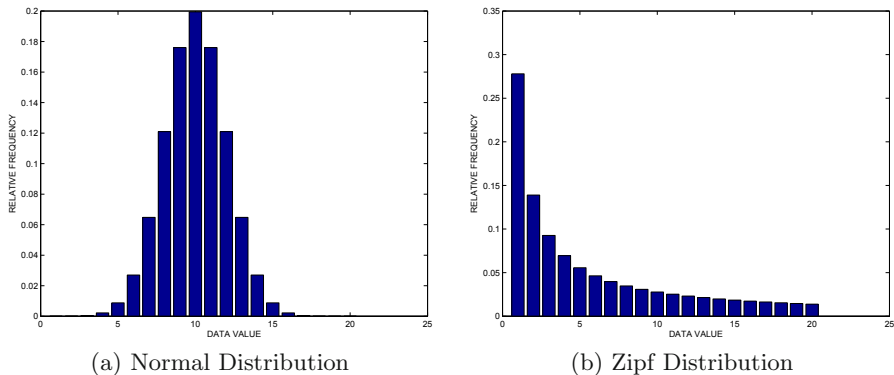
Figure 1.3. Applying Z-value test on the Normal and Zipf distributions

cases, the model may be algorithmically defined. For example, nearest neighbor-based outlier detection algorithms model the outlier tendency of a data point in terms of the distribution of its k-nearest neighbor distance. Thus, in this case, the assumption is that outliers are located at large distances from most of the data.

Clearly, the choice of the data model is crucial. An incorrect choice of data model may lead to poor results. For example, a fully generative model such as the gaussian mixture model may not work well, if the data does not fit the generative assumptions of the model, or if a sufficient number of data points are not available to learn the parameters of the model. Similarly, a linear regression-based model may work poorly, if the underlying data is clustered arbitrarily. In such cases, data points may be incorrectly reported as outliers *because of poor fit to the erroneous assumptions of the model*. In practice, the choice of the model is often dictated by the analyst's understanding of the kinds of deviations relevant to an application. For example, in a spatial application measuring a behavioral attribute such as the location-specific temperature, it would be reasonable to assume that unusual deviations of the temperature attribute in a spatial locality is a indicator of abnormality. On the other hand, for the case of high-dimensional data, even the definition of data locality may be ill-defined because of data sparsity. Thus, an effective model for a particular data domain may only be constructed after carefully evaluating the relevant modeling properties of that domain.

In order to understand the impact of the model, it is instructive to examine the use of a simple model known as the *Z-value test* for outlier analysis. Consider a set of 1-dimensional quantitative data observations, denoted by $X_1 \ldots X_N$, with mean μ and standard deviation σ. The Z-

value for the data point X_i is denoted by Z_i, and is defined as follows:

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

The Z-value test computes the number of standard deviations by which the data varies from the mean. This provides a good proxy for the outliers in the data. An implicit assumption is that the data is modeled from a normal distribution. In cases where mean and standard deviation of the distribution can be accurately estimated (or are available from domain knowledge), a good "rule of thumb" is to use $Z_i \geq 3$ as a proxy for the anomaly. However, in many scenarios, where a smaller number of samples are available, the mean and standard deviation of the underlying distribution cannot be estimated accurately. In such cases, the results from the Z-value test need to be interpreted more carefully. This issue will be discussed in Chapter 2.

It is often forgotten by practitioners during outlier modeling, that the test implicitly assumes an approximately normal distribution for the underlying data. When this is not the case, the corresponding Z-values need to be interpreted carefully. For example, consider the two data frequency histograms drawn on values between 1 and 20 in Figure 1.3. In the first case, the histogram is sampled from a normal distribution with $(\mu, \sigma) = (10, 2)$, and in the second case, it is sampled from a Zipf distribution $1/i$. It is evident that most of the data lies in the range $[10 - 2 * 3, 10 + 2 * 3]$ for the normal distribution, and all data points lying outside this range can be truly considered anomalies. Thus, the Z-value test works very well in this case. In the second case with the Zipf distribution, the anomalies are not quite as clear, though the data with very high values (close to 20) can probably be considered anomalies. In this case, the mean and standard deviation of the data are 5.24 and 5.56 respectively. As a result, the Z-value test does not declare *any* of the data points as anomaly (for a threshold of 3), though it does come close. In any case, the significance of the Z-value from the Zipf-distribution is not very meaningful at least from the perspective of distribution of probabilities. This suggests that if mistakes are made at the modeling stage, it can result in an incorrect understanding of the data. While such tests are often used as a *heuristic* to provide a rough idea of the outlier scores even for data sets which are far from normally distributed, it is important to interpret such scores carefully.

An example in which the Z-value test would not work even as a heuristic, would be one in which it was applied to a data point, which was an outlier only because of its relative position, rather than its extreme position. For example, if the Z-value test is applied to an individual di-

(a) 2-d data   (b) 3-d data
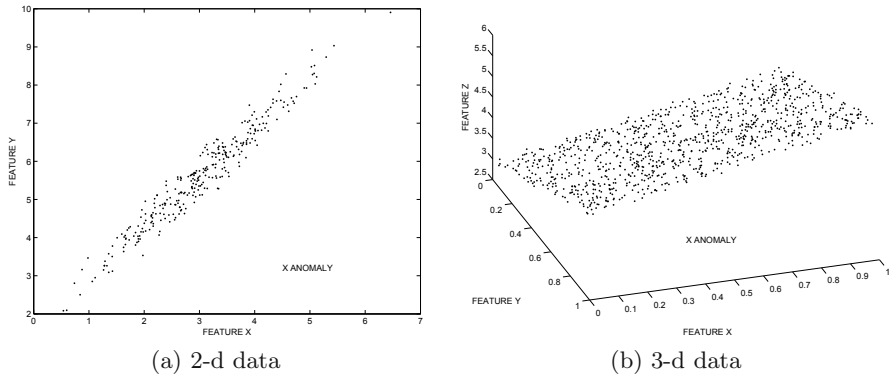
Figure 1.4. Linearly Correlated Data

mension in Figure 1.1(a), the test would fail miserably, because point A would be considered the most centrally located and normal data point. On the other hand, the test can still be reasonably applied to a set of *extracted* 1-dimensional values corresponding to the k-nearest neighbor distances of each point. Therefore, the effectiveness of a model depends both on the choice of the test used, and *how* it is applied.

The best choice of a model is often data set specific. This requires a good understanding of the data itself before choosing the model. For example, a regression-based model would be most suitable for finding the outliers in the data distributions of Figure 1.4, where most of the data is distributed along linear correlation planes. On the other hand, a clustering model would be more suitable for the cases illustrated in Figure 1.1. An attempt to use the wrong model for a given data set is likely to provide poor results. *Therefore, the core principle of discovering outliers is based on assumptions about the structure of the normal patterns in a given data set. Clearly, the choice of the "normal" model depends highly upon the analyst's understanding of the natural data patterns in that particular domain.*

There is no way around this issue; a highly general model with too many parameters will most likely overfit the data, and will also find a way to fit the outliers. A simple model, which is constructed with a good intuitive understanding of the data (and possibly also an understanding of what the analyst is looking for), is likely to lead to much better results. On the other hand, an oversimplified model, which fits the data poorly is likely to declare normal patterns as outliers. The initial stage of selecting the data model is perhaps the most crucial one in outlier analysis. The theme about the impact of data models will be repeated throughout the book, with specific examples.

3.   The Basic Outlier Models

This section will present the broad diversity of the models in the literature, and provide some idea of the impact of using different data models. A detailed discussion of these methods are provided in later chapters. Several factor influence the choice of an outlier model, including the data type, data size, availability of relevant outlier examples, and the need for interpretability in a model. The last of these criteria deserves some further explanation.

The *interpretability* of an outlier detection model is extremely important from the perspective of the analyst. It is often desirable to determine *why* a particular data point is an outlier in terms of its relative behavior with respect to the remaining data. This provides the analyst further hints about the diagnosis required in an application-specific scenario. This is also referred to as the *intensional knowledge* about the outliers [262]. Different models have different levels of interpretability. Typically, models which work with the original attributes, and use fewer transforms on the data such as principal component analysis have higher interpretability. While data transformations can sometimes enhance the contrast between the outliers and normal data points, such transformations do come at the expense of interpretability. Therefore, it is critical to keep these factors in mind, while choosing a specific model for outlier analysis.

3.1   Extreme Value Analysis

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

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

Extreme value statistics [364] is distinct from the traditional definition of outliers. The traditional definition of outliers, as provided by Hawkins, defines such objects by their *generative probabilities* rather than the extremity in their values. For example, in the data set $\{1, 2, 2, 50, 98, 98, 99\}$ of 1-dimensional values, the values 1 and 99, could very mildly, be considered extreme values. On the other hand, the value 50 is the average of the data set, and is most definitely not an extreme value. However, most probabilistic and density-based models would classify the value 50 as the strongest outlier in the data, on the basis of Hawkins' definition of generative probabilities. Confusions between extreme value analysis and outlier analysis are common, especially in the context of multivariate data. This is quite often the case, since many extreme value models also use probabilistic models in order to quantify the probability that a data point is an extreme value.

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

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

Extreme value analysis can also be extended to multivariate data with the use of distance-, or depth-based methods [243, 288, 388]. However, these methods are applicable only to certain kinds of specialized scenarios, where outliers are known to be present at the boundaries of the data. Many forms of post-processing on multi-criterion outlier scores may use such methods. On the other hand, such methods have often not found much utility in the literature for *generic* outlier analysis, because of their inability to discover outlier in the sparse *interior* regions of a data set.

3.2   Probabilistic and Statistical Models

In probabilistic and statistical models, the data is modeled in the form of a closed form probability distribution, and the parameters of this model are learned. Thus, the key assumption here is about the choice of the data distribution with which the modeling is performed. For example, a gaussian mixture model is a generative model, which characterizes the data in the form of a generative process containing a mixture of gaussian clusters. The parameters of these gaussian distributions are learned with the use of an *Expectation-Maximization (EM)* algorithm on the data set. A key output of this method is the membership probability of the data points to the different clusters, as well as the density-based fit to the modeled distribution. This provides a natural way to model the outliers, because data points which have very low fit values may be considered outliers. As discussed earlier, an extreme value test may be applied to these probability values in order to determine the outliers.

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

A downside of probabilistic models is that they try to fit the data to a particular kind of distribution, which may often not be appropriate for the underlying data. Furthermore, as the number of model parameters increases, over-fitting becomes more common. In such cases, the outliers

may fit the underlying model of normal data. Many parametric models are also harder to interpret in terms of intensional knowledge, especially when the parameters of the model cannot be intuitively presented to an analyst in terms of underlying attributes. This can defeat one of the important purposes of anomaly detection, which is to provide diagnostic understanding of the abnormal data generative process. A detailed discussion of probabilistic methods, including the EM algorithm is provided in Chapter 2.

3.3 Linear Models

These methods model the data into lower dimensional embedded subspaces with the use of linear correlations [387]. For example, in the case of Figure 1.4, the data is aligned along a 1-dimensional line in a 2-dimensional space. The optimal line which passes through these points is determined with the use of regression analysis. Typically, a least squares fit is used to determine the optimal lower dimensional subspace. The distances of the data points from this plane are determined. Extreme values analysis can be applied on these deviations in order to determine the outliers. For example, in the 2-dimensional example of Figure 1.4, a linear model of the data points $\{(x_i, y_i), i \in \{1 \ldots N\}$ in terms of two coefficients a and b may be created as follows:

$$y_i = a \cdot x_i + b + \epsilon_i \quad \forall i \in \{1 \ldots N\} \tag{1.2}$$

Here ϵ_i represents the *residual*, which is essentially the error of the modeling. The coefficients a and b need to be *learned from the data* in order to minimize the least squares error denoted by $\sum_{i=1}^{N} \epsilon_i^2$. This is a convex non-linear programming problem, whose solution can be obtained either in closed form through either matrix operations (principal component analysis), or by gradient descent. The derived residuals can then be used in conjunction with extreme value analysis in order to determine the underlying outliers.

The concept of dimensionality reduction and principal component analysis (PCA) is quite similar [244], except that it uses a non-parametric approach in order to model the data correlations. PCA can be derived through multivariate regression analysis, by determining the plane which optimizes the least squares error of representation in terms of the normal distance to the plane. In other words, it provides the subspaces, such that by projecting the data into these subspaces, the aggregate least square errors of the residuals are minimized. The absolute sizes of these residuals can be analyzed in order to determine the outliers. Data points, which have large residuals, are more likely to be outliers, because their behavior does not conform to the natural subspace patterns in the

data. In addition, Principal Component Analysis techniques can be used for noise *correction* [18], where the attributes of data points are modified in order to reduce the noise in the data. Clearly, outlier data points are likely to be corrected more significantly than other data points.

Dimensionality reduction and regression modeling are particularly hard to interpret in terms of original attributes, when the underlying data dimensionality is high. This is because the subspace embedding is defined as a linear combination of attributes with positive or negative coefficients. This cannot easily be intuitively interpreted in terms specific properties of the data attributes. Dimensionality reduction and regression analysis methods for outlier detection are discussed in Chapter 3.

3.3.1   Spectral Models.   Many of the matrix decomposition methods such as PCA are also used in the context of graphs and networks. The main difference is in how the matrix is created for decomposition. Such methods are also referred to as spectral models, when applied to certain kinds of data such as graphs and networks. Spectral methods are used commonly for clustering graph data sets, and are also used in order to identify anomalous changes in temporal sequences of graphs [229]. Such spectral models will be discussed in Chapter 11.

3.4   Proximity-based Models

The idea in proximity-based methods is to model outliers as points which are isolated from the remaining data. This modeling may be performed in one of three ways. Specifically, the three methods are cluster analysis, density-based analysis or nearest neighbor analysis. In clustering and other density-based methods, the dense regions in the data are found directly, and outliers are defined as those points, which do not lie in these dense regions. The main difference between clustering and density-based methods is that clustering methods segment the *points*, whereas the density-based methods segment the *space*.

In nearest neighbor methods [261, 381], the distance of each data point to its kth nearest neighbor is determined. By picking a value of $k > 1$, small groups of points, which are close together, but far away from the remaining data set are also treated as outliers. It is reasonable to treat such sets of data points as outliers, because small related sets of points can often be generated by an anomalous process. For example, consider the case illustrated in Figure 1.5, which contains a large cluster containing 4000 data points, and a small set of isolated but three closely spaced and related anomalies. Such situations are quite common, because anomalies which are caused by the same (rare) process, may result
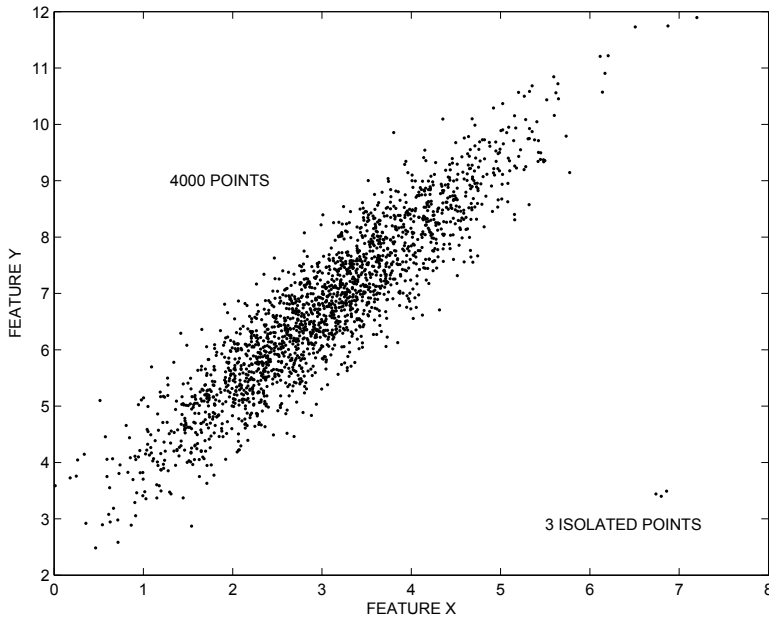
Figure 1.5. Small groups of anomalies can be a challenge to density-based methods

in small sets of data points which are similar to one another. In this case, the points within an anomaly set are close to one another, and cannot be distinguished on the basis of the 1-nearest neighbor distance. Such anomalies are often hard to distinguish from noise by using certain kinds of clustering and density-based algorithms, which are not sensitive to the global behavior of the data. On the other hand, the k-nearest neighbor approach can sometimes be effective. In the case of Figure 1.5, such sets of related anomalies may be identified by using $k \geq 3$. The kth nearest neighbor score provides an outlier score of the data set. This method can typically be computationally expensive, because it is required to determine the kth nearest neighbor of every point in the data set. Unless efficient indexing methods are available, this can require $O(N^2)$ time for a data set containing N points.

In the case of clustering methods, the first step is to use a clustering algorithm in order to determine the dense regions of the data set. In the second step, some measure of the fit of the data points to the different clusters is used in order to compute an outlier score for the data point. For example, in the case of a k-means clustering algorithm, the distance of the data point to the nearest centroid may be used as a measure of its anomalous behavior. One challenge with the use of many clustering algorithms is that they implicitly assume specific kinds of cluster shapes

depending upon the specific algorithm or distance function used within the clustering algorithm. Therefore, methods which divide the data into small regions in which the density can be estimated are very useful for scoring the sparsity of different regions in the data.

Density-based methods provide a high level of interpretability, when the sparse regions in the data can be presented in terms of combinations of the original attributes. For example, combinations of constraints on the original attributes can be presented as the specific criteria for particular data points being interpreted as outliers. Proximity-based methods for outlier detection are discussed in Chapter 4.

3.5   Information Theoretic Models

Many of the aforementioned models for outlier analysis use some form of data summarization method in terms of either generative probabilistic model parameters, clusters, or lower dimensional hyper-planes of projections. This provides a small summary of the data, the deviations from which are flagged as outliers. Information theoretic measures are broadly based on this principle. The idea is that outliers increase the minimum code length required to describe a data set. For example, consider the following two strings:

```
ABABABABABABABABABABABABABABABABAB
ABABACABABABABABABABABABABABABABAB
```

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

- A probabilistic model describes a data set in terms of generative model parameters, such as a mixture of gaussian distributions or a mixture of exponential power distributions [74].

- A clustering or density-based summarization model describes a data set in terms of cluster descriptions, histograms or other summarized representations, along with maximum error tolerances [233].

- A PCA model or spectral model describes the data in terms of lower dimensional subspaces of projection of multi-dimensional data or a condensed representation of a network [429].

- A frequent pattern mining method describes the data in terms of an underlying code book of frequent patterns. These are among the most common methods used for information-theoretic anomaly detection [34, 123, 410].

All these models represent the data approximately in terms of individual condensed components representing aggregate trends. In general, outliers increase the length of the description in terms of these condensed components to achieve the same level of approximation. For example, a data set with outliers will require a larger number of mixture parameters, clusters, PCA-based subspace dimensionality, or frequent patterns in order to achieve *the same level of approximation*. Correspondingly, in information theoretic methods, the key idea is to construct a *code book* in which to represent the data, and outliers are defined as points which removal results in the *largest decrease* in description length [123], or the most accurate summary representation in the same description length after removal [233]. The term "code book" is rather loosely defined in outlier analysis and refers to the condensed aggregate components of the data in terms of which the data is described. The actual construction of the coding is often heuristic, and an effective choice is key to the success of the approach. In general, the determination of the minimum length coding is a computationally intractable problem for a given data set, and therefore a variety of heuristic models (or code books) may be used for representation purposes [34, 123, 233, 410]. In many cases, these techniques can be related to conventional data summarization models for outlier analysis. In some cases, the coding is not explicitly constructed, but measures such as the entropy or Kolmogorov complexity are used as a surrogate in order to estimate the level of unevenness of a specific segment of the data [297, 259]. Segments with greater unevenness may be selectively explored to determine the outliers.

Conventional models look at this problem in a complementary way, by defining outliers as points which are expressed in the *least precise way* by (or deviations from) from a *fixed* model with a particular length. On the other hand, information theoretic models examine the *differential impact* of *removing* an outlier point from the data set on the tradeoff between coding length and representation accuracy. The two are clearly closely related. Since information theoretic methods largely differ from conventional models in terms of how the measure is defined, they often use similar methods as conventional techniques (eg. frequent pattern

mining [34, 410], histograms [233] or spectral methods [429]) in order to create the coding representation. Therefore, information theoretic methods will be discussed at various places in this book along with the chapter containing similar techniques or data types. Information theoretic methods can also be used for change detection in temporal data [96], by examining specific temporal segments of the data, and measuring the description length of these segments. The segments with the greatest change will typically have a larger description length.

3.6   High-Dimensional Outlier Detection

The high-dimensional case is particularly challenging for outlier detection. This is because, in high dimensionality, the data becomes sparse, and all pairs of data points become almost equidistant from one another [22, 215]. From a density perspective, all regions become almost equally sparse in full dimensionality. Therefore, it is no longer meaningful to talk in terms of extreme value deviations based on the distances in full dimensionality. The reason for this behavior is that many dimensions may be very noisy, and they may show similar pairwise behavior in terms of the addition of the dimension-specific distances. The sparsity behavior in high dimensionality makes all points look very similar to one another.

A salient observation is that the true outliers may only be discovered by examining the distribution of the data in a lower dimensional *local* subspace [4]. In such cases, *outliers are often hidden in the unusual local behavior of lower dimensional subspaces, and this deviant behavior is masked by full dimensional analysis.* Therefore, it may often be fruitful to explicitly search for the appropriate subspaces, where the outliers may be found. This approach is a generalization of both (full-dimensional) clustering and (full data) regression analysis. It combines local data pattern analysis with subspace analysis in order to mine the significant outliers. This can be a huge challenge, because the simultaneous discovery of relevant data localities and subspaces in high dimensionality can be computationally very difficult. Typically evolutionary heuristics such as genetic algorithms can be very useful in exploring the large number of underlying subspaces [4].

High-dimensional methods provide an interesting direction for intensional understanding of outlier analysis, when the subspaces are described in terms of the original attributes. In such cases, the output of the algorithms provide *specific combinations of attributes along with data locality*, which resulted in such data points being declared as outliers. This kind of interpretability is very useful, when a small number of interesting attributes need to be selected from a large number of possibil-

ities for outlier analysis. Methods for high dimensional outlier detection are discussed in Chapter 5.

4.   Meta-Algorithms for Outlier Analysis

In many data mining problems such as clustering and classification, a variety of meta-algorithms are used in order to improve the robustness of the underlying solutions. For example, in the case of the classification problem, a variety of ensemble methods such as bagging, boosting and stacking are used in order to improve the robustness of the classification [146]. Similarly, in the case of clustering, ensemble methods are often used in order to improve the quality of the clustering [20]. Therefore, it is natural to ask whether such meta-algorithms also exist for the outlier detection problem. The answer is in the affirmative, though the work on meta-algorithms for outlier detection is often quite scattered in the literature, and in comparison to other problems such as classification, not as well formalized. In some cases such as sequential ensembles, the corresponding techniques are often repeatedly used in the context of *specific* techniques, though are not formally recognized as general purpose meta-algorithms which can be used in order to improve outlier detection algorithms. The different meta-algorithms for outlier detection will be discussed in the following subsections.

There are two primary kinds of ensembles, which can be used in order to improve the quality of outlier detection algorithms:

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

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

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

Figure 1.6.   Sequential Ensemble Framework

4.1   Sequential Ensembles

In sequential-ensembles, one or more outlier detection algorithms are applied sequentially to either all or portions of the data. The core principle of the approach is that each application of the algorithms provides a better understanding of the data, so as to enable a more refined execution with either a modified algorithm or data set. Thus, depending upon the approach, either the data set or the algorithm may be changed in sequential executions. If desired, this approach can either be applied for a fixed number of times, or be used in order to converge to a more robust solution. The broad framework of a sequential-ensemble algorithm is provided in Figure 1.6.

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

convergence-based approach, as presented above. The broad principle of sequential ensembles is that a greater knowledge of data with successive algorithmic execution helps focus on techniques and portions of the data which can provide fresh insights.

Sequential ensembles have not been sufficiently explored in the outlier analysis literature as general purpose meta-algorithms. However, many *specific* techniques in the outlier literature use methods, which can be recognized as special cases of sequential ensembles. A classic example of this is the use of two-phase algorithms for building a model of the normal data. In the first-phase, an outlier detection algorithm is used in order to remove the obvious outliers. In the second phase, *a more robust* normal model is constructed after removing these obvious outliers. Thus, the outlier analysis in the second stage is much more refined and accurate. Such approaches are commonly used for cluster-based outlier analysis (for constructing more robust clusters in later stages) [55], or for more robust histogram construction and density estimation (see Chapter 4). However, most of these methods are presented in the outlier analysis literature as specific optimizations of *particular* algorithms, rather than as general meta-algorithms which can improve the effectiveness of an *arbitrary* outlier detection algorithm. There is significant scope for further research in the outlier analysis literature, by recognizing these methods as general-purpose ensembles, and using them to improve the effectiveness of outlier detection.

4.2   Independent Ensembles

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

The broad principle of independent ensembles is that different ways of looking at the same problem provides more robust results which are not dependent on specific artifacts of a particular algorithm or data set. Independent ensembles have been explored much more widely and formally in the outlier analysis literature, as compared to sequential ensembles. Independent ensembles are particularly popular for outlier analysis in high-dimensional data sets, because they enable the exploration of dif-

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

Figure 1.7.   Independent Ensemble Framework

ferent subspaces of the data in which different kinds of deviants may be found. These methods will be discussed in detail in Chapter 5.

Examples exist of both picking different algorithms and data sets, in order to combine the results from different executions. For example, the methods in [289, 310] sample subspaces from the underlying data in order to determine outliers from each of these executions independently. Then, the results from these different executions are combined in order to determine the outliers. The idea in these methods is that results from different subsets of sampled features may be bagged in order to provide more robust results. Some of the recent methods for subspace outlier ranking and outlier evaluation can be considered independent ensembles which combine the outliers discovered in different subspaces in order to provide more robust insights. These methods will be discussed in detail in Chapter 5.

5.   The Basic Data Types for Analysis

Most of our aforementioned discussion in the previous sections was focussed on multidimensional numerical data. Furthermore, it was assumed that the data records are independent of one another. However, in practice, the underlying data may be much more complex, both in terms of the kinds of attributes, and the relationships between different data records. Some examples of the different kinds of data, which may be encountered in real applications are discussed in this section.

5.1   Categorical, Text and Mixed Attributes

Many data sets in real applications may contain categorical attributes, which take on *discrete unordered* values. For example, demographic data may contain attributes such as race, gender, or the zip-code. Such attribute values are not ordered, and therefore require different data analysis techniques. Furthermore, the different kinds of attributes (numerical and categorical) may be mixed with one another. Many of the techniques for nearest neighbor and density-based classification can be extended to the case of such attributes, because the concept of proximity can be extended to such cases. The major challenge is to construct a distance function, which remains semantically meaningful for the case of discrete data.

Regression-based models can also be used in a limited way over discrete attribute values, when the number of possible values of an attribute is not too large. The typical methodology is to convert the discrete data to binary data by creating one attribute for each categorical value. Regression models such as principal component analysis may then be applied to this binary data set. Such methods can be more easily extended to text, where there is an inherent ordering among the frequencies of the words. In such cases, the correlations among occurrence of text words can be used in order to create linear-regression based models. In fact, some of the most successful models for text de-noising are based on latent semantic indexing (LSI), which is a form of linear regression analysis [133]. Other common methods for text and categorical data include clustering [26], proximity-based methods [515], probabilistic models [478], and methods based on frequent pattern mining [34, 208, 410]. Methods for outlier detection in categorical and mixed attribute data sets are discussed in Chapter 7.

5.2   When the Data Values have Dependencies

Most of the aforementioned discussion in this chapter is about the common multidimensional scenario, where it is assumed that the data records can be treated independently of one another. In practice, the different data values may be related to each other temporally, spatially, or through explicit network relationship links between the data items. The presence of such dependencies greatly changes the anomaly detection problem, because the nature of the dependencies plays a critical role in the anomaly detection process. In such cases, the *expected values* of data items are influenced by their contextual dependencies, and therefore outliers are defined on the basis of such contextually modeled deviations. When a single data item (eg. value from a time series) is

declared as an anomaly because of its *relationship* to its related data items, it is referred to as a *contextual* outlier or anomaly. Such outliers are also sometimes referred to as *conditional anomalies* [416]. For example, a sudden spike in a time series is a contextual anomaly, because it is very different from its expected value based on the values of its adjacent items. When a set of data items are declared anomalous *as a group of points*, it is referred to as a *collective* anomaly or outlier. For example, an unusual and rapid oscillation over time for a stock ticker value may be considered a collective anomaly, and it includes all the data items in the oscillation. Virtually, all anomalies in dependency-oriented data are *contextual* or *collective* anomalies, because they compute *expected* values based on relationships with adjacent data points in order to determine unexpected patterns. Furthermore, in such data sets, there are usually *multiple ways* to model anomalies, depending upon what an analyst might be looking for. Some examples of such data domains are presented in this section.

5.2.1   Times Series Data and Data Streams.   Time-series contains a set of values which are typically generated by continuous measurement over time. Therefore, the values in consecutive time-stamps do not change very significantly, or change in a smooth way. In such cases, *sudden changes* in the underlying data records, can be considered *anomalous events*. Therefore the discovery of anomalous points in time series, is usually closely related to the problem of anomalous *event detection*, in the form of either *contextual or collective anomalies over related time stamps* [9, 16, 260]. Typically such events are created by a sudden change in the underlying system, and may be of considerable interest to an analyst. For example, let us consider the following time-series of values, along with the corresponding time-stamps implicitly defined by the index of the data point.

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

The time-series is illustrated in Figure 1.8. It is evident that there is a sudden change in the data value at time-stamp 6 from 3 to 87. This corresponds to an outlier. Subsequently, the data stabilizes at this value, *and this becomes the new normal*. At time-stamp 12, the data value again dips to 3. *Even though this data value was encountered before*, it is still considered an outlier because of the sudden change in the consecutive data values. Thus, it is critical to understand that in this case, treating the data values independent of one another is not helpful for anomaly detection, because the data values are highly influenced by the adjacent values of the data points. Thus, the problem of outlier detection in time
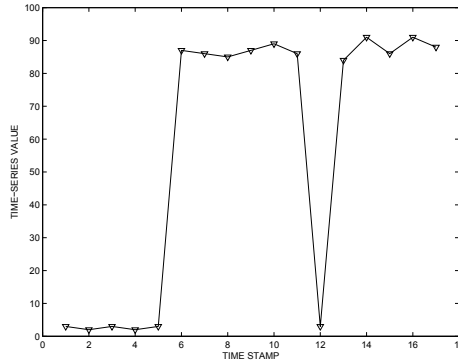
Figure 1.8. Example of Time Series

series data is highly related to the problem of change detection, because the normal models of data values are highly governed by adjacency in temporal ordering. When completely new data values are encountered, they are referred to as *novelties* [328, 329, 325], though outlier detection is relevant to any form of abrupt change, rather than only novelties, which are a specific kind of outliers.

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

- The values and trends in the data stream change slowly over time, a phenomenon which is referred to as *concept drift* [327, 10]. In such cases, the concept drift can only be detected by detailed analysis over a long period of time, and is not immediately obvious in many circumstances.

- The values and trends in the data stream change *abruptly*, so as to *immediately arouse suspicion that the underlying data generation mechanism has somehow changed fundamentally*.

The first scenario does not necessarily correspond to outliers, though the second scenario is more relevant to outlier detection. It is easy to see the parallels between the second scenario and the definition of outliers due to Hawkins [205], which was introduced at the very beginning of this chapter.

A common challenge in such scenarios is to perform the outlier detection *in real time*, as new data values are encountered. Many scenarios of change analysis and anomaly detection in temporal data are too tightly integrated to be treated separately. In such cases, solutions for one can

be used for the other, and vice-versa. On the other hand, the formulations of anomaly detection in temporal data are very diverse, not all of which are directly related to change detection. Usually online analysis is suited to change detection, whereas offline analysis may explore other unusual aspects of the data. Some examples are as follows:

- When the data is in the form of a time-series (eg, sensor data) large changes in *trends* may correspond to anomalies. These can be discovered as deviations from forecasted values using window-based analysis. In some cases, it may be desired to determine time-series subsequences of unusual shapes rather than change points in the data.

- For multidimensional data streams, changes in the aggregate distribution of the streaming data may correspond to unusual events. For example, network intrusion events may cause *aggregate* change points in a network stream. On the other hand, *individual* point novelties may or may not correspond to aggregate change points. The latter case is similar to multidimensional anomaly detection with an efficiency constraint for the streaming scenario.

Methods for anomaly detection in time series data and multidimensional data streams are discussed in detail in Chapter 8.

5.2.2   Discrete Sequences.   Many discrete sequence-based applications such as intrusion-detection and fraud-detection are clearly temporal in nature. This scenario can be considered a categorical or discrete analogue of time series data. Discrete sequences may not necessarily be temporal in nature, but may be based on their relative *placement* with respect to one another. An example is the case of biological data, where the sequences are defined on the basis of their relative placement.

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

Therefore, discrete sequences are analogous to continuous sequences, except that the different data representation typically requires different similarity functions, representation data structures, and more complex predictive techniques such as markovian models as opposed to autore-

gressive forecasting techniques. The problem formulations for the two cases are also similar at the high level. On the other hand, the specific techniques used for the two cases are very different. This is quite simply because numerical time series values are *ordered*, and therefore the values can be meaningfully compared across a continuous spectrum. However, two different discrete values cannot be meaningfully compared in a similar way. Value-continuity is lost in discrete data. Therefore, in order to maintain a coherent presentation, the case of discrete sequences has been addressed in a different chapter.

Discrete data is common in many real applications. Most biological sequences are discrete in nature, where each value in the sequence is drawn from a discrete set of possibilities. Similarly, host-based intrusion applications typically lead to discrete data, because numerous diagnostic events are drawn from a discrete set of instances [108]. Methods for anomaly detection in discrete sequences are discussed in Chapter 9.

5.2.3 Spatial Data. In spatial data, many non-spatial attributes (eg. temperature, pressure, image pixel color intensity) are measured at spatial locations. Unusual local changes in such values are reported as outliers. It should be pointed out that outlier detection in temporal data shares some resemblance to that in spatial data [433]. Both typically require the attribute of interest to exhibit a certain level of continuity. For example, consider the measurement of the temperature, where the measurement could be associated with a time-stamp and spatial coordinates. Just as it is expected that temperatures at consecutive time-stamps do not vary too much (temporal continuity), it is also expected that temperatures at spatially close locations do not vary too much (spatial continuity). In fact, such unusual spatial variations in sea surface temperatures and pressures [433] are used in order to identify significant and anomalous spatiotemporal events in the underlying data (eg. formation of cyclones). Spatiotemporal data is a generalization of both spatial and temporal data, and the methods used in either domain can often be generalized to such scenarios. Methods for finding outliers in spatial data are discussed in Chapter 10.

5.2.4 Network and Graph Data. In network or graph data, the data values may correspond to nodes in the network, whereas the relationships among the data values may correspond to the edges in the network. In such cases, outliers may be modeled in different ways depending upon the irregularity of *either* the nodes in terms of their relationships to other nodes, or the edges themselves. For example, a node which shows irregularity in its structure within its locality may be

(a) Node Outlier   (b) Edge Outlier
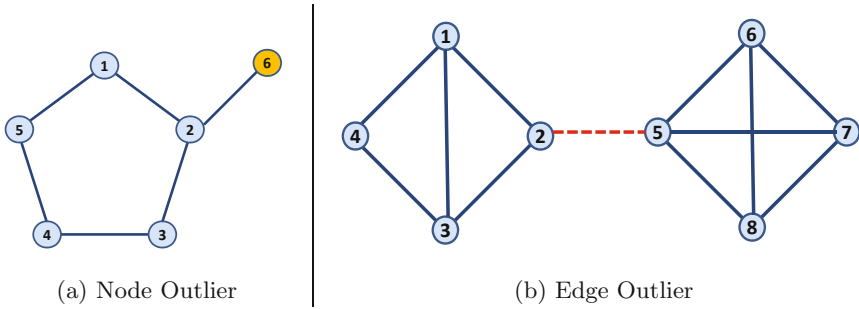
Figure 1.9.   Examples of Node and Edge Outliers

considered an outlier [33]. Similarly, an edge which connects disparate communities of nodes may be considered a *relationship* or *community outlier* [15, 180]. In Figure 1.9, two examples of outliers in networks are illustrated. The left example in Figure 1.9(a) contains an example of a node outlier, because the node 6 has an unusual locality structure which is significantly different from the other nodes. On the other hand, the edge (2, 5) in Figure 1.9(b) may be considered a relationship outlier or community outlier, because it connects two communities which are usually not connected to one another. Thus, in graph data, there is significantly more complexity and flexibility in terms of how outliers may be defined or modeled. In general, *the more complex the data, the more the analyst has to make prior inferences of what is considered normal for modeling purposes.*

It is also possible to combine different kinds of dependencies in a given data set. For example, graphs may be temporal in nature. In such a case, the data may have both structural and temporal dependencies, which change and also influence each other over time [15]. Therefore, outliers may be defined in terms of significant changes in the underlying network community or distance structure. Such models combine network analysis and change detection in order to detect structural and temporal outliers from the underlying data. A detailed discussion of methods for temporal and non-temporal outlier detection in graphs is provided in Chapter 11.

6.   Supervised Outlier Detection

In many scenarios, previous examples of outliers may be available. A subset of the data may be labeled as anomalies, whereas the remaining data may be considered normal. The corresponding methods are referred to as *supervised outlier detection*, because the labels are used in order to train a model which can determine *specific kinds* of anomalies.

Supervised methods are *generally* designed for anomaly detection, rather than noise removal, because they are based on the assumption that the labels represent what an analyst might specifically be looking for, rather than examples of what one might want to remove for data cleaning. Supervised models may often provide *very different* results from the unsupervised case, because they reflect an understanding of the underlying data. For example, let us consider the following time-series data:

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

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

Supervised anomaly detection finds numerous applications in fraud detection, intrusion detection, fault and disease diagnosis. In all these cases, the class of interest is very rare. It is this rarity that makes these instances outliers. Furthermore, it is usually much more important to correctly identify all the outliers, rather than the normal instances.

Supervised outlier detection is a (difficult) special case of the classification problem. The main characteristic of this problem is that the labels are extremely unbalanced in terms of relative presence [102]. The normal data is usually easy to collect and is therefore copiously available. On the other hand, outlier examples are very sparsely available in the data. In the classical machine learning literature, this problem is also referred to as the *rare class detection* problem. The imbalance in the class labels may often make the problem rather difficult to solve, because very few instances of the rare class may be available for modeling purposes. This may also make standard classification models prone to over-training, since the actual data *distinguishing* the rare class from the normal data is quite small. Furthermore, several variations of the classification problem also correspond to different levels of supervision:

- A limited number of instances of the positive (outlier) class may be available, whereas the "normal" examples may contain an unknown proportion of outliers [152]. This is referred to as the Positive-Unlabeled Classification (PUC) problem in machine learning. This variation is still quite similar to the fully supervised rare-class scenario, except that the classification model needs to be more cognizant of the contaminants in the negative (unlabeled) class. In cases, where the unlabeled training instances do not ac-

curately reflect the test instances, it may be desirable to discard the training instances for the unlabeled class, and treat it as a one-class problem, where only positive instances are available.

- Only instances of a subset of the normal and anomalous classes may be available, but some of the anomalous classes may be missing from the training data [325, 326, 445]. Such outliers are also referred to as *semi-supervised novelties*. This is distinct from *unsupervised novelties*, which tracks the formation of new clusters and trends in the data [26, 503, 515]. For example, in a bio-terrorist attack modeling scenario, no examples of the attack may be available, whereas copious examples of normal behavior and other kinds of more common anomalies may be available. This variation is also a semi-supervised scenario for learning, though it is quite similar to the unsupervised version of the problem. A more interesting case is one in which labeled examples of all normal and some anomalous classes are available, though the labels for the anomalous classes are not exhaustive. Such situations are quite common in scenarios such as intrusion detection, where some intrusions may be known, but other intrusions are continually created over time.

- Supervised outlier detection is closely related to active learning, in which human feedback is utilized in order to identify relevant outlier examples. This is because such methods do create models distinguishing between positive and negative examples of outliers, even when the example identification process is executed in parallel with the classification [360]. This process is also referred to as *Active Learning*.

All these different variants require careful design of the underlying classification algorithms. For example, cost-sensitive variations of standard machine learning algorithms can be used in order to make accurate predictions of anomalies in the data [151]. In such variations, the classifier is tuned, so that errors in classification of the anomalous class are penalized more heavily than the errors in classification of the majority class. The idea is that it is better to predict a negative class as an anomaly (false positive), rather than miss a true outlier (false negative). The different choices on costs may lead to different tradeoffs between false positives and false negatives. This tradeoff is characterized by either a *Precision-Recall (PR)* curve, or a *Receiver Operating Characteristics (ROC)* curve. These two kinds of curves are intimately related to one another. The issue of outlier evaluation will be discussed in the next section. Supervised methods for anomaly detection are discussed in greater detail in Chapter 6.
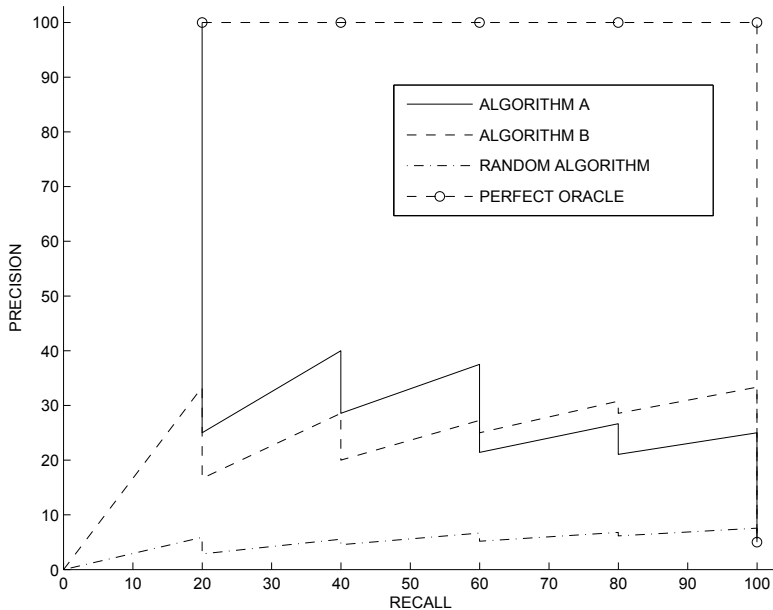
Figure 1.10. Precision-Recall Curves

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

Table 1.1. Rank of ground-truth outliers can be used to construct Precision-Recall curves

7. Outlier Evaluation Techniques

A key question arises as to how the effectiveness of an outlier detection algorithm should be evaluated. Unfortunately, this is often a difficult task, because outliers, by definition, are rare. This means that the ground-truth about which data points are outliers, is often not available. This is especially true for unsupervised algorithms, because if the ground-truth were indeed available, it could have been used to create a more effective supervised algorithm. In the unsupervised scenario (without ground-truth), it is often the case, that no realistic quantitative methods can be used in order to judge the effectiveness of the underlying algorithms in a rigorous way. Therefore, much of the research literature uses case studies in order to provide an intuitive and qualita-
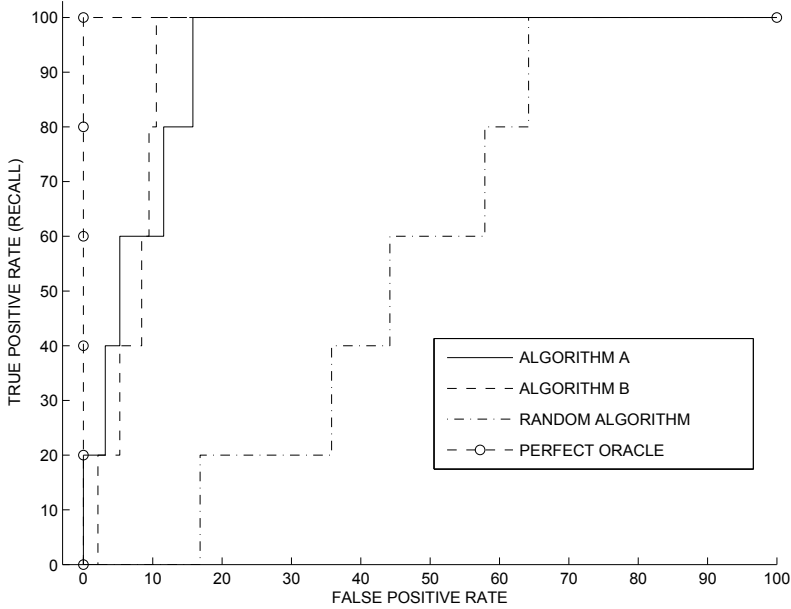
Figure 1.11.   Receiver Operating Characteristic Curves

tive evaluation of the underlying outliers in unsupervised scenarios. In some cases, the data sets may be adapted from imbalanced classification problems, and the rare labels may be used as surrogates for the ground truth outliers.

Nevertheless, many scenarios do exist, in which ground-truth is available. In most supervised algorithms, ground-truth is available, a part of which can be used in order to perform the supervision, and the remaining can be used for evaluation. Even in unsupervised scenarios, the ground-truth often becomes available after a period of time, even though it may not have been available at the time of outlier analysis. Therefore, a natural question arises as to how the ground-truth can be used to evaluate effectiveness. Most outlier detection algorithms output an outlier score, and a threshold on this score is used in order to declare data points as outliers. If the threshold is picked too restrictively in order to minimize the number of declared outliers, then the algorithm will miss true outlier points (false negatives). On the other hand, if the algorithm declares too many data points as outliers, then it will lead to too many false positives. This tradeoff can be measured in terms of *precision* and *recall*, which is commonly used for measuring the effectiveness of set-based retrieval.

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

$$Precision(t) = 100 * \frac{|S(t) \cap G|}{|S(t)|}$$

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

$$Recall(t) = 100 * \frac{|S(t) \cap G|}{|G|}$$

By varying the parameter t, it is possible to plot a curve between the precision and the recall. This is referred to as the *Precision-Recall* curve. This curve is *not necessarily* monotonic. On the other hand, for more effective algorithms, high values of precision may often correspond to low values of recall and vice-versa. The precision-recall (PR) curve can also be generated by using thresholds on the *rank* of the data points, when sorted by outlier score. In the absence of ties in the outlier scores, a rank-based and score-based PR curve would be identical.

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

$$TPR(t) = Recall(t)$$

$$FPR(t) = 100 * \frac{|S(t) - G|}{|D - G|}$$

Note that the end points of the ROC curve are always at $(0, 0)$ and $(100, 100)$, and a random method is expected to exhibit performance along the diagonal line connecting these points. The *lift* obtained above this diagonal line provides an idea of the accuracy of the approach. The ROC curve is simply a different way to characterize the tradeoffs than the precision-recall curve, though the two can be derived from one another. The ROC curve has the advantage of being monotonic, and more easily

interpretable in terms of its lift characteristics. On the other hand, the tradeoffs are sometimes more clearly understood at a detailed level with the use of a PR curve.

In order to illustrate the insights gained from these different graphical representations, consider an example of a data set with 100 points, from which five points are outliers. Two algorithms A and B are applied to this data set, which rank all data points from 1 to 100, with lower rank representing greater propensity to be an outlier. Thus, the precision and recall values can be generated by determining the ranks of the 5 ground truth outlier points. In Table 1.1, some hypothetical ranks for the 5 ground truth outliers have been illustrated for the different algorithms. In addition, the ground truth ranks for a random algorithm have been indicated. The random algorithm which outputs a random score for outlier detection of a given data point. Similarly, the ranks for a "perfect oracle" algorithm which ranks the correct top 5 points as outlier have also been illustrated in the table. The corresponding PR curve for this hypothetical output of outlier scores are illustrated in Figure 1.10. Other than the oracle algorithm, all the tradeoff curves are non-monotonic. This is because the discovery of a new outlier at any particular relaxation in rank threshold results in a spike in the precision, which becomes less pronounced at higher values of the recall. The corresponding ROC curve is illustrated in Figure 1.11. Unlike the PR curve, this curve is clearly monotonic.

What do these curves really tell us? For cases in which one curve strictly dominates another, it is clear that the algorithm for the former curve is superior. For example, it is immediately evident that the oracle algorithm is superior to all algorithms, and the random algorithm is inferior to all the other algorithms. On the other hand, the algorithms A and B show domination at different parts of the ROC curve. In such cases, it is hard to say that one algorithm is strictly superior. From Table 1.1, it is clear that Algorithm A, ranks three of the correct ground truth outliers very highly, but but the remaining two outliers are ranked poorly. In the case of Algorithm B, the highest ranked outliers are not as well ranked as the case of Algorithm A, though all five outliers are determined much earlier in terms of rank threshold. Correspondingly, Algorithm A dominates on the earlier part of the PR curve, whereas Algorithm B dominates on the later part. Some practitioners use the area under the ROC curve as a proxy for the overall effectiveness of the algorithm, though such a measure should be used very carefully, because all parts of the ROC curve may not be equally important for different applications.

8.   Conclusions and Summary

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

9.   Bibliographic Survey

A number of books and surveys have been written on the problem of outlier analysis. The classic books [58, 205, 387] in this area have mostly been written from the perspective of the statistics community. Most of these books were written before the wider adoption of database technology, and are therefore not written from a computational perspective. More recently, this problem has been studied quite extensively by the computer science community. These works consider practical aspects of outlier detection, corresponding to the cases, where the data may be very large, or may have very high dimensionality. Numerous surveys have also been written, which discuss the concept of outliers from different points of view, methodologies or data types [30, 62, 107, 108, 325, 326]. Among these, the survey by Chandola et al [107], is the most recent, and arguably the most comprehensive. It is an excellent review, which covers the work on outlier detection quite broadly from the perspective of multiple communities.

The issue of distinguishing between spurious abnormalities (or noise) and true outliers has also been discussed in [9], where the challenges of finding true anomalies in time series have been discussed. The Z-value test discussed in section 2 is used commonly in the statistical literature, and many variants for limited sample sizes such as the Grubb's test [188] and t-value test are also available. While this test makes the normal distribution assumption for large data sets, it has been used

fairly extensively as a good heuristic even for data distributions which do not satisfy the normal distribution assumption.

The basic models discussed in this chapter have also been researched extensively, and have been studied widely in the literature. Details of these methods (along with the corresponding bibliographic notes) will be provided in later chapters. Here only the most important works in each area are covered. The key statistical techniques on regression-based modeling are covered in [387]. The basic EM-algorithm for unsupervised modeling of data sets was first proposed in [135]. The non-parametric technique of principal component analysis (PCA) discussed in section 2 is described well in [244]. The core technique of PCA was extended to text (with some minor variations) as Latent Semantic Indexing [133]. A variety of distance-based methods for outlier detection are proposed in [261, 381, 441], and density-based methods for outlier detection were proposed in [78]. Methods for interpreting distance-based outliers were first proposed in [262]. A variety of information theoretic methods for outlier detection are discussed in [34, 45, 74, 96, 123, 211, 212, 297, 410].

The issues of poor behavior of high dimensional applications such as clustering and nearest neighbor search have been observed in several prior works in the literature [5, 7, 8, 22, 215]. The problem of high-dimensional outlier detection was first proposed in [4]. Subspace approaches for outlier detection were proposed in this paper, and a number of other recent methods have followed a similar line of work [256, 273, 337–339, 341, 498–501, 513].

Outliers have been studied extensively in the context of different data domains. While numeric data is the most commonly studied case, numerous methods have also been proposed for categorical and mixed data [30, 478]. Methods for unsupervised outlier detection in text corpora are proposed in [197]. The problem of detecting outliers with dependencies has also been studied extensively in the literature. Methods for detecting outliers and changes in time series and streams were proposed in [9, 15, 16, 26, 257–260]. Novelty detection [325] is an area which is closely related to outlier analysis, and it is often studied in the context of supervised models, where novel classes from a data stream are detected in real time [328, 329], with the use of learning methods. However, novelty detection is also studied often in the unsupervised scenario, particularly in the context of *first story detection* in topic detection and tracking in text streams [515]. Spatial outliers [3, 268, 317, 401–404] are closely related to the problem of finding outliers in temporal data, since such data also shows spatial continuity, just as temporal data shows temporal continuity. Some forms of spatial data also have a temporal component

to them, which requires the determination of spatiotemporal outliers [113, 114].

Outlier detection in discrete sequences is related to the problem of temporal outlier detection in continuous sequences. For discrete sequences, an excellent survey may be found in [108]. Methods for finding node outliers with unusual neighborhood behavior in graphs were proposed in [33], and techniques for finding relationship outliers, subgraph outliers and community outliers were proposed in [15, 180, 349, 378]. The primary ideas in all these methods is that outlier regions in a network are caused by unusual relationships in the form of edges, subgraphs, and communities. The temporal analysis of graph streams in the context of significant community evolution was studied in [17, 192, 429]. The problem of discovering significant structural change in temporal networks in the form of distance changes was studied in [193].

Recently, some *meta-algorithms for outlier detection* have been designed. The core-idea of this approach is that multiple methods for outlier detection will provide different results, and these results can be combined in order to provide more robust results. This approach lies at the core of *ensemble-based methods* [289, 310, 271]. In the case of sequential ensembles, most of the currently available techniques are ad-hoc, and apply to specific algorithms. These techniques are often not recognized as general-purpose meta-algorithms, which can be used in order to improve the effectiveness of *any* arbitrary outlier detection algorithm, though the interests in this area have increased recently. Independent ensemble algorithms are based on the idea that multiple ways of solving the same problem are likely to lead to more robust results. For example, if two different methods find the same data point as an outlier, this is a more robust indicate of outlierness, since it does not result from a particular overfitting of the specific algorithm. The work in [289] designs methods for using different subsets of features in outlier detection methods, and combining them in order to provide more effective results. The work in [337–339] shows how to combine the scores from different subspaces found by outlier detection algorithms in order to provide a unified and more robust result. The work in [271] also shows how outlier scores of different algorithms can be best interpreted and unified into more robust outputs.

The supervised version of the outlier detection problem has been studied extensively in the form of *rare class detection*. For the supervised case, readers are referred to a general book on classification [146], since this problem is essentially a cost-sensitive variation [102, 151] on the standard classification problem, in which the class distributions are very imbalanced. In particular, the readers are referred to [102, 151] for a

thorough discussion on the foundations of cost-sensitive learning from imbalanced data sets. A number of methods for classification from positive and unlabeled data are discussed in [152], and a good review of the previous work in this area may also be found from the references in this paper. The work in [360, 512, 513] first showed how human supervision could be used to significantly improve the effectiveness of outlier detection. Finally, the *semi-supervised* scenario of novelty detection has been discussed extensively in [325, 326, 445].

Evaluation methods for outlier analysis are essentially identical to the techniques used in information retrieval for understanding precision-recall tradeoffs, or in classification for ROC curve analysis. A detailed discussion of the proper construction of such curves may be found in [159]. While the ROC and PR curves are the traditional methods for outlier evaluation, it has recently been noted [337] that these methods may not necessarily provide all the insights needed for different kinds of analysis. Therefore, the work in [337] has proposed a coefficient, which was based on the Spearman correlation between the best possible ranking and the ranking determined by the algorithm. The work in [395] provides further ways of examining the ranks of outlier scores, which also provides insights into the effectiveness of outlier ensembles. Other visual methods of evaluating outliers include the LOCI plot [356] (discussed in Chapter 4), and the ELKI [2] software, which shows the contrasts in outlier scores in the form of histograms and bubble plots.

10.   Exercises

1. Which of the following points from each of the following sets of points below is an outlier? Why?

   - (1-dimensional) { 1, 3, 2, 1, 3, 2, 75, 1, 3, 2, 2, 1, 2, 3, 2, 1 }
   - (1-dimensional) { 1, 2, 3, 4, 2, 19, 9, 21, 20, 22 }
   - (2-dimensional) { (1, 9), (2, 9), (3, 9), (10, 10), (10, 3), (9, 1), (10, 2) }

2. Use MATLAB or any other mathematical software to create a histogram of the data distribution along each of the dimensions in the different cases of Exercise 1. Can you see the outliers visually? Which ones? In which case are the outliers not clear and why?

3. For the 2-dimensional case of Exercise 1, plot the data points on a 2-dimensional plane. Can you see the outliers visually? Which ones?

4. Apply the Z-value test to each of the cases in Exercise 1. For the 2-dimensional case, apply the Z-value test to the individual dimensions. Do you discover the correct outliers?

5. For the 2-dimensional case in Exercise 1, construct the function $f(x_1, x_2) = |x_1 - x_2|$. Apply the Z-value test to $f(x_1, x_2)$ over each of the data points. Do you obtain the correct outliers, as suggested by your visual analysis in Exercise 3? Why?

6. Determine the nearest neighbor of each data point for the cases in Exercise 1. Which data points have the largest value of the nearest neighbor distance? Are they the correct outliers?

7. Apply a k-means clustering algorithm to each of the cases in Exercise 1, while setting $k = 2$. Which data points lie furthest from the two means thus found? Are these the correct outliers?

8 Consider the following time-series:

   ▪ 1, 2, 3, 3, 2, 1, 73, 1, 2, 3, 5

   ▪ 1, 2, 3, 4, 3, 2, 1, 3, 73, 72, 74, 73, 74, 1, 2, 3, 4, 2

   ▪ 1, 2, 3, 5, 6, 19, 11, 15, 17, 2, 17, 19 , 17, 18

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

9. Consider the undirected network $G = (N, A)$ of 8 nodes in N indexed from 1 through 8. Let the edge set A be { (1, 2), (1, 3), (1, 4), (1, 5), (1, 6), (1, 7), (1, 8) }. Draw the network on paper to visualize it. Is there any node, which you would consider an outlier? Why?

   ▪ Now delete the edge $(1, 7)$. Does this change the set of nodes you would consider outliers? Why?

10. Consider the undirected network $G = (N, A)$ of 8 nodes in N indexed from 1 through 8. Let the edge set A be { (1, 2), (1, 3), (1, 4), (2, 3), (2, 4), (5, 7), (4, 7), (5, 6), (6, 8), (5, 8), (6, 7) }. Draw the network on paper to visualize it. Is there any edge, which you would consider an outlier? Why?

11. Consider three algorithms A, B and C, which are run on a data set with 100 points and 5 outliers. The rank of the outliers by score for the three algorithms are as follows:

A: 1, 3, 5, 8, 11
B: 2, 5, 6, 7, 9
C: 2, 4, 6, 10, 13
Draw the PR curves for each of the algorithms. Would you consider
any of the algorithms strictly superior to any of the others? Why?

Chapter 2

PROBABILISTIC AND STATISTICAL MODELS FOR OUTLIER DETECTION

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

1. Introduction

The oldest methods for outlier detection are rooted in probabilistic and statistical models, and date back to the nineteenth century [149]. The earliest methods were proposed well before the advent and popularization of computer technology. Therefore, these methods were designed without much focus on practical issues such as data representation or computational efficiency. Nevertheless, the underlying mathematical models are extremely useful, and have eventually been adapted to a variety of computational scenarios.

A popular form of statistical modeling in outlier analysis is that of detecting *extreme univariate values*. In such cases, it is desirable to determine data values at the tails of a univariate distribution, along with a corresponding level of statistical significance. This would seem a rather restrictive case, since most multidimensional outliers do not correspond to extremes in data values. Rather, outliers are typically defined by the *relative* positions of the data values with respect to each other. While extreme univariate values correspond to a very specific kind of outliers, they have numerous applications beyond the univariate case. This is because virtually all outlier detection algorithms perform some kind of numerical scoring, in order to measure the anomalousness of data points. In some cases, the scores may come with a confidence value or probability, though this capability is often not directly built

into outlier analysis algorithms. Therefore, the final step in all these algorithms is to determine the extreme values from these scores. The determination of statistically extreme values helps in the conversion of outlier scores into binary labels.

Some examples of outlier scoring mechanisms, which are returned by different classes of algorithms, are as follows:

- In probabilistic modeling, the likelihood fit of a data point to the model is the outlier score.

- In proximity-based modeling, the k-nearest neighbor distance, distance to closest cluster centroids, or local density value is the outlier score.

- In linear modeling, the residual distance of a data point to a lower-dimensional representation of the data is the outlier score.

- In temporal modeling, a function of the distance from previous data points (or the deviation from a forecasted value) is used to create the outlier score.

Thus, even when extreme value modeling cannot be performed on the original data, the ability to determine the extreme values effectively from a set of outlier scores forms the cornerstone of all outlier detection algorithms as a final step. Some recent work has been devoted exclusively to the problem of determining such extreme values [179] from outlier scores, by converting these scores into probabilities. Therefore, the issue of extreme value modeling will be studied extensively in this chapter. Extreme value modeling can also be easily extended to multivariate data, and will be discussed in this chapter.

It is also possible to use probabilistic modeling for finding general outliers beyond extreme values. Mixture models can be considered probabilistic versions of clustering algorithms, and can therefore be used for outlier analysis. A significant advantage of these methods is that they are fairly easy to generalize to different data formats, or even heterogenous data attributes, once a generative model for the data has been defined. Most probabilistic models assume a particular form to the underlying distribution, according to which the data is modeled. Subsequently, the parameters of this model are learned, typically with a maximum-likelihood estimation technique [135]. This model then becomes a *generative* model for the data, and the probability of a particular data point being generated can be computed from this model. Data points which have an unusually low probability of being generated from the model are returned as outliers.

This chapter is organized as follows. The next section discusses statistical models for extreme value analysis. Methods for extreme-value analysis in multivariate data are discussed in section 3. Section 4 discusses methods for probabilistic modeling of outliers. Section 5 discusses the limitations of probabilistic models for outlier analysis. Section 6 presents the conclusions and summary.

2.   Statistical Methods for Extreme Value Analysis

In this section, we will present probabilistic and statistical methods for extreme value analysis in univariate data distributions. The extreme values in a probability distribution are collectively referred to as the distribution *tail*. Statistical methods for extreme value analysis quantify the probabilities in the tails of distributions. Clearly, a very low probability value of a tail indicates that a data value inside it should be considered anomalous. A number of tail inequalities bound these probabilities.

2.1   Probabilistic Tail Inequalities

Tail inequalities can be used in order to bound the probability that a value in the tail of a probability distribution should be considered anomalous. The most fundamental tail inequality is the *Markov inequality*, which is defined for distributions, which take on only non-negative values. Let X be a random variable, with probability distribution $f_X(x)$, a mean of $E[X]$, and a variance of $Var[X]$.

THEOREM 2.1 (MARKOV INEQUALITY) *Let X be a random variable, which takes on only non-negative random values. Then, for any constant α satisfying $E[X] < \alpha$, the following is true:*

$$P(X > \alpha) \leq E[X]/\alpha \qquad (2.1)$$

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

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

The first inequality follows from the non-negativity of x, and the second follows from the fact that the integral is only defined over the cases

where $x > \alpha$. Now, the term on the right-hand side of the last line is exactly equal to $\alpha \cdot P(X > \alpha)$. Therefore, the following is true:

$$E[X] \geq \alpha \cdot P(X > \alpha) \tag{2.2}$$

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

$\blacksquare$

The Markov inequality is defined only for probability distributions of non-negative values, and provides a bound only on the upper tail. In practice, it is often desired to bound both tails of arbitrary distributions. Consider the case where X is an arbitrary random variable, which is not necessarily non-negative. Tail bounds may be derived in a symmetric way with the *Chebychev inequality*. The Chebychev inequality is a direct application of the Markov inequality to a non-negative derivative (square deviation-based) distribution of X.

THEOREM 2.2 (CHEBYCHEV INEQUALITY) *Let X be an arbitrary random variable. Then, for any constant α, the following is true:*

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

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

$\blacksquare$

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

The Markov and Chebychev inequalities are relatively weak inequalities, and often do not provide tight enough bounds to be useful in many

practical scenarios. This is because these inequalities do not assume any specific shape of the probability distribution, or any specific form of the random variable X. Many practical scenarios can however be captured with the use of specific forms of the random variable. In such cases, much tighter bounds on tail distributions are possible. A particular case is one in which a random variable X may be expressed as a sum of other independent bounded random variables.

2.1.1 Sum of Bounded Random Variables. Many practical observations, which are defined in the form of *aggregates* can be expressed as a sum of bounded random variables. Some examples of practical scenarios in which such data could arise are as follows:

EXAMPLE 2.3 (SPORTS STATISTICS) *The NBA draft teams have access to college basketball statistics for the different candidate players. For each player and each game, a set of quantitative values describe their various scoring statistics over different games. For example, these quantitative values could correspond to the number of dunks, assists, rebounds, etc. For a particular statistic, the aggregate performance of any player can be expressed as the sum of their statistics over N different games:*

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

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

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

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

EXAMPLE 2.4 (GROCERY SHOPPING) *A grocery store keeps track of the number of customers (from its frequent purchaser program), which have frequented the store on a particular day. The long term probability of any customer i attending the store on a given day is known to be p_i. The behavior of the different customers is also known to be independent of one another. For a given day, determine the probability that the store receives more than η (frequent purchase program) customers.*

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

EXAMPLE 2.5 (MANUFACTURING QUALITY CONTROL) *A company uses a manufacturing assembly line to produce a product, which may have faults in it with a pre-defined (low) probability p. The quality control process periodically samples N products from the assembly line, and examines them closely to count the number of products with defects. For a given count of faulty products, determine the probability that the assembly line is behaving anomalously.*

In the last example, the sample size N is typically relatively large. In such cases, it is possible to use the Central Limit Theorem to approximate the sample distribution as a normal distribution. This provides the tightest possible bound. The different kinds of bounds and approximations will be addressed in this section.

The Chernoff bounds and the Hoeffding inequality will be discussed first. Since the expressions for the lower tail and upper tails are slightly different, they will be addressed separately. The lower tail Chernoff bound is introduced below.

THEOREM 2.6 (LOWER TAIL CHERNOFF BOUND) *Let X be random variable, which can be expressed as the sum of N independent binary (Bernoulli) random variables, each of which takes on the value of 1 with probability p_i.*

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

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

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

where e is the base of the natural logarithm.

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

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

The *unknown* parameter $t > 0$ is introduced in order to create a parameterized bound. The lower tail inequality of X is converted into an upper tail inequality on $e^{-t \cdot X}$. This can be bounded by the Markov inequality,

and it provides a bound which is a function of t. This function of t can be optimized, in order to obtain the tightest possible bound. By using the Markov Inequality on the exponentiated form, the following can be derived:

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

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

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

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

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

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

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

The expression on the right is true for any value of $t > 0$. It is desired to pick a value t which provides the *tightest possible* bound. Such a value of t may be obtained by using the standard optimization process of using the derivative of the expression with respect to t. It can be shown by working out the details of this optimization process that the optimum value of $t = t^*$ is as follows:

$$t^* = \ln(1/(1 - \delta)) \tag{2.7}$$

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

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

∎

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

THEOREM 2.7 (UPPER TAIL CHERNOFF BOUND) *Let X be random variable, which can be expressed as the sum of N independent binary (Bernoulli) random variables, each of which takes on the value of 1 with probability p_i.*

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

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

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

where e is the base of the natural logarithm.

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

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

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

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

■

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

THEOREM 2.8 (HOEFFDING INEQUALITY) *Let X be random variable, which can be expressed as the sum of N independent random variables, each of which is bounded in the range $[l_i, u_i]$.*

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

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

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

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

Proof Sketch: The proof for the upper tail will be briefly described here. The proof of the lower tail inequality is identical. For an unknown parameter t, the following is true:

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

The Markov inequality can be used to show that the right hand probability is at most $E[e^{(X - E[X])}] \cdot e^{-t \cdot \theta}$. The expression within $E[e^{(X - E[X])}]$ can be expanded in terms of the individual components X_i. Since, the expectation of the product is equal to the product of the expectations of independent random variables, the following can be shown:

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

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

Therefore, the following is true:

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

This inequality holds for any non-negative value of t. Therefore, in order to find the tightest bound, the value of t, which minimizes the RHS of the above equation needs to be determined. The optimal value of $t = t^*$ can be shown to be:

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

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

■

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

Result	Scenario	Strength
Chebychev	Any Random Variable	Weak
Markov	Non-negative Random Variable	Weak
Hoeffding	Sum of Indep. Bounded Random Variables.	Strong (Exp.)
Chernoff	Sum of Independent Bernoulli Variables	Strong (Exp.)
CLT	Sum of **many** iid variables	Almost Exact
Gen. CLT	Sum of **many** independent bounded variables	Almost Exact

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

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

THEOREM 2.9 (CENTRAL LIMIT THEOREM) *The sum of a large number N of independent and identically distributed random variables with mean μ and standard deviation σ converges to a normal distribution with mean $\mu \cdot N$ and standard deviation $\sigma \cdot \sqrt{N}$.*

A more generalized form of the CLT can also be applied to sums of independent variables (not necessarily identical), in which the variables are sufficiently bounded in terms of underlying moment measures. An example of such a generalization of the CLT is the *Lyapunov* CLT. A discussion of this generalized version is omitted, since it is beyond the scope of this book. Interested readers are referred to [70]. In this case, if the mean and variance of the ith variable is μ_i and σ_i^2, then the mean and variance of the corresponding normal distribution are the sums of these values. In the next section, the common use of the normal distribution assumption for confidence testing will be discussed.

2.2   Statistical Tail Confidence Tests

The most basic statistical tests assume a normal distribution for the underlying data values. Normal distributions are very common in measurements in many real domains. This is true not just for variables which are expressed as sums of samples (as discussed in the last section), but many variables which are generated by a variety of different processes. The density function $f_X(x)$ for the normal distribution with mean μ and

standard deviation σ is defined as follows:

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

A *standard* normal distribution is one in which the mean is 0, and the standard deviation σ is 1. In some cases, it is appropriate to assume that the mean μ and standard deviation σ of the normal distribution are known. This is the case, when a very large number of samples of the data are available, or specific domain knowledge is available about the generating process. In that case, the Z-number z_i of an observed value x_i can be computed as follows:

$$z_i = (x_i - \mu)/\sigma \tag{2.17}$$

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

2.2.1   *t*-value test.   The aforementioned discussion was based on the assumption that the mean and standard deviation of the distribution are either *known*, because of *domain knowledge*, or can be estimated very accurately from a *large number of samples*. However, in practice, little domain knowledge of the data may be available, and the available data sets may be small. For example, for a sample with 20 data points, it is much harder to model the mean and standard deviations accurately on the basis of sample statistics. How do we accurately perform statistical significance tests in such cases?

The *student's t-distribution* provides an effective way to model anomalies in such scenarios. This distribution is defined by a parameter known as the *number of degrees of freedom ν*, which is closely defined by the available sample size. The t-distribution approximates the normal distribution extremely well for larger degrees of freedom (> 1000), and converges to the normal distribution in the limit where it goes to ∞.
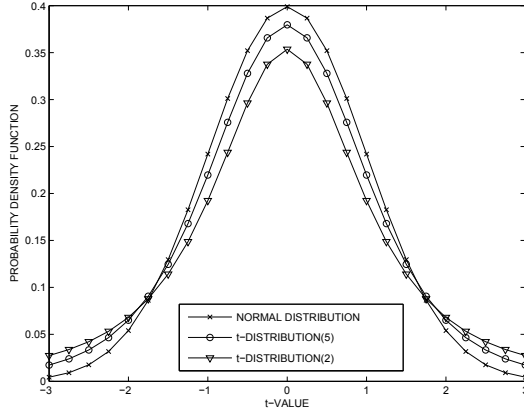
Figure 2.1.   The *t*-distributions for different numbers of degrees of freedom (corresponding to different sample sizes)

For fewer degrees of freedom (or sample size), the *t*-distribution has a similar bell-shaped curve as the normal distribution, except that it has heavier tails. This is quite intuitive, because the heavier tail accounts for the loss in statistical significance from the inability to accurately estimate the parameters of the underlying normal distribution from fewer samples.

The *t*-distribution is expressed as a function of *several independent identically-distributed standard normal distributions*. It has a single parameter ν, which corresponds to the number of *degrees of freedom*. This regulates the *number* of such normal distributions, in terms of which it is expressed. The parameter ν is set to $N - 1$, where N is the total number of available samples. Let $U_0 \ldots U_\nu$ be $\nu + 1$, independent and identically distributed normal distributions with mean 0 and a standard deviation of 1. Then, the *t*-distribution is defined as follows:

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

The intuition for using the *t*-distribution is that the denominator now explicitly models the randomness of estimating the standard deviation of the underlying normal distribution with the use of only a *small* number of independent samples. The term $\sum_{i=1}^{\nu} U_i^2$ in the denominator is a χ^2 distribution with parameter ν, and the entire (scaled) denominator converges to 1, when $\nu \Rightarrow \infty$. Therefore, in the limiting case, when a large number of samples are available, the randomness contributed by the denominator disappears, and the *t*-distribution converges to the normal

distribution. For smaller values of ν (or sample sizes), this distribution has a heavier tail, and may be used in order to provide the corresponding tail probability. Examples of the t-distribution for different values of ν are provided in Figure 2.1. It is easy to see, that t-distributions with fewer degrees of freedom have heavier tails.

The process of extreme value detection with a small number of samples $x_1 \ldots x_N$ proceeds as follows. First, the mean and standard deviation of the sample are estimated. This is then used to compute the t-value of each data point directly from the sample. The t-value is computed in an identical way as the Z-value. The tail probability of each data point is computed from the cumulative density function of the t-distribution with $(N - 1)$-degrees of freedom. As in the case of the normal distribution, standardized tables are available for this purpose. From a practical perspective, if more than 1000 samples are available, then the t-distribution (with at least 1000 degrees of freedom) is so close to the normal distribution, that it is possible to use the normal distribution as a very good approximation.

2.2.2 Sum of Squares of Deviations. A common situation which is encountered in the context of multidimensional data is the case, where the deviations along a set of independent orthogonal directions are aggregated together. Each of these deviations are typically modeled as a Z-value from an independent and identically distributed standard normal distribution. The aggregate deviation measure is then computed as the sum of the squares of these values. For a d-dimensional data set, this is a χ^2-distribution with d degrees of freedom. A χ^2-distribution with d degrees of freedom is defined as sum of the squares of d independent standard normal random variables. In other words, consider the variable V, which is expressed as the square sum of independent and identically distributed standard normal random variables $Z_i \sim N(0, 1)$:

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

Then, V is a random variable drawn from a χ^2 distribution with d degrees of freedom.

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

While a detailed discussion of the characteristics of the χ^2-distribution is skipped here, its cumulative distribution is not available in closed form, but it needs to computationally evaluated. From a practical standpoint, cumulative probability tables are typically available for modeling purposes. The cumulative probability tables of the χ^2-distribution can

then be used in order to determine the probabilistic level of significance for that aggregate deviation value. This approach is particularly useful when the deviations are modeled to be statistically independent of one another. As we will see in Chapter 3, such situations could arise in models such as principal component analysis, where the errors along the different components are often modeled as independent normal random variables.

3.   Extreme Value Analysis in Multivariate Data

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

While the methods discussed in this section are effective in finding outliers at the *outer boundaries* of a data space, they are not good at finding outliers within the inner regions of the data space. Such methods can effectively find outliers for the case illustrated in Figure 2.5, but not the outlier *A* illustrated in Figure 2.7. Nevertheless, the determination of such outliers can be useful in many specialized scenarios. For example, in cases where multiple deviation values may be associated with records, multivariate extreme value analysis may be useful. Consider a weather application in which multiple attributes such as temperature and pressure are measured at different spatial locations, and the local spatial deviations from the expected values are computed as an intermediate step. These deviations from expected values on different attributes may need to be transformed into a single meaningful outlier score. An example is illustrated in section 2.3 of Chapter 10, where deviations are computed on the different measured values of spatial data. In general, such methods are useful as a post-processing approach on a multidimensional vector of outlier scores, where each outlier score is derived using a different and possibly independent criterion. As discussed in Chapter 1, it is particularly common to confuse methods for extreme value analysis with general outlier analysis methods, which are defined in terms of

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

Figure 2.2. Pseudocode for finding depth-based outliers

generative probabilities. However, it is important to distinguish between the two, since the application-specific scenarios in which the two kinds of methods are used are quite different.

3.1 Depth-based Methods

In depth-based methods, convex hull analysis is used in order to find outliers. The idea is that the points in the outer boundaries of the data lie at the corners of the convex hull. Such points are more likely to be outliers. A depth-based algorithm proceeds in an iterative fashion. In the k-th iteration, all points at the corners of the convex hull of the data set are removed from the data set. These points are assigned a depth of k. These steps are repeated until the data set is empty. All points with depth at most r are reported as the outliers. The steps of the depth-based approach are illustrated in Figure 2.2.

The algorithm is also pictorially illustrated on a sample data set in Figure 2.3. A number of efficient methods for finding depth-based outliers have been discussed in [243, 388]. The computational complexity of convex-hull methods increases exponentially with dimensionality. Furthermore, with increasing dimensionality, a larger proportion of data points lie at the corners of a convex hull. This is because a convex hull in d-dimensional space contains at least 2^d points. Therefore, such methods are not only computationally impractical, but also increasingly ineffectual in higher dimensionality. Depth-based methods are generally quite different from most of the probabilistic and statistical models discussed in this chapter. In fact, they cannot really be considered prob-
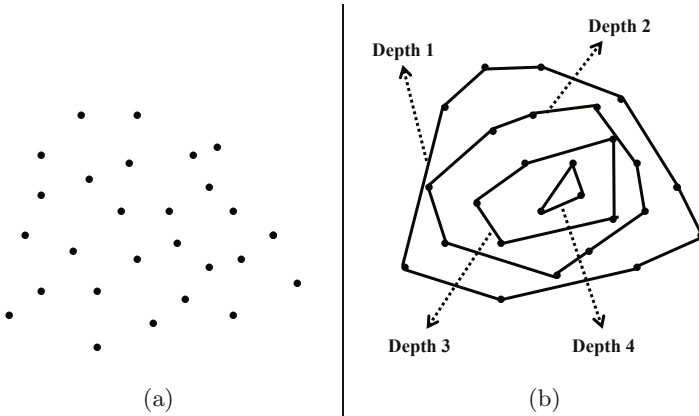
Figure 2.3. Depth-based outlier detection

abilistic or statistical methods. However, they are presented here, because all *multivariate extreme value* methods are presented at one place. Such methods share many characteristics in common, in spite of being methodologically different. For example, they work well only in scenarios where outliers lie at the boundaries of data space, rather than as isolated points in the interior of the data.

3.2   Deviation-based Methods

Deviation-based methods measure the impact of outliers on the data variance. For example, the method proposed in [49] tries to measure how much the variance in the underlying data is reduced, when a particular data point is removed. Since the basic assumption is that the outliers lie at the boundary of the data, it is expected that the removal of such data points will significantly reduce the variance. This is essentially an *information-theoretic* method, since it examines the reduction in complexity, when a data point is removed. Correspondingly, the *smoothing factor* for a set of data points R is defined as follows:

DEFINITION 2.10 *The smoothing factor $SF(R)$ for a set R is the reduction in the data set variance, when the set of points in R are removed from the data.*

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

$$SF(E) \geq SF(R)$$
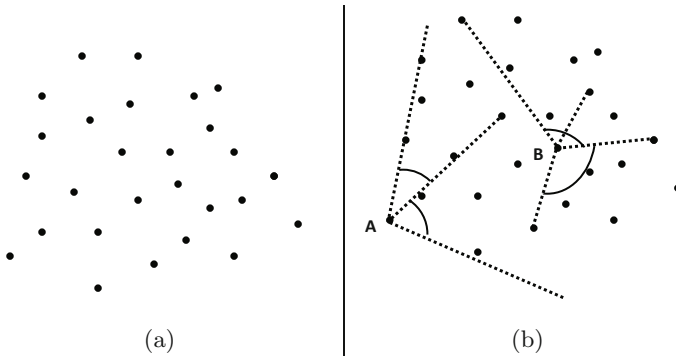
Figure 2.4. Angle-based outlier detection

If more than one set have the same reduction in variance, then the smaller set is preferred. This follows the standard information theoretic principle of finding the sets which increase the description length of the data as much as possible, in as little space. The determination of the optimal set E is a very difficult problem, because 2^N possibilities exist for a data set containing N points. The work in [49] uses a number of heuristics such as best-first search and random sampling. One good aspect of this approach is that it is distribution-independent, and can be applied to any kind of data set, as long as an appropriate definition of the smoothing factor can be constructed. In the original work in [49], this approach has been applied to the case of sequence data.

3.3   Angle-based Outlier Detection

This method was originally proposed as a general outlier analysis method, though this book has reclassified it to an extreme multivariate analysis method. The idea in angle-based methods is that data points at the boundaries of the data are likely to enclose the entire data within a smaller angle, whereas points in the interior are likely to have data points around them at different angles. For example, consider the two data points A and B in Figure 2.4, in which point A is an outlier, and point B lies in the interior of the data. It is clear that all data points lie within a limited angle centered at A. On the other hand, this is not the case for data point B, which lies within the interior of the data. In this case, the angles between different pairs of points can vary widely. In fact, the more isolated a data point is from the remaining points, the smaller the underlying angle. Thus, data points with a smaller angle spectrum are outliers, whereas those with a larger angle spectrum are not outliers.

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

$$WCos(\overline{Y} - \overline{X}, \overline{Z} - \overline{X}) = \frac{< (\overline{Y} - \overline{X}), (\overline{Z} - \overline{X}) >}{||\overline{Y} - \overline{X}||_2^2 \cdot ||\overline{Z} - \overline{X}||_2^2}$$

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

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

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

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

In order to speed up the approach, a natural possibility is to use sampling in order to approximate this value of the angle-based outlier factor. A sample of k data points can be used in order to approximate the ABOF of a data point $\overline{X}$. One possibility is to use an unbiased sample. However, since the angle-based outlier factor is inverse weighted by distances, it follows that the nearest neighbors of a data point have the largest contribution to the angle-based outlier factor. Therefore the k-nearest neighbors of $\overline{X}$ can be used to approximate the outlier factor much more effectively than a unbiased sample of the all the data points. It has also been shown in [269] that many data points can be filtered out on the basis of approximate computation, since their approximate values of the ABOF are too high, and they cannot possibly be outliers. The exact values of the ABOF are computed only for a small set of

points, and the points with the lowest values of the ABOF are reported as outliers. We refer the reader to [269] for the details of these efficiency optimizations. An approximation algorithm [363] for the problem has also been proposed in later work.

Because of the inverse weighting by distances, angle-based outlier analysis methods can be considered a hybrid between distance-based and angle-based methods. As discussed earlier with the use of the illustrative example, the latter factor is primarily optimized to finding multivariate extreme values in the data. The precise impact of each of these factors[1] does not seem to be easily quantifiable in a statistically robust way. In most data sets such as in Figure 2.7, outliers lie not just on the boundaries of the data, but also in the interior of the data. Unlike extreme values, outliers are defined by generative probabilities. While the distance factor can provide some impact for the outliers in the interior, the work is primarily focussed on the advantage of angular measures, and it is stated in [269] that the degree of impact of distance factors is minor compared to the angular factors. This implies that outliers on the boundaries of the data will be highly favored in terms of the overall score, because of the lower spectrum of angles. Therefore, the angle-based method treats outliers with similar generative probabilities in the interior and the boundaries of the data in a differential way, which is not statistically desirable for general outlier analysis. Specifically, the outliers at the boundaries of the data are more likely to be favored in terms of the outlier score. Such methods can effectively find outliers for the case illustrated in Figure 2.5, but the outlier A illustrated in Figure 2.7 will be favored less. Therefore, while this approach was originally presented as a general outlier analysis method, it has been classified in the section on multivariate extreme value analysis methods in this book.

It has been claimed in [269] that the approach is more suitable for high dimensional data because of its use of angles, as opposed to distances. However, it has been shown in earlier work [380], that angle-based measures are not immune to the dimensionality curse, because of concentration effects in the cosine measure. Such concentration effects would also impact the spectrum of the angles, even when they are combined with distances. The variation in the angle spectrum in Figure 2.4 is easy to show visually in 2-dimensional data, but the sparsity effects will also impact the spectrum of angles in higher dimensions. Therefore, the use of the spectrum of angles simply pushes the challenges of high dimensions to a different part of the analysis. A clear explanation of

---

[1]When a random variable is scaled by a factor of a, its variance is scaled by a factor of a^2. However, the scaling here is not by a constant factor.

why the spectrum of angles should be more robust to high dimensionality than distances has not[2] been provided in [269]. More importantly, such methods do not address the issue of locally irrelevant attributes [4], which are the primary impediment to effective outlier analysis methods with increasing dimensionality. Another important point to note is that multivariate extreme value analysis is much simpler than general outlier analysis in high dimensionality, because the parts of the data to explore are approximately known, and therefore the analysis is global rather than local. The evidence over different dimensions can be accumulated with the use of a very simple classical distance-distribution method [288, 406]. The approach, discussed in the next section, is also suitable for high-dimensional extreme value analysis, because it implicitly weights globally relevant and irrelevant directions in the data in a different way, and is statistically sound, in terms of probabilistic interpretability of the extreme values.

3.4   Distance Distribution-based Methods

A *distribution-dependent* approach is to model the entire data set to be normally distributed about its mean in the form of a multivariate Gaussian distribution. Let $\overline{\mu}$ be the d-dimensional mean vector of a d-dimensional data set, and Σ be its $d \times d$ co-variance matrix. In this case, the (i, j)th entry of the covariance matrix is equal to the covariance between the dimensions i and j. Then, the probability distribution $f(\overline{X})$ for a d-dimensional data point $\overline{X}$ can be defined as follows:

$$f(\overline{X}) = \frac{1}{\sqrt{|\Sigma|} \cdot (2 \cdot \pi)^{(d/2)}} \cdot \exp(-\frac{1}{2} \cdot (\overline{X} - \overline{\mu}) \cdot \Sigma^{-1} \cdot (\overline{X} - \overline{\mu})^T)$$

The value of $|\Sigma|$ denotes the determinant of the covariance matrix. We note that the term in the exponent is (half) the *Mahalanobis distance* between the data point $\overline{X}$ and the mean $\overline{\mu}$ of the data. The computation of the Mahalanobis distance requires the inversion of the covariance matrix Σ. The value in the exponent of the normal distribution above is used as the outlier score.

The Mahalanobis distance is similar to the euclidian distance, except that it normalizes the data on the basis of the inter-attribute correlations. For example, if the axis system of the data were to be rotated to

---

[2]The use of the cosine function in some high-dimensional domains such as text has been cited as an example in a later work [270]. In domains with small and varying non-zero attributes, the cosine is preferred because of important normalization properties, and not because of greater dimensionality resistance. The cosine function is not immune to the dimensionality curse even for the unique structure of text [380]. An increasing fraction of non-zero attributes, towards more general distributions, directly impacts the data hubness.
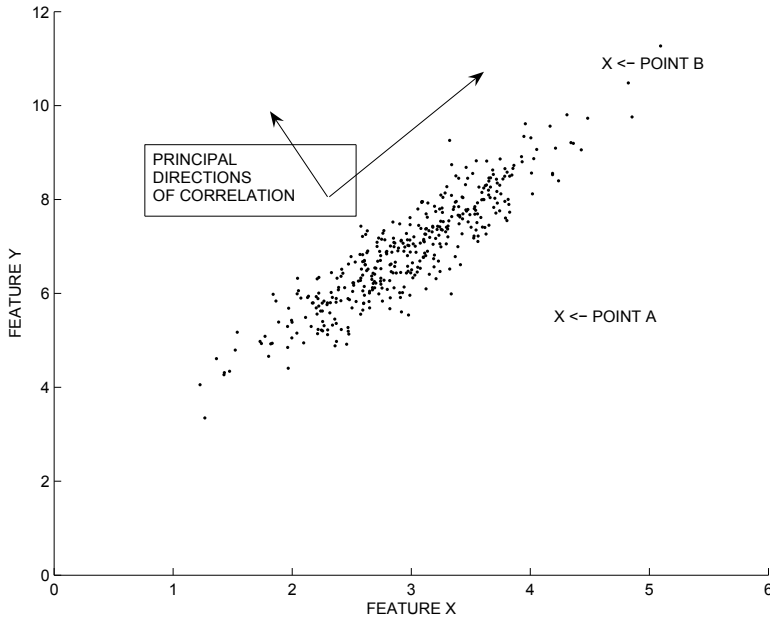
Figure 2.5. Extreme value analysis in multivariate data with Mahalanobis distance

the principal directions (shown in Figure 2.5), then the data would have no inter-attribute correlations. As we will see in section 3 of Chapter 3, it is actually possible to determine such directions of correlations generally in d-dimensional data sets. The Mahalanobis distance is simply equal to the Euclidean distance in such a transformed (axes-rotated) data set *after* dividing each of the transformed coordinate values by the standard-deviation of that direction. While principal component analysis can also be used in order to compute the value in the exponent of the normal distribution above, a simpler way to do it is by evaluating the term in the exponent of the modeled normal distribution. More will be discussed about this issue in Chapter 3.

This approach recognizes the fact that the different directions of correlation have different variance, and the data should be treated in a statistically normalized way along these directions. For example, in the case of Figure 2.5, the data point A can be more reasonably considered an outlier than data point B, on the basis of the natural correlations in the data. On the other hand, the data point A is closer to the centroid of the data (than data point B) on the basis of *euclidian distance*, but not on the basis of the Mahalanobis distance. Interestingly, data point A also seems to have a much higher spectrum of angles than data point B, at least from an average sampling perspective. This implies that,

at least on the basis of the primary criterion of angles, the angle-based method would likely favor data point B. This is because it is unable to account for the relative relevance of the different directions, an issue which becomes more prominent with increasing dimensionality. The Mahalanobis method is robust to increasing dimensionality, because it uses the covariance matrix in order to summarize the high dimensional deviations in a statistically effective way.

We further note that each of the distances along the principal correlation directions can be modeled as a one-dimensional standard normal distribution, which is approximately independent from the other orthogonal directions of correlation. As discussed earlier in this chapter, the sum of the squares of d variables drawn independently from a standard normal distributions, will result in a variable drawn from a χ^2 distribution with d degrees of freedom. Therefore, the cumulative probability distribution tables of the χ^2 distribution can be used in order to determine the outliers with the appropriate level of significance.

This simple approach is effective for the example of Figure 2.5, because the entire data set is distributed in one large cluster about the mean. For cases in which the data may have many different clusters with different orientations, such an extreme value approach may not be effective. An example of such a data set is illustrated in Figure 2.7. For such cases, more general distribution-based modeling algorithms are needed. This will be the subject of the discussion in the next section.

4. Probabilistic Mixture Modeling for Outlier Analysis

The previous section was focussed on the problem of extreme value analysis for outlier modeling. However, in practice, most outliers are defined on the basis of their *relative values* in multidimensional space, rather than simply being in the outer boundaries of the data. In such cases, the key idea is to use probabilistic mixture modeling of the data points. Such models are typically *generative* models, where for each data point, we can estimate the generative probability (or the fit probability) to the model. First, we assume a specific form of the generative model (eg. mixture of gaussians), and then estimate the parameters of this model with the use of the EM algorithm. The available data set is used in order to estimate the parameters in such as way, that they have a *maximum likelihood fit* to the generative model. Given this model, we then estimate the generative probabilities (or fit probabilities) of the underlying data points. Data points which fit the distribution well will have high fit probabilities, whereas anomalies will have very low fit
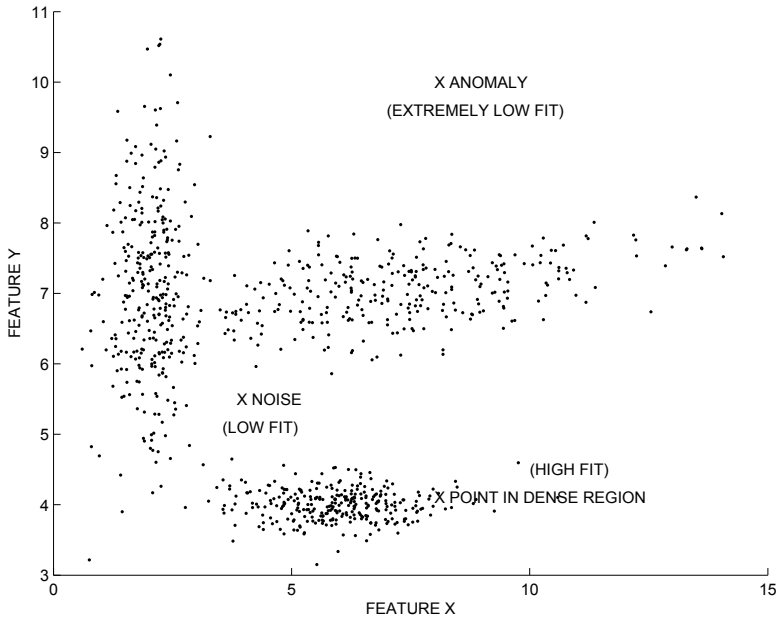
Figure 2.6. Relating Fit Probabilities to the Anomalous Behavior

probabilities. Some examples of how different kinds of data points would relate to the fit probability in such a model are illustrated in Figure 2.6.

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

- Pick a data distribution with probability α_i, where $i \in \{1 \ldots k\}$, in order to pick one of the k distributions. Let us assume that the rth one is picked.

- Generate a data point from $\mathcal{G}_r$.

We denote this generative model by $\mathcal{M}$. We note that the different values of α_i, and the parameters of the different distributions $\mathcal{G}_r$ are not known in advance. In some simplified settings, the values of the prior probabilities α_i may be fixed to $1/k$, though this value also needs to be learned from the data in the most general case. Typical forms for the distribution $\mathcal{G}_r$ include the Gaussian distribution. These need to be *estimated* from the data, so that the data has the maximum likelihood fit of being generated. Therefore, we first need to define the concept of the fit of the data set to a particular component of the mixture. Let us assume that the density function of $\mathcal{G}_i$ is given by $f^i(\cdot)$. The probability

(density function) of the data point $\overline{X_j}$ being generated by the model is given by:

$$f^{point}(\overline{X_j}|\mathcal{M}) = \sum_{i=1}^{k} \alpha_i \cdot f^i(\overline{X_j}) \qquad (2.19)$$

Then, for a data set $\mathcal{D}$ containing N records denoted by $\overline{X_1} \ldots \overline{X_N}$, the probability of the data set being generated by the model $\mathcal{M}$ is the product of the corresponding individual point probabilities.

$$f^{data}(\mathcal{D}|\mathcal{M}) = \prod_{j=1}^{N} f^{point}(\overline{X_j}|\mathcal{M})$$

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

$$\mathcal{L}(\mathcal{D}|\mathcal{M}) = \log(\prod_{j=1}^{N} f^{point}(\overline{X_j}|\mathcal{M})) = \sum_{j=1}^{N} \log(\sum_{i=1}^{k} \alpha_i \cdot f^i(\overline{X_j})) \qquad (2.20)$$

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

- (E-Step) Given current value of the parameters in Θ, determine the probability $P(\overline{X_j} \in \mathcal{G}_i|\Theta)$. This is the probability that the data point $\overline{X_j}$ was generated by component i.

- (M-Step) Given current probabilities of assignments of data points to clusters, use maximum likelihood approach to determine the

value of all the parameters Θ, which maximizes the log-likelihood fit on the basis of current assignments.

It now remains to explain the details of the E-Step and the M-Steps. The E-Step simply computes the probability density of the data point $\overline{X_j}$ being generated by each component of the mixture, and then computes the fractional value for each component. This provides the Bayes probability that the data point $\overline{X_j}$ was generated by component i (with model parameters fixed to the current set of the parameters Θ). Therefore, we have:

$$P(\overline{X_j} \in \mathcal{G}_i | \Theta) = \frac{\alpha_i \cdot f^{i,\Theta}(\overline{X_j})}{\sum_{r=1}^{k} \alpha_r \cdot f^{r,\Theta}(\overline{X_j})} \qquad (2.21)$$

With some abuse of notation, a superscript Θ has been added to the probability density functions in order to denote the fact that they are evaluated for model parameters Θ.

The M-step is slightly more involved. In order to optimize the fit, we need to compute the partial derivative of the log-likelihood fit with respect to corresponding model parameters, and set them to 0 in order to determine the optimal value. The values of α_i are easy to estimate and simply equal to the expected fraction of the points assigned to each cluster, based on the current values of $P(X_j \in \mathcal{G}_i | \Theta)$. In practice, in order to obtain more robust results for smaller data sets, the expected number of data points belonging to each cluster in the numerator is augmented by 1, and the total number of points in the denominator is $N + k$. Therefore, the estimated value is $(1 + \sum_{j=1}^{N} P(X_j \in \mathcal{G}_i | \Theta))/(k + N)$. This approach is also referred to as *Laplacian smoothing*.

In order to determine the other parameters specific to a particular component r of the mixture, we simply treat each value of $P(X_j \in \mathcal{G}_r | \Theta)$ as a weight of that data point in that component, and then perform maximum likelihood estimation of the parameters *of that component*. This is generally a much simpler process than having to deal with all components of the mixture at one time. For example, for a Gaussian mixture model in d dimensions, we have:

$$f^{r,\Theta}(\overline{X_j}) = \frac{1}{\sqrt{|\Sigma_r|} \cdot (2 \cdot \pi)^{(d/2)}} \cdot \exp(-\frac{1}{2} \cdot (\overline{X_j} - \overline{\mu_r}) \cdot \Sigma_r^{-1} \cdot (\overline{X_j} - \overline{\mu_r})^T)$$

Here $\overline{\mu_r}$ is the d-dimensional mean vector and Σ_r is the $d \times d$ co-variance matrix of the generalized Gaussian distribution of the rth component. The value of $|\Sigma_r|$ denotes the determinant of the covariance matrix. In practice, in order to minimize the number of estimated parameters, the non-diagonal entries are often set to 0. In such cases, the determinant
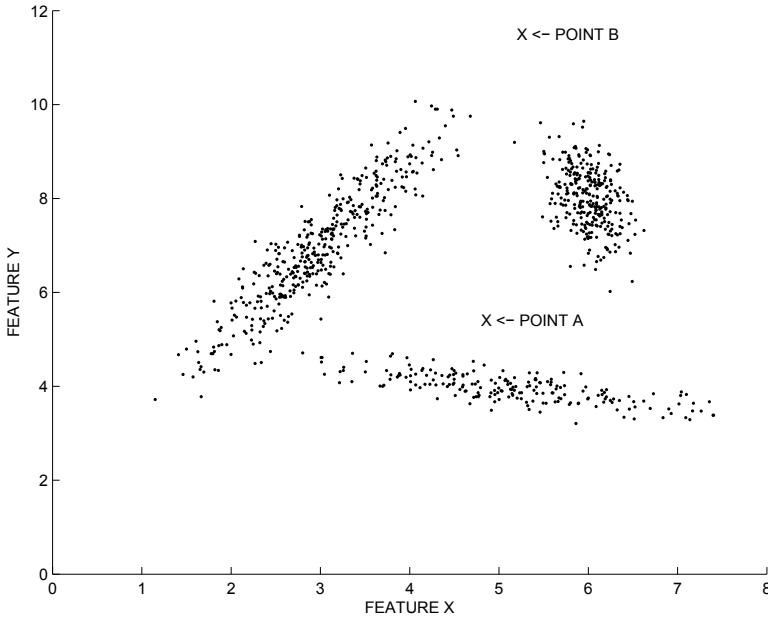
Figure 2.7.   EM-Algorithm can determine clusters with arbitrary correlations

of Σ_r simplifies to the product of the variances along the individual dimensions.

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

At the end of the process, we have a probabilistic model, which describes the entire data set in terms of a generative model. This model also provides a probabilistic fit value for each data point in the form of Equation 2.19. Thus, we can use this fit in order to rank all the data points, and determine the most anomalous ones. The idea is that points which are far away from the dense regions in the data (such as the one shown in the upper region of Figure 2.6) will have very low fit values. These points are the anomalies in the data. If desired, statistical hypothesis testing can be applied to the fit values in order to determine the data points whose fit values are extremely low.

The EM-algorithm can determine arbitrarily oriented clusters in the data, when the clusters have elongated shapes in different directions

of correlation. We note that the term in the exponent of the normal distribution is the Mahalanobis distance from the centroid of the cluster, which was introduced earlier in this chapter. As discussed earlier, this distance normalizes for the distances along the different directions of correlation in the data. The fit value is normalized along the different directions of correlation. For example, in the case of Figure 2.7, the fit of point A would be lower than that for point B, even though point B is closer to a cluster on the basis of absolute distances. This is quite appropriate, since data point A is more obviously an outlier.

The fit value of Equation 2.19 is a probability *density* value, and cannot be interpreted as a numerical probability. The ability to characterize an outlier in terms of numerical probabilities is a very useful step for intuition and interpretability. This issue is not specific to EM algorithms, but virtually all outlier detection algorithms, which output an outlier score with little physical interpretability. Interestingly, EM algorithms can also be used as a final step after many such outlier detection algorithms (including a first application of the EM method itself) for converting the scores into probabilities [179]. The idea is that the distribution of the outlier scores can be treated as univariate data set, which can then be fit to a probabilistic generative model. An example of such a generative model would be a mixture of exponential and gaussian functions, along with a special component for the mixture, known as the outlier class. These can be used in order to convert the outlier scores into probabilities with the use of the Bayes rule, since it is now possible to compute the probability that the data point belongs to the outlier component. We note that the assignment of a component of the mixture to the outlier class is critical in being able to estimate the probability that a data point is an outlier. A second approach would be to apply the EM-modeling *on the original data set* differently, and explicitly model an outlier class (rather than interpreting low fit values to the normal classes as outliers). However, such an approach, when applied directly to the original data set, is generally more useful for noise removal [88], and often does not work well for determining anomalies. The univariate case of outlier scores is much more easily addressed with such an approach because of the ability to model normal and outlier classes with more realistic (and different) distributions.

Probabilistic mixture modeling is a stochastic version of clustering methods, which can also be used for outlier detection. This is also closely related to outliers derived from generalized projected clustering [7], and those derived from generalized subspace analysis. However, in the parametric versions of the methods presented here, the large number of parameters required for the clustering process may sometimes

make the estimation process difficult. Methods for outlier detection in generalized subspaces will be presented in section 5 of Chapter 5.

5.   Limitations of Probabilistic Modeling

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

- When the particular assumptions of the model are too restrictive (eg. Gaussian distribution, specific number of clusters etc.), the data is unlikely to fit the model well. As a result, a lot of spurious data points may be reported as outliers.

- When the model is too general, the number of parameters to describe the model increases. This may overfit the data, and miss the true outliers. This case is more common in parametric modeling, especially when the data sizes are small.

The ability of such methods to distinguish between noise and abnormalities is limited because of several simplifying assumptions, which ensure that most of the commonly used models are not a very good match for real distributions. For example, the clusters in the data may be of arbitrary shape, and may not fit the Gaussian assumption well. Furthermore, a common assumption is that the data values along different dimensions are independent of one another. This corresponds to a matrix Σ_r which is diagonal, in the Gaussian case. This is because it is much harder to learn $O(d^2)$ parameters in a d-dimensional data, than to learn d parameters.

In real data sets, significant correlations may exist among the different dimensions. Therefore, such assumptions could result in poor fitting of the model to the data. We further note that the overall distribution is usually assumed to be a product of several 1-dimensional Gaussians. This is also referred to as the *naive* independence assumption. The use of such an independence assumption implies that it is much harder to interpret the point membership probabilities as true probabilities of membership of data points in a cluster. This reduces the attractiveness of probabilistic methods, where the primary claim is to be able to explicitly model probabilities. The number of parameters also increases with the dimensionality of the data, and this further reduces the effectiveness

of the method with increasing dimensionality. In practice, parametric methods require a large number of data points in order to work well.

Parametric methods are usually not very efficient for large data sets. This is because these methods use the iterative EM algorithm, which needs to scan the entire data step in each iteration of the E- and M-steps. This can be rather slow, when a large number of iterations are required. A significant amount of work has been performed in the data mining community in order to extend outlier analysis methods to non-parametric scenarios. For example, non-parametric variations of clustering and distance-based techniques are much more effective for outlier analysis. Many of these methods have also been scaled to large data sets, and generally do not overfit the data quite as much. These methods will be discussed in later chapters of this book.

Finally, the issue of *interpretability* remains a concern for many parametric methods. For example, consider the generalized Gaussian model, which tries to learn clusters with non-zero co-variances. In such a case, it is difficult to intuitively interpret the clusters with the use of these parameters. Correspondingly, it is also difficult to define simple and intuitive rules, which provide critical ideas about the underlying outliers. We note that this issue may not necessarily be a problem for *all* parametric methods. If the parameters are chosen carefully enough, then the final model can be described simply and intuitively. For example, simplified versions of the Gaussian model without co-variances may sometimes be described simply and intuitively in terms of the original features of the data. On the other hand, such simplifications may not provide very good results in terms of quality. Such tradeoffs remain a major challenge for parametric methods.

6. Conclusions and Summary

In this chapter, a number of fundamental probabilistic and statistical methods for outlier analysis were introduced. Such techniques are very useful for confidence testing and extreme-value analysis. A number of tail inequalities for extreme value analysis were also introduced. These methods can also be generalized to the multivariate scenario. Extreme value analysis has immense utility as a final step in converting the scores from many outlier analysis algorithms into binary labels. The EM approach for probabilistic mixture modeling of outliers was introduced in this chapter. Probabilistic modeling provides a formal framework for quantification of numerous algorithms and methods, which will be discussed later in this book.

7.   Bibliographic Survey

The classical inequalities such as the Markov, Chebychev, Chernoff and Hoeffding are widely used in probability and statistics for bounding the accuracy of aggregation-based statistics. A detailed discussion of these different methods may be found in [342]. A generalization of the Hoeffding's inequality is the McDiarmid's inequality [330], which can be applied to a more general function of the different values of X_i (beyond a linearly separable sum). The main restriction on this function is that if the ith argument of the function (i.e. the value of X_i) is changed to any other value, the function cannot change by more than c_i.

The central limit theorem has been studied extensively in probability and statistics [70]. Originally, the theorem was proposed for the case of sums of independent and identically distributed variables. Subsequently, it was extended by Aleksandr Lyapunov to cases where the variables are not necessarily identically distributed [70], but they do need to be independent. A weak condition is imposed on these distributions, which ensures that the sum is not dominated by a few of the components. In such a case, the sum of the variables converges to the normal distribution as well. Thus, this is a generalized version of the Central Limit Theorem.

Statistical hypothesis testing has been used widely in the literature in order to determine statistical levels of significance for the tails of distributions [58]. A significant literature exists on hypothesis testing, where the anomalous properties of not just individual data points, but also the collective behavior of groups of data points can be tested. Such techniques are also used in online analytical processing scenarios where the data is organized in the form of data cubes. It is often useful to determine outliers in different portions of a data cube with the use of hypothesis testing [392].

The statistical method for deviation detection with variance reduction was first proposed in [49]. Angle-based methods for extreme value analysis in multivariate data were proposed in [269]. A more efficient approximation algorithm, which is based on this model was recently proposed in [363]. The multivariate method for extreme value analysis with the use of the Mahalanobis distance was proposed in [288]. This technique does not work well, when the outliers lie in sparse regions between clusters. A number of depth-based methods have been proposed in [243, 388]. These methods compute the convex hull of a set of data points, and progressively peel off the points at the corners of this hull. The depth of a data point is defined as the order of convex hull which is peeled. These techniques have not found much popularity because they suffer the same drawback as the method of [288] for finding internally

located outliers. Furthermore, convex hull computation is extremely expensive with increasing dimensionality. Furthermore, with increasing dimensionality, an increasing proportion of the points will lie on the outermost convex hull. Therefore, such methods can only be applied to 2- or 3-dimensional data sets in practice.

It should be noted that the use of probabilistic methods for outlier detection is distinct from the problem of outlier detection in probabilistic or uncertain data [23, 238, 459]. In the former case, the data is uncertain, but the methods are probabilistic. In the latter case, the data itself is probabilistic. The seminal discussion on the EM-algorithm is provided in [135]. This algorithm has a particularly simple form, when the components of the mixture are drawn from the exponential family of distributions. The work in [478] proposed an *online* mixture learning algorithm, which can handle *both* categorical and numerical variables. An interesting variation of the EM-algorithm treats one component of the mixture model specially as an anomaly component [154]. Correspondingly, this component is drawn from a uniform distribution [154], and is also assigned a low a-priori probability. Therefore, instead of determining the anomalous points which do not fit any mixture component well, this approach tries to determine the points which fit this special component of the mixture. Such an approach would generally be more effective at modeling noise rather than anomalies, because the special component in the mixture model is likely to model the noise patterns. Finally, a Gaussian Mixture Model has also been used recently in order to create a global probabilistic model for outlier detection [483].

The EM-algorithm has also been used for clutter removal from data sets [88]. In this case, noise is removed from the data set, by modeling the derived data as a mixture of Poisson distributions. We note that the approach in [88] is designed for noise detection, rather than determining true anomalies. It was shown in [88] that the improvement in data quality after removal of the clutter (noise) was significant enough to greatly ease the identification of relevant features in the data. The approach of using a special component of the mixture in order to convert the distribution of outlier scores into probabilities has been used in [179], which is discussed in some detail below.

Extreme value analysis has always been an important problem in outlier analysis because of its utilization as a final step in most outlier detection algorithms. Some recent work has been done [179] on the issue of probabilistic modeling of outlier scores in order to determine the extreme values from these scores. Two methods are proposed in this work. Both of these techniques use parametric modeling methods. The first method assumes that the posterior probabilities follow a logistic

sigmoid function. The underlying parameters are then learned from the
EM framework from the distribution of outlier scores. The first approach
assumes that the posterior probabilities follow a logistic sigmoid func-
tion and learns the parameters of the function from the distribution of
outlier scores. The second approach recognizes the fact that the outlier
scores of data points in the outlier component of the mixture is likely to
show a different distribution (Gaussian distribution), than the scores of
data points in the normal class (Exponential distribution). Therefore,
this approach models the score distributions as a mixture of exponen-
tial and Gaussian probability functions. As before, the parameters are
learned with the use of the EM-framework. The posterior probabilities
are calculated with the use of the Bayes rule. Finally, a method was
proposed in [271] to improve the effectiveness of converting the scores
into probabilities. Methods for converting outlier scores into probability
in the supervised scenario have been discussed in [495].

8.   Exercises

1. **[Upper Tail Chernoff Bound]** The chapter provides a proof
   sketch of the upper-tail Chernoff bound, but not the full proof.
   Work out the full proof of bound on the upper tail, using the lower
   tail proof as a guide. Where do you use the fact that $\delta < 2 \cdot e - 1$?

2. Suppose you flip an "unbiased" coin 100 times. You would like
   to investigate whether the coin is showing anomalous behavior (in
   terms of not being "unbiased" as claimed). Determine the mean
   and standard deviation of the random variable representing the
   number of "tails", under the assumption of an unbiased coin. Pro-
   vide a bound on the probability that you obtain more than 90 tails
   with the use of the (i) Markov Inequality (ii) Chebychev Inequality
   (iii) Chernoff Upper Tail Bound, (iv) Chernoff Lower Tail Bound
   and (v) Hoeffding Inequality. [Hint: Either the upper tail or lower
   tail Chernoff bound can be used, depending upon which random
   variable you look at.]

3. Repeat exercise 2, when you know that the coin is rigged to show
   "tails" every eight out of nine flips. Do you get meaningful bounds
   with the Markov and Chebychev inequalities? What does this tell
   you?

4. Use the central limit theorem to approximate the number of tails
   by a normal distribution. Use the cumulative normal distribution
   to approximate the probability that the number of "tails" should
   be more than 90 for both the cases of exercises 2 and 3.

5. A manufacturing process produces widgets, each of which is 100 feet long, and has a standard deviation of 1 foot. Under normal operation, these lengths are independent of one another.

   ■ Use the normal distribution assumption to compute the probability that something anomalous is going on in the manufacturing process, if a sampled widget is 101.2 feet long?

   ■ How would your answer change, if the sampled widget was 96.3 feet long?

6. In the example above, consider the case where 10,000 widgets from the assembly line were sampled, and found to have an average length of 100.05. What is the probability that something anomalous is going on in the manufacturing process?

7. Use MATLAB or any other mathematical software to plot the t-distribution with 100 degrees of freedom. Superimpose a standard normal distribution on this plot. Can you visually see the difference? What does this tell you?

8. Work out the steps of the EM-algorithm, when all non-diagonal elements of the covariance matrix Σ are set to zero, and each diagonal element in a given component has the same value. Now perform the following modifications:

   ■ Change the E-step, so that each data point is deterministically assigned to the cluster with the highest probability (hard assignment), rather than a soft probabilistic assignment. Under what distance-based conditions does a data point get assigned to a cluster?

   ■ How does this algorithm relate to the k-means algorithm?

   ■ How would your answers change, if all components were constrained to have the same cluster variance?

9. Using the insights gained from Exercise 8, work out how the EM-algorithm with a Gaussian mixture model with a complete set of covariance matrices Σ_r, and a fixed set of priors, relates to a generalized k-means algorithm. [Hint: Consider the concept of Mahalanobis distance computations for assignments in k-means. How should the prior probabilities be defined?]

10. Download the KDD Cup 1999 data set from the UCI Machine Learning Repository [169]. Extract the quantitative attributes

from the data set. Apply the EM-algorithm with 20 mixture components, when non-diagonal elements are set to 0.

- Determine the fit of each data point to the learned distribution. Determine the top 10 points with the least fit. Do these data points correspond to intrusion attacks or normal data?
- Repeat the process while allowing non-zero non-diagonal elements. How does your answer change?
- Randomly sample 990 points from the data set, and then add the 10 points found in the first case above. Repeat the procedure on this smaller data set. Do you find significant anomalies in terms of fit probabilities? Do the lowest fit probabilities correspond to the same data points as in the first case above?
- Repeat the same procedure with the second case above.

11. Repeat the first two portions of Exercise 9 on the Ionosphere data set from the UCI Machine Learning Repository. Note that the Ionosphere data set has much higher dimensionality (of quantitative attributes) and smaller number of records. Do you determine the same top-10 anomalies in the two cases? What are the absolute fit probabilities? What does this tell you about applying such algorithms to small and high dimensional data sets?

12. Let Z be a random variable satisfying $E[Z] = 0$, and $Z \in [a, b]$.

- Show that $E[e^{t \cdot Z}] \le e^{t^2 \cdot (b-a)^2/8}$.
- Use the aforementioned result to complete the proof of the Hoeffding inequality.

Chapter 3

LINEAR MODELS FOR OUTLIER DETECTION

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

1. Introduction

The different dimensions in real data sets are highly correlated with one another. This is because the different attributes are usually generated by the same underlying process in closely related ways. In the classical statistics literature, this is referred to as *regression modeling*, a parametric form of correlation analysis. Some forms of correlation analysis attempt to predict individual attribute values from others, whereas other forms summarize the entire data in the form of latent variables. An example of the latter is the method of *principal component analysis.* Both forms of modeling can be very useful in different scenarios of outlier analysis. This chapter will discuss the different methods for using linear correlation analysis for outlier detection.

The main assumption of this model is that the data is embedded in a lower dimensional subspace. In the case of proximity-based methods, which will be discussed in the next chapter, the goal is to determine specific *regions of the space* in which outlier points behave very differently from other points. On the other hand, in linear methods, the goal is to find *lower dimensional subspaces*, in which the outlier points behave very differently from other points. This can be viewed as an orthogonal point of view to clustering- or nearest-neighbor based methods, which try to summarize the data *horizontally* (i.e. on the rows or data values), rather than *vertically* (i.e. on the columns or dimensions). As will be discussed in the chapter on high-dimensional outlier detection, it is in

principle, possible to combine these methods for more general local subspace models, which can determine outliers on the basis of a combination of horizontal and vertical criteria.

The assumption of approximately linear correlations is a critical one for ensuring the effectiveness of the model. This may or may not be true for a given data set. For example, consider the behavior of two data sets from the *UCI Machine Learning Repository* [169]. In particular, consider the behavior of the *Autompg* and *Arrythmia* data sets from this repository. The first data set measures various characteristics of cars, and relates them to the mileage (mpg) of the cars. The second data set contains different kinds of features derived from ECG readings of human patients.

In the first set of Figures 3.1(a) and (b), the dependence of the *Miles per Gallon* attribute has been shown on each of the *displacement* and *horsepower* attributes respectively for the *Autompg* data set. It is evident that a significant level of correlation exists between these attributes. While a significant amount of noise exists in the data, the linear dependence between the attributes is apparent. In fact, it can be shown for this data set, that with increasing dimensionality (by picking more attributes from the data set), the data can be aligned along much lower dimensional planes. This is also evident in the 3-dimensional plot of Figure 3.1(e). On the other hand, when various views along three of the measured dimensions of the *Arrythmia* data set (Figures 3.1(c), (d) and (f)) are examined, it is evident that the data separates out into two clusters, one of which is slightly larger than the other. Furthermore, it is rather hard to embed this kind of data distribution into a lower dimensional subspace. This data set is much more suitable for proximity-based analysis, which will be presented in Chapter 4. The reason for introducing this example is to revisit the point made in the first chapter about the impact of the choices made during the crucial phase of picking the correct data model. In general, the most difficult case is when different views of the *same* data set may be suitable for different models. Such data sets are best addressed with the use of subspace methods discussed in Chapter 5, which can combine the power of row and column selection for outlier analysis. However, in many cases, simplified models such as linear models or proximity-based models are sufficient, without incurring the complexity of subspace methods. From a model-selection perspective, exploratory and visual analysis of the data is rather critical in the first phase of outlier detection in order to find out whether a particular data model is suitable for a particular data set. This is particularly true in the case of unsupervised data models.
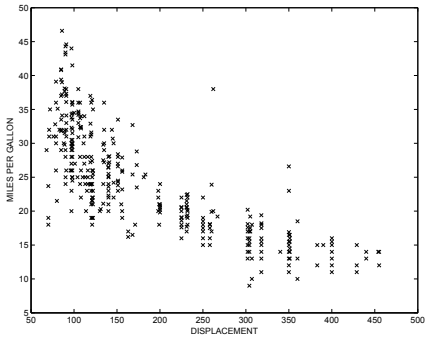
(a) View 1 (*Autompg*)
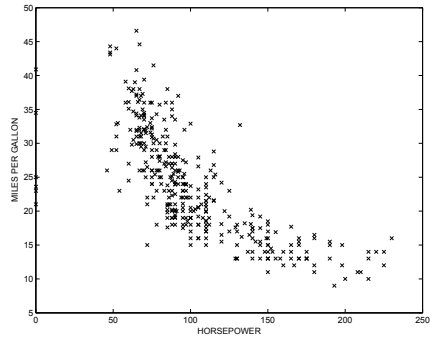
(b) View 2 (*Autompg*)
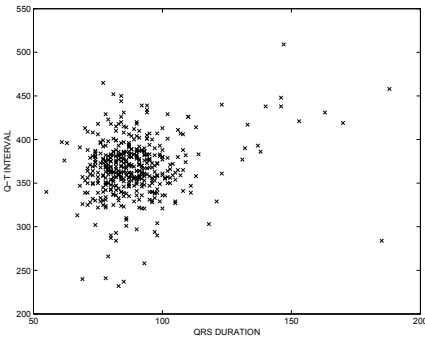
(c) View 1 (*Arrythmia*)
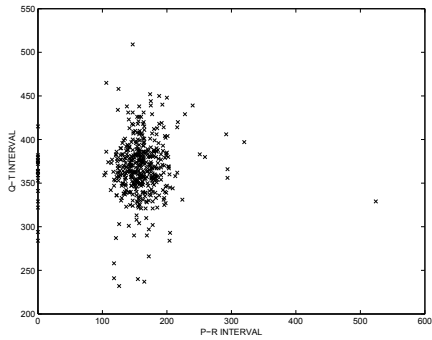
(d) View 2 (*Arrythmia*)
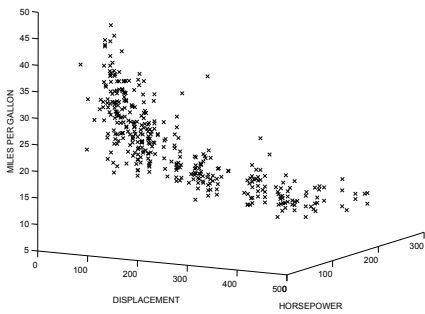
(e) 3-d View (*Autompg*)
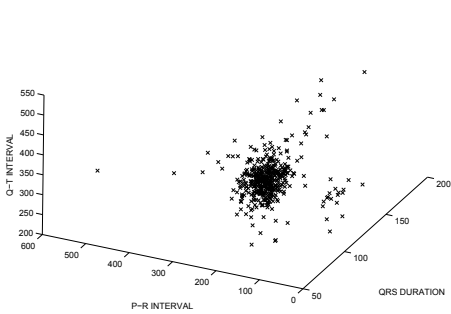
(f) 3-d View (*Arrythmia*)

Figure 3.1.   Effectiveness of linear assumption is data set dependent

In this chapter, two main classes of linear models will be studied. The first class of models uses statistical regression modeling between dependent and independent variables in order to determine *specific kinds of dependencies* in the data. Such forms of modeling are more useful when some of the attributes in an application should be monitored on a prioritized basis (eg. the *last* value of a time-series, where the previous history of values are the independent variables used for modeling). The second class of models uses principal component analysis in order to treat all attributes in a homogeneous way, and determine the lower dimensional subspaces of projection. At a technical and mathematical level, both forms of modeling are quite similar, and use very similar methods in order to derive the optimal lower dimensional representations. The main difference is in how the objective function of the two models is formulated.

It should be emphasized that regression-analysis is used extensively to detect anomalies in time-series data, and many of the basic techniques discussed in this chapter are applicable to that scenario as well. However, since the time-series aspect of the problem is also based on dependencies of *temporally adjacent* data values, there are a number of subtle differences in how anomalies are detected in those cases. Therefore, in this chapter, the much simpler case of multidimensional outlier analysis will be addressed. At the same time, the discussion will be general enough, so that the fundamentals necessary for the discussion of applying regression analysis in the time-series scenario (Chapter 8) are introduced.

This chapter is organized as follows. In section 2, the basic linear regression models for outlier analysis will be introduced. In section 3, the principal component method for outlier analysis will be introduced. This can be considered an important special case of linear regression models, which is used frequently in outlier analysis. Therefore it is given a dedicated treatment in its own section. Section 4 will study the limitations of linear models for outlier analysis. Section 5 contains the conclusions and summary.

2.   Linear Regression Models

In linear regression, the observed values in the data are modeled using a linear system of equations. Specifically, the different dimensions in the data are related to one another using a set of linear coefficients. Since the number of observed values are typically much larger than the dimensionality of the data, this system of equations is an *over-determined* one, and cannot be solved exactly. Therefore, these models optimize

the square error of the deviations of data points from values predicted by the linear model. The exact choice of the error function determines whether a particular variable is treated specially (i.e. error of predicted variable value), or whether variables are treated homogeneously (i.e. error distance from estimated lower dimensional plane). These different choices of the error function do *not* lead to the same model. In fact, as the following discussion will show, the models can be very different *especially in the presence of outliers.*

Regression analysis is generally considered an important application of its own in statistics. In classical instantiations of this application, it is desirable to learn a specific dependent variable from a set of independent variables. This is a common scenario in time-series analysis, which will be discussed in detail in Chapter 8. Thus, a specific variable is treated *specially* from the other variables. Most applications on outlier analysis do not treat any particular variable as special, and the definition of outliers is generally based on the *overall* distribution of the underlying data points. However, the special case of regression analysis with dependent variables is also important in many applications. This is because in many real-life domains such as temporal and spatial data, the attributes are partitioned into *contextual* and *behavioral* attributes. In such cases, a particular behavioral attribute value is predicted as a function of the behavioral attributes in its *contextual* neighborhood in order to determine deviations from expected values. Therefore, the importance of the dependent variable is paramount. In such cases, outliers are defined on the basis of how other independent variables impact the dependent variable, and anomalies within the relationships of independent variables with each other are considered less important. The identification of outliers in such cases is also very useful for *noise reduction* in regression modeling, which is an important problem in its own right. This problem is considered so important, that an entire book has been devoted to this subject [387]. Therefore, the special case of regression analysis with dependent variables will be studied first. Then, the general application of regression methods to outlier analysis will be discussed. The focus in this section is to discuss the impact of outliers on the linear modeling process of a *dependent variable* on a set of *explanatory* variables. The discussion of this case also sets the stage for a more detailed discussion for the cases of time-series data in Chapter 8, and spatial data in Chapter 10.

In a later subsection, the more general problem of utilizing regression modeling for generic outlier analysis will be discussed. In that case, no particular variable is considered special, and regression modeling is a *tool* (rather than an application in its own right). Such a tool may be used

either to remove noise for other applications, or to identify interesting anomalies. This latter form of the problem is the focus of most of this book, though dependent variable regression analysis is also important in many applications such as time-series data.

2.1   Modeling with Dependent Variables

A variable Y can be modeled as a linear function of d dependent variables as follows:

$$Y = \sum_{i=1}^{d} a_i \cdot X_i + a_{d+1}$$

The variable Y is the response variable or the dependent variable, and the variables $X_1 \ldots X_d$ are the independent or the explanatory variables. The coefficients $a_1 \ldots a_{d+1}$ need to be learned from the data. The data may contain N different instances, which provide examples of how Y may be related to the different values of X_i. The jth instances of the data are denoted by $(x_{j1} \ldots x_{jd})$ and y_j. The jth instance of the response variable is related to the explanatory variables as follows:

$$y_j = \sum_{i=1}^{d} a_i \cdot x_{ji} + a_{d+1} + \epsilon_j$$

Here ϵ_j represents the error in modeling the jth instance. In *least squares regression*, the goal is to determine the regression coefficients $a_1 \ldots a_{d+1}$, which minimize the error $\sum_{j=1}^{N} \epsilon_j^2$. The $N \times (d+1)$-matrix whose j-th row is $(x_{j1} \ldots x_{jd}, 1)$ is denoted by U, and the $N \times 1$ matrix of the different values of Y is denoted by V. Thus, the first d dimensions of U can be considered a d-dimensional data set containing the N instances of the independent variables, and V is corresponding vector of response variables. The $(d+1) \times 1$ column vector of coefficients $a_1 \ldots a_{d+1}$ is denoted by A. This creates an *over-determined* system of equations denoted by:

$$V \approx U \cdot A \tag{3.1}$$

The least-squares error *of predicting the response variable* is optimized by minimizing $||V - U \cdot A||$ over all values of the coefficient A. It will be seen later, that more general ways of formulating the error function may exist, rather than simply predicting the error of the response variable. Clearly, the choice of the error function has an impact on the optimal hyperplane found by the regression analysis process. It can be shown through simple optimization methods via differential calculus, that the optimal coefficients for this minimization problem is provided by the

following equation:

$$A = (U^T \cdot U)^{-1} \cdot (U^T \cdot V) \tag{3.2}$$

Note that $U^T \cdot U$ is a $(d+1) \times (d+1)$ matrix, which needs to be inverted in order to solve this system of equations. The system of equations above thus needs to be over-determined in order for the matrix $U^T \cdot U$ to have full rank, and be invertible. The closed form solution to this problem is particularly convenient, and is one of the cornerstones of regression analysis in classical statistics. It is useful to examine the special case of two dimensional data:

$$Y = a_1 \cdot X_1 + a_2 \tag{3.3}$$

In this case, the estimation of the coefficient a_1 has a particularly simple form, and it can be shown that the best estimate for a_1 is as follows:

$$a_1 = \frac{Cov(X_1, Y)}{Var(X_1)}$$

Here $Var(\cdot)$ and $Cov(\cdot)$ correspond to the variance and covariance of the underlying random variables. The value a_2 can further be easily estimated, by plugging in the means of X_1 and Y into the linear dependence, once a_1 has been estimated. In general, if X_1 is regressed on Y instead of the other way around, one would have obtained $a_1 = \frac{Cov(X_1, Y)}{Var(Y)}$. Note that the regression dependencies would have been different for these cases. This shows the impact of the error term on the final regression plane which is found by the method.

The set of coefficients $a_1 \ldots a_{d+1}$ define a lower dimensional hyperplane which fits the data as well as possible in order to optimize the error in the dependent variable. This hyperplane may be different for the same data set, depending upon which variable is chosen as the dependent variable. In order to explain this point, let us examine the behavior of two attributes from the *Auto-Mpg* data set of the UCI Machine Learning repository [169].

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

- One regression plane is drawn for the case, when the *Horsepower (Y-axis)* is dependent on the *Displacement (X-axis)*. The residual in this case is the error of prediction of the *Horsepower* attribute. The sum of squares of this residual is optimized.
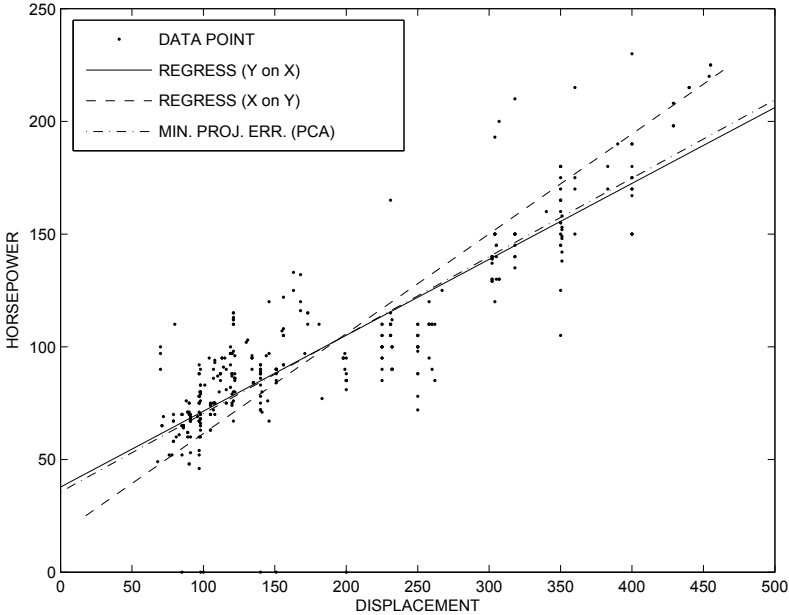
Figure 3.2.   Optimal regression plane depends upon the choice of residual which is optimized

- The second regression plane is drawn for the case, when the *Displacement (X-axis)* is dependent on the *Horsepower (Y-axis)*. The residual in this case is the error in prediction of the *Displacement* attribute.

- In the last case, the goal is to optimize the mean square error of the data points in terms of their absolute distance to the best fitting hyperplane. Thus, the residual in this case is the distance of each point to the hyperplane, in a direction which is normal to the hyperplane. Thus, this hyperplane minimizes the mean square distances between the data points, and their projection into the hyperplane. So far, the determination of such a hyperplane has not been discussed. This will be done in a later section on Principal Component Analysis (PCA).

It is evident from Figure 3.2 that the optimal hyper-planes in these different cases are quite different. While the optimization of the mean square projection distance produces a hyperplane which is somewhat similar to the case of Y-on-X regression, the two are not the same. This is because these different cases correspond to different choices of errors on the residuals which are optimized, and therefore correspond to

(a) Regression with no
outliers

(b) Regression with one
perturbed point
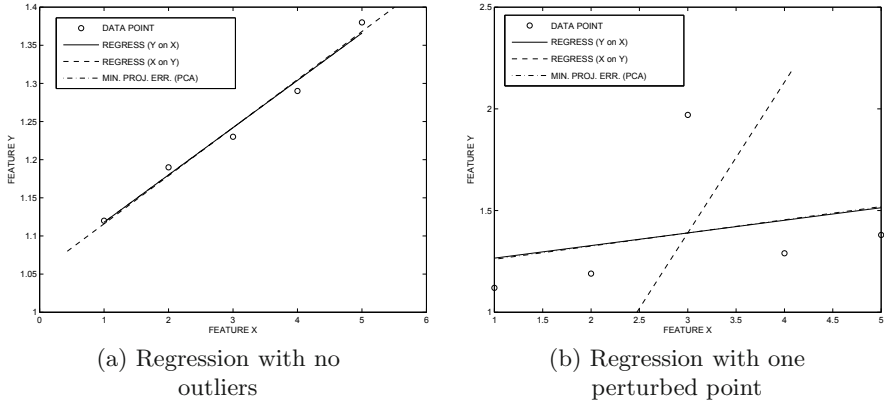
Figure 3.3. Drastic effects of outliers on quality of regression analysis

different best fitting hyperplanes. It is also noteworthy that the three projection planes are collinear and pass through the mean of the data set.

When the data fits the linear assumption very well, all these hyperplanes are likely to be very similar and not very different from one another. However, the presence of noise and outliers can result in rather drastic negative effects on the modeling process, when some of the outliers show significant deviations. In order to illustrate this point, a variation of an example from [387] is used. In Figure 3.3, the different regression planes for two sets of five data points have been presented corresponding to different dependent variables. The two sets of five data points in Figures 3.3(a) and (b) are different by only one point, in which the Y-coordinate was assumed to be somehow perturbed during data collection. As a result, this point does not fit the remaining data very well.

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

Clearly, the removal of outliers is crucial in such applications, in order to improve the quality of the regression analysis. Therefore, a useful approach would be to examine the residuals ϵ_j, and remove those data points which are detrimental for outlier analysis. The mean of these residuals is expected to be 0, and the variance of these residuals can be estimated directly from the data.

The most common assumption for outlier analysis is to assume that the error term ϵ_i is a normal distribution, which is centered at zero. Then, the t-value test discussed in Chapter 2 can be used directly on the different residuals, and the outlying observations can be subsequently removed. The normal assumption on the residuals implies that the vector of coefficients is also normally distributed with mean and variances, as discussed earlier. When the outliers have drastic effects on the regression, such as in the case of the X-on-Y regression in Figure 3.3(b), the removal of outliers is likely to result in the removal of the wrong observations, since the regression parameters are drastically incorrect. On the other hand, in all cases, the projection based minimization seems to provide more robust results (as opposed to picking a particular dependent variable) to the presence of outliers. Therefore, even for dependent variable analysis, it may sometimes be helpful to use such projection-based error minimization. This is the method of *Principal Component Analysis (PCA)*. The formulation for this case will be discussed in the next subsection, and a more detailed discussion of the solution and different aspects of principal component analysis will be discussed in a dedicated section of its own.

2.2   Regression Modeling for Mean Square Projection Error

The previous section discussed the case, where a particular variable is considered special, and the optimal plane is determined in order to minimize the mean-square error of the residuals for this variable. In the most general form of regression-modeling, all variables are treated in a similar way, and the optimal regression plane is determined the minimize the *projection error* of the data to the plane. This can be considered an unsupervised form of outlier analysis, because the outliers are determined without treating any particular variable specially.

The projection error of the data to the plane is the sum of the squares of the distances of the points to their projection into the plane. The projection of a point to the plane is performed by using the normal direction to the plane which passes through the data point and the plane. The point at which this normal intersects the plane is the projection

point. Thus, in this case, let us assume that we have a set of variables $X_1 \ldots X_d$, and the corresponding regression plane is as follows:

$$a_1 \cdot X_1 + \ldots + a_d \cdot X_d + a_{d+1} = 0 \qquad (3.4)$$

Each variable is associated with a coefficient, and the "special" (dependent) variable (without a coefficient) is missing in this case. For simplification of the subsequent discussion of computing distances of different observations to this plane, a normalization constraint will be assumed.

$$\sum_{i=1}^{d} a_i^2 = 1 \qquad (3.5)$$

Note that the $(d + 1)$th term (constant coefficient) is not used in the normalization. As before, let U be a $N \times (d + 1)$ matrix containing the set of N observations corresponding to the variables $X_1 \ldots X_d, 1$. The last column in the matrix U corresponds to the constant term, and therefore only contains unit values. Let A be a column vector containing $a_1 \ldots a_{d+1}$. It can be shown that the N-dimensional column vector of distances for the different data points to this regression plane is given by $U \cdot A$. The L_2-norm $||U \cdot A||_2$ of the column vector of distances is the objective function, which needs to be minimized over the different possible values of the coefficients $a_1 \ldots a_{d+1}$, under the normalization constraint. It can be shown that a effective (and much more general) solution to the problem can be obtained with Principal Component Analysis (PCA). Because of its importance to outlier analysis, this method will be discussed in a dedicated section of its own, along with corresponding applications.

3.   Principal Component Analysis

The least-squares formulation of the previous section simply tries to find a *single* $(d-1)$-dimensional hyperplane which has an optimum fit to the data values. The principal component analysis method can be used to solve a *generalized* version of this problem. Specifically, it can find optimal representation hyperplanes of *any* dimensionality. Specifically, the PCA method can determine the k-dimensional hyperplane (for any value of $k < d$), which minimizes the squared projection error. In principal component analysis, the $d \times d$ covariance matrix over d-dimensional data is computed, where the (i, j)th entry is equal to the covariance between the dimensions i and j for the set of N observations of the variables $X_1 \ldots X_d$.

It is easier to think in terms of a multidimensional data set of dimensionality d and size N, rather than a set of d variables with N
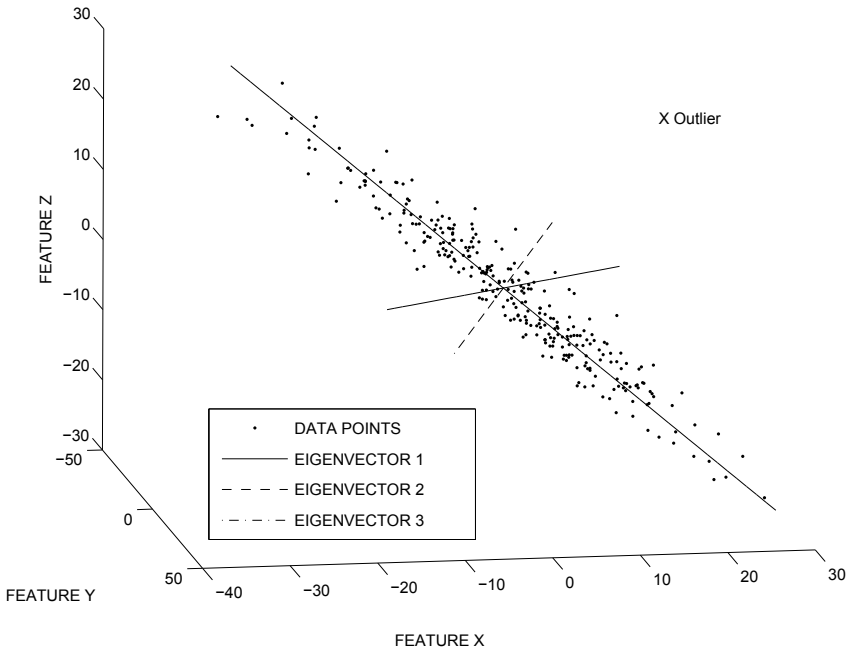
Figure 3.4.   Eigenvectors correspond to directions of correlations in the data. A small number of eigenvectors can capture most of the variance in the data.

observations (as presented in the earlier portions of this chapter). Thus, in the context of a multidimensional data set, the value of d represents the dimensionality, and the value of N represents the number of records (or rows). The i-th record is a row of the multidimensional data set, and is denoted by $R_i = [x_{i1} \ldots x_{id}]$, where x_{ij} is the ith observation for the jth variable X_j. Let us denote the $d \times d$ covariance matrix of the data set by Σ, in which the (i, j)th entry is the covariance between the ith and jth dimensions. This matrix can be shown to be symmetric and positive semi-definite. It can therefore be diagonalized as follows:

$$\Sigma = P \cdot D \cdot P^T$$

Here D is a diagonal matrix, and P is an orthonormal matrix, whose columns correspond to the (orthonormal) eigenvectors of Σ. The corresponding entries in the diagonal matrix D provide the eigenvalues. These orthonormal vectors provides the axes directions along which the data should be projected. The key properties of principal component analysis, which are relevant to outlier analysis, are as follows:

PROPERTY 3.1 (PCA PROPERTIES) *Principal component analysis provides a set of eigenvectors satisfying the following properties:*

- *If the top-k eigenvectors are picked (by largest eigenvalue), then the k-dimensional hyperplane defined by these eigenvectors, and passing through the mean of the data, is a plane for which the mean square distance of all data points to it is as small as possible among all hyperplanes of dimensionality k.*

- *If the data is transformed to the axis-system corresponding to the orthogonal eigenvectors, the variance of the transformed data along each eigenvector dimension is equal to the corresponding eigenvalue. The covariances of the transformed data in this new representation are 0.*

- *Since the variances of the transformed data along the eigenvectors with small eigenvalues are low, significant deviations of the transformed data from the mean values along these directions may represent outliers.*

A formal proof of these properties may be found in [244]. Note that this provides a *much* more general solution than the determination of the optimal coefficients of Equation 3.4. Specifically, the optimal solution for the coefficients of Equation 3.4 may be simply derived as the coefficients of the top *one* eigenvector representing $a_1 \ldots a_d$, and the constant term a_{d+1} may be inferred by substituting the mean of the data in Equation

3.4. On the other hand, the PCA-solution provides a recursive solution of *any* dimensionality by picking the top k eigenvectors.

The data can be transformed to this new axis system, with transformed d-dimensional records denoted by $Y_1 \dots Y_N$. This can be achieved by using the product between the original vector representation R_i and the orthonormal eigenvector matrix P containing the new axis-system:

$$Y_i = [y_{i1} \dots y_{id}] = R_i \cdot P$$

In this new representation, the inter-attribute covariances of Y_i are zero, and most of the variances along the individual attributes correspond to the coordinates along the eigenvectors with the largest eigenvalues. In fact, the eigenvalues represent the variances of the transformed vectors Y_i along these directions in the new coordinate system. For example, if the jth eigenvalue is very small, then the value of y_{ij} in this new transformed representation does not vary much over the different values of i. The beautiful part about PCA is that, in a single shot, it provides all the key directions of *global* correlation, which retain most of the information in the underlying data. These directions are also referred to as the *principal components* in the data, since their second-order correlations are zero, and most of the variance of the data is retained along these directions. In many real scenarios involving very high-dimensional data sets, a very large fraction of the eigenvalues often turn out to be very close to zero. This essentially means that most of the data aligns along a *much lower dimensional subspace.* This is very convenient from the perspective of outlier analysis, because the observations which lie very far away from these directions of projection can be assumed to be outliers. For example, for an eigenvector j which has a small eigenvalue, a large deviation of y_{ij} for the ith record from other values of y_{kj} is indicative of outlier behavior. This is because the values of y_{kj} do not vary much, when j is fixed and k is varied. Therefore, the value y_{ij} is unusual.

The effectiveness of principal component analysis in exposing outliers from the underlying data set can be illustrated with an example. Consider the scatterplot of the 3-dimensional data illustrated in Figure 3.4. In this case, the corresponding eigenvectors have been ordered by decreasing eigenvalues (variances), though this is not immediately obvious from the figure in this 2-d perspective. In this case, the standard deviation along the first eigenvector is three times that along the second eigenvector and nine times that along the third eigenvector. Thus, most of the variance would be captured in the lower-dimensional subspace formed by the top two eigenvectors, though a significant amount of variance would also be captured by picking only the first eigenvector. If the normal distances of the original data points to the 1-dimensional line

corresponding to the first eigenvector (and passing through the mean of the data) are computed, the data point 'X' in the figure would be immediately exposed as an outlier. In the case of high-dimensional data, most of the variance of the data can be captured along a much lower k-dimensional subspace. The residuals for the data points can then be computed by examining the projection distances to this k-dimensional hyperplane passing through the mean of the data points. Data points which have very large distances from this hyperplane can be discarded as outliers. As before, it is possible to model these residuals as a normal distribution, and perform a Z-value test for the corresponding statistical significance.

A more accurate way of modeling the abnormality level without picking any particular set of k dimensions, would be to use the eigenvalue to compute the normalized distance of the data point to the centroid along the direction of *each principal component*. Let $\overline{e_j}$ be the jth eigenvector with a variance (eigenvalue) of λ_j along that direction. The overall normalized outlier score of a data point $\overline{X}$, to the centroid $\overline{\mu}$ of the data is given by the sum of squares of these values:

$$Score(\overline{X}) = \sum_{j=1}^{d} \frac{|(\overline{X} - \overline{\mu}) \cdot \overline{e_j}|^2}{\lambda_j} \tag{3.6}$$

It is important to note that most of the contribution to the outlier score is provided by deviations along the principal component with small values of λ_j, when a data point deviates significantly along such directions. The sum of the squares of these values over all dimensions is a χ^2-distribution with d degrees of freedom. The value of the aggregate residual is compared to the cumulative distribution for the χ^2-distribution in order to determine a probability value for the level of anomalousness. The aforementioned approach was first used in [406].

While it may not be immediately apparent, the score computed above is closely related to the multivariate extreme value analysis method discussed in section 3.4 of Chapter 2. Specifically, the Mahalanobis distance value between $\overline{X}$ and $\overline{\mu}$ computed in that section is *exactly the same*[1] as the score above, except that the eigenvector analysis above provides a better understanding of how this score is decomposed along the different directions of correlation. This decomposition also allows the ability to use only the dimensions with the small eigenvalues in order to obtain an outlier score, which ignores the long eigenvalues. It is possible to use a score which is constructed with only the *smallest* $\delta < d$ eigenvalues.

---

[1]See Exercise 11 of this chapter for the systematic steps.

However, it should also be noted that the approach already performs a kind of soft pruning because of the inverse weighting by the eigenvalues in the score. By explicitly pruning the score, the danger is that if a long eigenvector is relevant to the outlier, then that outlier will be missed. It is not uncommon for a rare value to also align along a long eigenvector. An unusual deviation of a similarly correlated nature in two correlated attributes will cause such a situation. In the event that a pruned score is used, the score may be modeled as a χ^2 distribution with δ degrees of freedom. Therefore, the score may be converted into a probability. This is quite desirable, because it provides a clear idea of the outlierness of the underlying object.

Principal component analysis is much more stable to the presence of a few outliers, than the dependent variable analysis methods. This is because principal component analysis computes the errors with respect to the *optimal hyperplane*, rather than a *particular variable*. When more outliers are added to the data, the optimal hyperplane usually does not change drastically enough to impact the choice of data points which should be considered outliers. Therefore, such an approach is more likely to pick the correct outliers, because the regression model is more accurate to begin with. If desired, this approach can be combined with a sequential ensemble methodology of Chapter 1 in order to determine the outliers robustly. In each iteration, the obvious outliers are removed, and a more refined PCA model is constructed. The final outliers are deviation levels in the last iteration of the sequential ensemble.

3.1   Normalization Issues

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

issue is not unique to linear modeling, and it is often advisable to use such pre-processing for most outlier detection algorithms.

3.2 Applications to Noise Correction

Most of this book is devoted to either removal of outliers as noise, or identification of outliers as anomalies. However, in many applications, it is possible that even though parts of a data record may be erroneous, and may show up as outliers, it may be useful to correct that data record, under the assumption that it should show similarity to the broad patterns in the data. Principal Component Analysis (PCA) provides an approach for achieving this goal. In this case, the core idea of the approach is that *projection of the data point onto the k-dimensional hyperplane corresponding to the largest eigenvalues (and passing through the data mean) provides the optimal correction to the data values.* Obviously such an approach is likely to correct the outlier points significantly more than most of the other normal data points. Some theoretical results (along with experimental evidence) of why such an approach is likely to reduce noise and improve data quality for a variety of applications is provided in [18]. A similar approach to PCA (called *Latent Semantic Indexing*) has also been used in the context of text data, in order to reduce the noise, and significantly improve retrieval quality [133, 355]. In particular, it has been observed in [355] that the use of such dimensionality reduction methods in text data significantly improves the effectiveness of similarity computations, because of the reduction in the noise effects of *synonymy* and *polysemy*. Text representations are inherently noisy because the same word may mean multiple things (synonymy) or the same concept can be represented with multiple words (polysemy). This leads to numerous challenges in virtually all similarity-based applications. The technique of LSI [133] is essentially a variant of PCA, which was originally developed for efficient indexing and retrieval. However, it was eventually observed that the quality of similarity computations, in terms of the underlying precision and recall, actually improves with the use of LSI [355]. This observation was taken to its logical conclusion in [18], where it was theoretically and experimentally shown that significant noise reduction is likely to occur, with the proper use of PCA-based techniques.

An even more effective approach for noise correction is to combine outlier removal and re-insertion with the correction process. The first step is to perform PCA, and remove the top outliers on the basis of a *t*-test with respect to the optimal plane of representation. Subsequently, PCA is performed again on this cleaner data set in order to generate
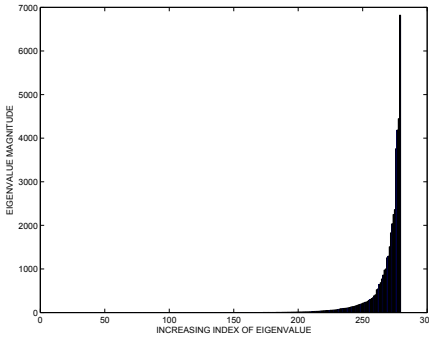
(a) Magnitude of Eigenvalues
(Increasing Index): *Arrythmia*
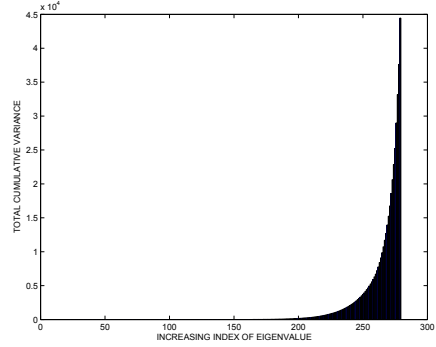
(b) Variance in smallest k
Eigenvalues: *Arrythmia*
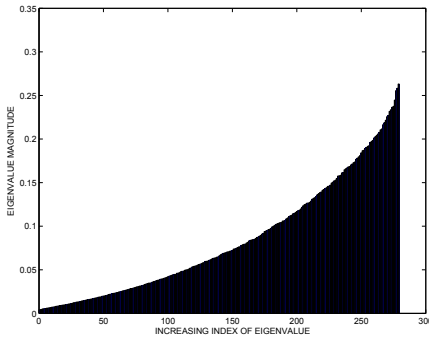
(c) Magnitude of Eigenvalues
(Increasing Index)
Uniform: 452 records only
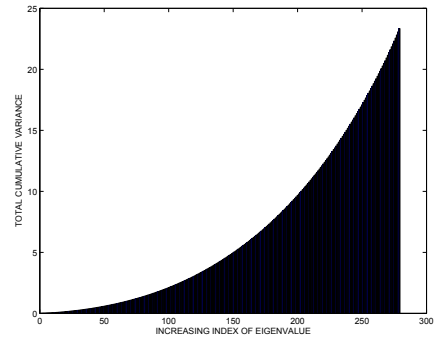
(d) Variance in smallest k
Eigenvalues
Uniform: 452 records only

Figure 3.5. Most of the Energy is Retained in a Small Number of Eigenvalues for the *Arrythmia* data set

the projection subspaces more accurately. The projections can then be performed on this corrected subspace. This process can actually be repeated iteratively, if desired in order to provide further refinement. A number of other approaches to perform regression analysis and outlier removal in a robust way are presented in [387].

3.3   How Many Eigenvectors?

As discussed earlier, the eigenvectors with the largest variance provide the most informative subspaces for data representation, and outlier analysis. In many applications such as noise correction, the data needs to be projected into a subspace of lower dimensionality by picking a specific

number of eigenvectors. Therefore, a natural question arises, as to how the dimensionality k of the projection subspace should be determined.

One observation in most real data sets is that the vast number of eigenvalues are relatively small, and most of the variance is concentrated in a few eigenvectors. An example illustrated in Figure 3.5 shows the behavior of the 279 eigenvectors of the *Arrythmia* data set of the UCI Machine Learning Repository [169]. Figure 3.5(a) shows the absolute magnitude of the eigenvalues in increasing order, whereas Figure 3.5(b) shows the total amount of variance retained in the top-k eigenvalues. In essence, Figure 3.5(b) is derived by using the cumulative sum over the eigenvalues in Figure 3.5(a). While it was argued at the beginning of the chapter that the *Arrythmia* data set is weakly correlated along many of the dimensions, on a pairwise basis, it is interesting to note that that it is still possible[2] to find a small number of directions of global correlation along which most of the variance is retained. In fact, it can be shown that the first 215 eigenvalues (out of 279) *cumulatively* contain less than 1% of the variance in the data set.

In other words, most eigenvalues are very small. Therefore, it pays to retain the eigenvectors corresponding to extremely large values, with respect to the average behavior of the eigenvalues. How to determine, what is "extremely large"? This is a classical case of extreme value analysis methods, which were introduced in Chapter 2. Therefore, each eigenvalue is treated as a data sample, and the statistical modeling is used to determine the large values with the use of hypothesis testing. A challenge in this case is that the sample sizes are small. Even for relatively high dimensional data sets (eg. 50-dimensional data sets), the number of samples (50 different eigenvalues) available for hypothesis testing is relatively small. Therefore, this is a good candidate for the t-value test. The t-value test can be used in conjunction with a particular level of significance and appropriate degrees of freedom in order to determine the number of eigenvectors which should be picked for analysis.

---

[2]Part of the reason for this is that the data set is relatively small with only 452 records. In such cases, it is much easier to find a small number of directions of correlation. As an example, the results of Figure 3.5(c) and (d) show that even for a uniformly distributed data set of the same size, it is possible to find some skews in the eigenvalues. This is one of the limitations of regression analysis, which will be discussed in a later section. Furthermore, the cumulative effects of even weak correlations become magnified with increasing dimensionality, when it is desired to find a much lower dimensional subspace contain the informative projections. This is of course a strength of Principal Component Analysis.

4.   Limitations of Regression Analysis

Regression analysis has a few limitations as a tool for outlier detection. The most significant of these shortcomings was discussed at the very beginning of this chapter, in which the data-specific nature of regression analysis was explored. In particular, the data needs to be highly correlated, and aligned along lower dimensional subspaces, in order for regression analysis techniques to be effective. When the data is uncorrelated, but highly clustered in certain regions, such methods may not work effectively. On the other hand, even when the data is weakly correlated on a pairwise basis between different dimensions, it is often the case that subspaces of much lower dimensionality contain most of the variance in the data, because of the cumulative effect of inter-attribute correlations.

Another related issue is that the correlations in the data may not be global in nature. A number of recent analytical observations [7] have suggested that the subspace correlations are specific to particular localities of the data. In such cases, the global subspaces found by PCA are sub-optimal for outlier analysis. Therefore, it can sometimes be useful to combine linear models with proximity-models (discussed in the next chapter), in order to create more general local subspace models. This will be the topic of high-dimensional and subspace outlier detection, which is discussed in detail in Chapter 5.

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

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

application. This is undesirable, because it is usually interesting to be able to explain *why* a data point is an outlier in terms of the features of the original data space.

Finally, the computational complexity of the approach may be an issue when the dimensionality of the data is large. When the data has dimensionality of d, this results in an $d \times d$ covariance matrix, which may be rather large. Furthermore, the diagonalization of this matrix will slow down at least quadratically with increasing dimensionality. A number of techniques have recently been proposed, which can perform PCA in faster time than quadratic dimensionality [191]. With advances in methods for matrix computation and the increasing power of computer hardware, this issue has ceased to be as much of a problem in recent years. Such dimensionality reduction techniques are now easily applied to large text collections with a dimensionality of several hundreds of thousands of words.

5.   Conclusions and Summary

This chapter presents linear models outlier detection. Many data sets show significant correlations among the different attributes. In such cases, linear modeling may provide an effective tool for removing the outliers from the underlying data. Since linear modeling is a tool in of itself for other regression-based applications, the removal of outliers can be very useful for improving the effectiveness of such applications. In most cases, principal component analysis provides the most effective methods for outlier removal, because it is more robust to the presence of a few outliers in the data. A major limitation of linear modeling is that it does not try to recognize that the correlation behavior of the data in different localities may be different, and tries to fit the data into a single global model. However, it provides a general framework, which can be used for generalized local linear models, which are discussed in Chapter 5.

6.   Bibliographic Survey

The relationships between the problems of regression and outlier detection has been explored extensively in the literature [387]. Outlier analysis is generally seen as an enormous challenge to robust regression in terms of the *noise* effects, and this has motivated an entire book on the subject. In many cases, the presence of outliers may lead to unstable behavior of regression analysis methods. An example of this was illustrated in in Figure 3.3(b) of this chapter, where a single outlier completely changes the regression slope to one which does not reflect the

true behavior of the data. It can be shown that under certain circumstances, a certain number of outliers can have an arbitrarily large effect on the estimation of the regression coefficients. This is also referred to as the *breakdown point* [202, 219] of regression analysis. Such circumstances are very undesirable in outlier analysis, because of the likelihood of very misleading results. Subsequently, numerous estimators have been proposed with higher breakdown points [387]. In such cases, a higher level of contamination would need to be present in the data in order for breakdown to occur.

The method of *Principal Component Analysis* is also used frequently in the classical literature [244] for regression analysis and dimensionality reduction. Its application for noise correction in the text domain was first observed in [355], and then modeled theoretically in [18]. It was shown that the projection of the data points onto the hyper-planes with the greatest variance provides a data representation, with higher quality of similarity computations, because of the effects of removing noise from the data. In the context of text data [355], a variant of PCA, known as Latent Semantic Indexing [133]. Initially, the approach was proposed as a dimensionality reduction technique for retrieval, and was not designed for noise reduction. However, over many years of experience with LSI, it was observed that the quality of retrieval actually improved with LSI, a point which was explicitly pointed out in [355], and later theoretically modeled in [18] for relational data. It should be noted that PCA and LSI are dimensionality reduction techniques which can summarize the data by finding linear correlations among the dimensions. In principle, any dimensionality reduction technique can be used for outlier analysis. An example of an outlier analysis method which uses a different dimensionality reduction technique such as matrix-factorization is discussed in [476]. The core principle is that dimensionality reduction methods provide an approximate representation of the data along with a corresponding set of residuals. These residuals can be used as the outlier scores.

PCA-based techniques have been used in order to detect outliers in a wide variety of domains such as statistics [93], astronomy [147], ecological data [231], network intrusion detection [280, 406, 448], and many kinds of time-series data. Some of the aforementioned applications are temporal, whereas others are not. Because of the relationship between PCA and time series correlation analysis, much of the application of such regression methods has been to the temporal domain. However, it should be emphasized that regression-based methods can also be applied to many non-temporal scenarios. In particular, the use of PCA for non-temporal and unsupervised outlier analysis seems to be relatively

unexplored, and is worthy of further study. Regression based methods will be re-visited in Chapter 8, where a number of methods for temporal outlier analysis will be discussed. In the context of temporal data, the outlier analysis problem is closely related to the problem of *time series forecasting*, where deviations from forecasted values in a time series are flagged as outliers. A variety of regression-based methods for noise reduction and anomaly detection in time-series sensor data streams are also discussed in [19]. In addition, a number of methods which resemble structural and temporal versions of PCA have been used for anomaly detection in graphs [229, 429]. In such methods, an augmented form of the adjacency matrix, or the similarity matrix may be used for eigenvector analysis. Such methods are commonly referred to as *spectral methods*, and are discussed in Chapter 11.

A more general model than global PCA is one in which the data is modeled as a probabilistic mixture of PCAs [451]. This is referred to as *Probabilistic PCA (PPCA)*. Such methods are quite prone to noise in the underlying data during the process of mixture modeling. A method proposed in [132] increases the robustness of PCA by modeling the underlying noise in the form of a student t-distribution. The effect of outliers on PCA-based clustering algorithms are significant. The work in [7] provides a methods for providing the outliers as a side product of the output of the clustering algorithm. Furthermore, methods for using local PCA in outlier analysis will be discussed in detail in Chapter 5 on outlier analysis in high dimensional data.

7.   Exercises

1. Consider the data set of the following observations: { (1, 1), (2, 0.99), (3, 2), (4, 0,98), (5, 0,97) }. Perform a regression with Y as the dependent variable. Then perform a regression with X as the dependent variable. Why are the regression lines so different? Which point should be removed to make the regression lines more similar to one another?

2. Perform Principal Component Analysis on the data set of Exercise 1. Determine the optimal 1-dimensional hyperplane to represent the data. Which data point is furthest from this 1-dimensional plane?

3. Remove the outlier point found in Exercise 2, and perform regression analysis on the remaining four points. Now project the outlier point onto the optimal regression plane. What is the value of the corrected point?

4. Provide a formal derivation for the closed form of the estimates of the regression coefficients in least squares regression. [Hint: Use partial derivatives with respect to regression coefficients.]

5. Provide a formal derivation for the closed form of the optimal k-dimensional subspace in Principal Component Analysis.

6. Download the *KDD CUP 1999 data set* from the UCI Machine Learning Repository [169], and perform PCA on the quantitative attributes. What is the dimensionality of the subspace required to represent (i) 80% of the variance, (ii) 95% of the variance, and (iii) 99% of the variance.

7. Repeat Exercise 6 with the use of the *Arrythmia* data set from the *UCI Machine Learning Repository* [169].

8. Generate 1000 data points randomly in 100-dimensional space, where each dimension is generated from the uniform distribution in $(0, 1)$. Repeat Exercise 6 with this data set. What happens, when you use 1,000,000 data points instead of 1000?

9. Consider a 2-dimensional data set with variables X and Y. Suppose that $Var(X) << Var(Y)$. How does this impact the slope of the X-on-Y regression line, as compared to the slope of the Y-on-X regression lines. Does this provide you with any insights about why one of the regression lines in Figure 3.3(b) shifts significantly compared to that in Figure 3.3(a), because of the addition of an outlier?

10. Scale each dimension of the *Arrythmia* data set, such that the variance of each dimension is 1. Repeat Exercise 7 with the scaled data set. Does the scaling process increase the number of required dimensions, or reduce them? Why? Is there any general inference that you can make about an arbitrary data set from this?

11. Let Σ be the covariance matrix of a data set. Let the Σ be diagonalized as follows:
$$\Sigma = P \cdot D \cdot P^T$$
Here D is a diagonal matrix containing the eigenvalues λ_i, and D^{-1} is also a diagonal matrix containing the inverse of the eigenvalues (i.e. $1/\lambda_i$)

- Show that $\Sigma^{-1} = P \cdot D^{-1} \cdot P^T$
- For a given data point $\overline{X}$ from a data set with mean $\overline{\mu}$, show that the value of the Mahalanobis distance $(\overline{X} - \overline{\mu}) \cdot \Sigma^{-1} \cdot$

$(\overline{X} - \overline{\mu})^T$ between $\overline{X}$ and the mean $\overline{\mu}$ reduces to the same expression as the score in Equation 3.6.

Chapter 4

PROXIMITY-BASED OUTLIER DETECTION

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

1.   Introduction

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

- **Cluster-based:** The non-membership of a data point in any cluster, its distance from other clusters, and the size of the closest cluster, are used as criteria in order to compute the outlier score. The clustering problem has a complementary relationship to the outlier detection problem, in which points either belong to clusters or outliers.

- **Distance-based:** The distance of a data point to its k-nearest neighbor (or other variant) is used in order to define proximity. Data points with large k-nearest neighbor distances are defined as outliers. Distance-based algorithms typically perform the analysis at a much more detailed granularity than the other two methods. On the other hand, this greater granularity often comes at a significant computational cost.

- **Density-based:** The *number* of other points within a specified local region (grid region or distance-based region) of a data point,

is used in order to define local density. These local density values may be converted into outlier scores. Other kernel-based methods or statistical methods for density estimation may also be used. The major difference between clustering and density-based methods is that clustering methods partition the data *points*, whereas density-based methods partition the data *space.*

Clearly, all these techniques are closely related, because they are based on some notion of *proximity* (or similarity). The major difference is at the detailed level of *how* this proximity is defined. These different ways of defining outliers may have different advantages and disadvantages, and this chapter will try to address these issues in a unified way. Furthermore, most of these methods generally work well when the data is highly clustered, and the outliers can be clearly distinguished from dense regions of the data. In many cases, the distinctions between these different classes of methods become blurred, when the definition of sparsity combines[1] more than one of these concepts.

One major difference between distance-based and the other two classes of methods is the level of *granularity* at which the analysis is performed. In both clustering- and density-based methods, the data is pre-aggregated before outlier analysis by either partitioning the points or the space. The data points are compared to the distributions in this pre-aggregated data for analysis. On the other hand, in distance-based methods, the k-nearest neighbor distance to the *original data points* (or a similar variant) is computed as the outlier score. Thus, the analysis in nearest neighbor methods is performed at a much more detailed level of granularity. Correspondingly, these methods provide different tradeoffs between effectiveness and efficiency for data sets of different sizes. Nearest neighbor methods may require $O(N^2)$ time to compute all k-nearest neighbor distances for a data set with N records, unless indexing techniques are used to speed up the computations. Even in those cases, nearest neighbor methods can sometimes be slow, if the underlying data patterns do not support efficient pruning. On the other hand, nearest neighbors can often provide more detailed and accurate analysis, especially for smaller data sets, which may not support robust clustering or density analysis. Thus, the particular choice of the model should depend on the nature of the data and its size. Different methods may be more effective in different scenarios. This relates directly to the

---

[1]It will be discussed later in this chapter, that the well-known LOF method [78] can be interpreted either as a distance-based or density-based method, depending upon how it is presented.

theme repeated throughout the book about the crucial importance of picking the correct data model early in the outlier analysis process.

Proximity-based methods are naturally designed to detect both noise and anomalies, though different methods are suited to these different kinds of outliers. For example, weak definitions of proximal sparsity, such as the non-membership of data points in clusters are naturally designed to detect weak outliers (or noise), whereas large levels of deviation or sparsity in terms of density- or distance-based definitions can also detect strong outliers (or anomalies).

Proximity-based outlier detection methods are extremely popular because of their intuitive simplicity, and their high levels of interpretability. In fact, a number of methods for intuitive exploration and explanation of outliers [262] are based on proximity-centered definitions. Because of the simplicity of the underlying measures, it is possible to design many intuitive variations of these schemes (eg. enhanced local analysis), which provide superior results. Furthermore, as will be evident from the discussion in later chapters, proximity based methods have been generalized to almost all kinds of data such as time-series data, sequence data, or graph data.

This chapter is organized as follows. Section 2 discusses methods for using clusters in outlier analysis. Section 3 discusses distance-based methods for outlier detection. Density-based methods are discussed in Section 4. The limitations of proximity-based outlier detection are discussed in Section 5. Section 6 presents the conclusions and summary.

2. Clusters and Outliers: The Complementary Relationship

Clustering and outlier detection share a well known complementary relationship. A simplistic view would be that every data point, is either a member of a cluster or an outlier. In clustering, the goal is to partition the points into dense subsets, whereas in outlier detection, the goal is to determine points which do not seem to fit naturally in these dense subsets. In fact, most clustering algorithms report outliers as a side-product of their analysis.

However, it is important to understand that outliers which picked purely on the basis of their complementary relationship to clusters are typically *weak* outliers, or noise. This is not necessarily indicative of a true anomaly in the data. This is because non-membership of data points in clusters is a rather blunt hammer to measure the *level of* deviation of a data point from the normal patterns. For example, a data point which is located at the fringes of a large cluster is very different from a point
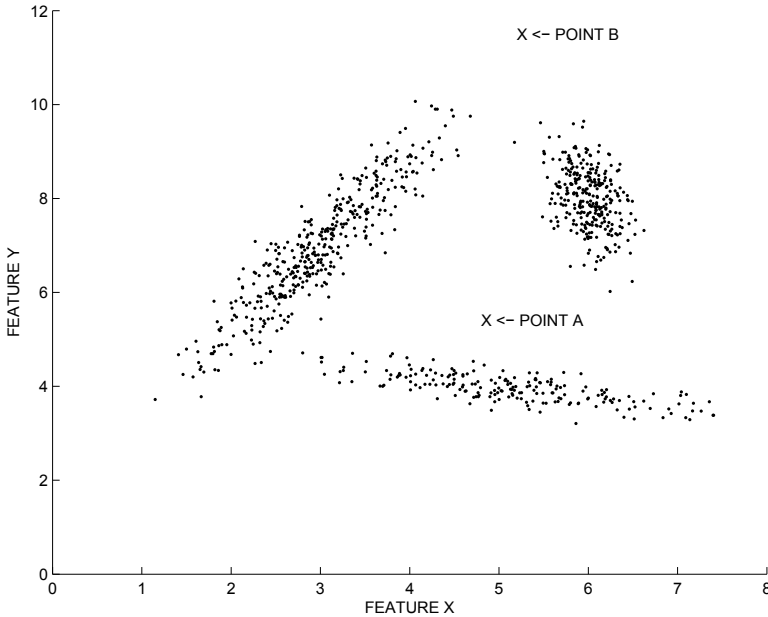
Figure 4.1. The example of Figure 2.7 revisited: Proper distance computations can detect better outliers

which is completely isolated from all the other clusters. Furthermore, *all* data points in very small clusters may sometimes also be considered outliers. Therefore, a much more nuanced measure is often required in order to quantify the outlier score of data points in terms of the clusters in their proximity.

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

$$\mathcal{MB}(\overline{X}, \overline{\mu_r}) = (\overline{X} - \overline{\mu_r}) \cdot \Sigma_r^{-1} \cdot (\overline{X} - \overline{\mu_r})^T$$

Intuitively, this metric scales the square distances by the cluster variances along the different directions of correlation. If the data were to be rotated into a new axis system, which is oriented along the principal components of the cluster, the value of Σ_r would be a diagonal matrix containing the variances along the principal components. Therefore, the individual square distances along the principal components are scaled by the inverse of the variances. This distance scaling process provides much more accurate results. The intuition in using the Mahalanobis distance is really about effective statistical normalization, based on the characteristics of a particular data locality. Even small distances along directions in which cluster variance are small may be *statistically* significant within that data locality. Similarly, large distances along directions in which cluster variances are large may not be statistically significant within that data locality. The Mahalanobis distance uses these normalizations in order to effectively add the contributions of the different dimensions. Note that the normalization is different for different clusters, and it is possible for a data point which is closer to one of the clusters to have a much larger Mahalanobis distance than a data point which is further away on the basis of Euclidean distance. This is evident from the example illustrated in Figure 4.1, in which the data point A is more obviously an outlier than data point B. However, this cannot be detected with the use of the Euclidean distance, according to which the data point A is closest to the nearest cluster centroid.

The Mahalanobis distance can also be used for many distance-based clustering algorithms such as the k-means algorithm, in order to provide more effective results. This is because such a distance computation is sensitive to the different shapes and orientations of the underlying clusters (as in Figure 4.1). In fact, the EM algorithm discussed in Chapter 2 can be considered a soft version of a k-means algorithm [20], when used with the Mahalanobis distance and fixed priors of equal weight. Note that the term in the exponent of the Gaussian distribution for each mixture component of the probabilistic model in Chapter 2 is essentially the Mahalanobis distance. Furthermore, the fit value computed by the EM algorithm, is generally dominated by the exponentiated Mahalanobis distance to the cluster nearest centroid, though other clusters may also have some contributions to the fit value because of the use of soft assignments. The k-means algorithm simply truncates the soft probabilities into a hard assignment. Thus, cluster-based outlier analysis methods are very closely related to the probabilistic mixture models introduced in Chapter 2. Such methods are also closely related to generalized projected clustering methods [7].
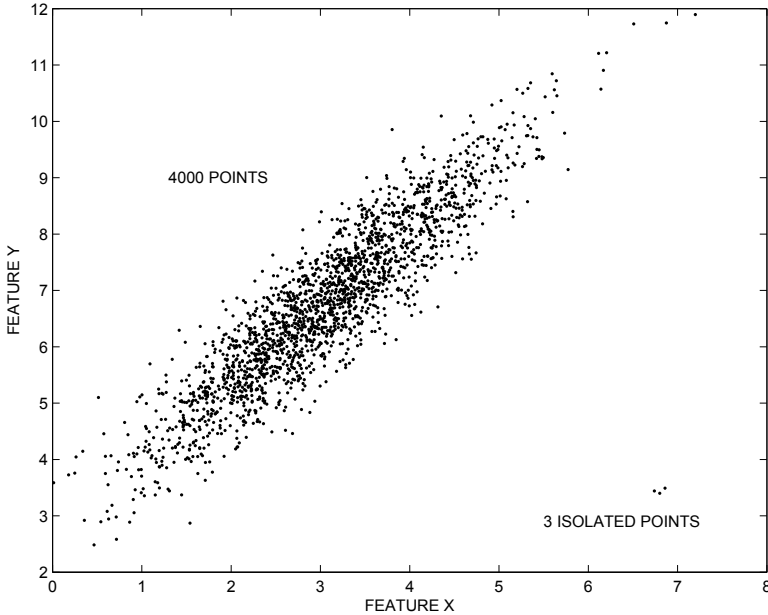
Figure 4.2. The example of Figure 1.5 revisited: Proper combination of global and local analysis in proximity-based methods can identify such outliers

As discussed above, the local Mahalanobis distance of the data point to its closest cluster centroid may be used in order to report the outlier score. Any form of extreme value analysis can be applied to these outlier scores in order to convert the scores to binary labels. It should be mentioned that the effectiveness of the outlier analysis method is usually dependent on that of the clustering method used, because of the complementary relationship between these two problems.

One advantage of clustering methods is that they are based on global analysis of the data, they can determine small closely related groups of data points, and which do not naturally fit with the major patterns in the data. This can be achieved by using a minimum threshold on the number of data points in a cluster. An example is illustrated in Figure 4.2, where the three isolated data points can be identified by any clustering method which has a minimum threshold of four on the number of data points in a cluster. As will be discussed later, some density-based methods which are based purely on local analysis may find it difficult to identify such outliers. Such outliers are common in real applications, because the same (rare) process may generate these outliers multiple times, albeit a small number of times. On the other hand, because clustering methods also ignore the noise in data for calculating deviations

with respect to large cluster centroids, they have a harder time in distinguishing between anomalies and noise. A number of methods can be used in order to improve the effectiveness of cluster-based methods for distinguishing between anomalies and noise.

- The distances of candidate outlier points to cluster centroids should be used, rather than the membership of these points in clusters.

- The distances should be normalized as discussed above with the use of the Mahalanobis method. The local covariance matrix Σ may be constructed from the data points in the corresponding cluster *to which* the distances are being computed. Of course, it is assumed that such a cluster contains a sufficient number of points (at least $(d + 1)$ linearly independent points in dimensionality d) in order to create an invertible covariance matrix.

- The cardinality of the closest clusters should be factored into the outlier score.

- Clusters should have a minimum threshold on their cardinality in order to be considered true clusters, rather than a closely related group of outliers.

These factors can be combined in a wide variety of ways, in order to define different kinds of outlier scores. The bibliographic survey section of this chapter points out some methods used in the literature for combining these different factors.

Cluster-based methods are used often in sparse data domains, in which most attributes take on zero values. In such cases, distance-computations between individual data points are not robust. Therefore, better results are achieved with aggregate cluster representatives in order to perform similarity or distance computations. For example, in the text and market-basket domains, similarity computations between individual text documents may sometimes be quite noisy. Therefore, clustering methods are often used in order to defines outliers and novelties in text and binary market basket data [26, 504, 515]. These methods are discussed in detail in Chapter 7.

On the other hand, in many data domains, cluster analysis may not provide insights at the level of required detail. This is particularly true, when the size of the data set is small. In order to better distinguish between anomalies and noise, it is sometimes necessary to increase the granularity of outlier analysis methods. This can be achieved by using distance computations directly with respect to the original data points, rather than with respect to aggregated representatives such as cluster centroids.
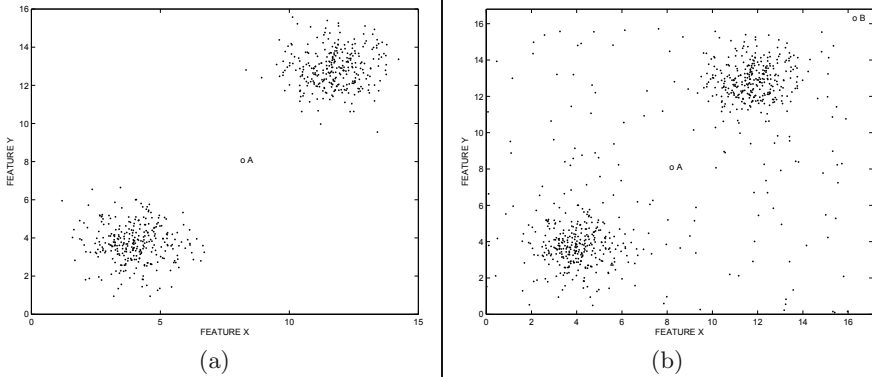
Figure 4.3. The example of Figure 1.1 re-visited: Nearest neighbor algorithms may be more effective than clustering-based algorithms in noisy scenarios because of better granularity of analysis

3.   Distance-based Outlier Analysis

Distance-based methods are a popular class of outlier-detection algorithms across a wide variety of data domains, and define outlier scores on the basis of *nearest neighbor distances*. While this chapter focusses on multidimensional numerical data, such methods have been generalized to a wide variety of other domains such as categorical data, text data, time-series data and sequence data. The later chapters of this book will present distance-based methods for those cases.

Distance-based outlier analysis methods work with the assumption that the k-nearest neighbor distances of outlier data points are much larger than normal data points. Different variations of this definition specify k as an absolute number [381], or as a fraction of the database size [261]. A major difference between clustering and distance-based methods is the *granularity* of the outlier analysis. This can enable a better ability to distinguish between weak and strong outliers in noisy data sets. For example, in the case of Figure 4.3, a clustering-based algorithm will not be able to distinguish between noise and anomalies easily. This is because the distance to the nearest cluster centroid for the data point A will remain the same in Figures 4.3(a) and (b). On the other hand, a k-nearest neighbor algorithm will distinguish between these situations much better because the noisy data points will be included among the distance evaluations, rather than the cluster centroids. Of course, it is also possible to modify cluster-based methods to include the effects of noise. In those cases, the two approaches converge to very similar schemes. One advantage of distance-based methods is that their higher level of granularity in analysis allows effective handling of most tricky

situations. For example, the isolated set of closely related outliers can also be identified by distance-based methods in which an appropriate value of k is used for the k-nearest neighbor analysis. While this can also be identified by clustering methods by setting a threshold on the number of points in each cluster, such points may sometimes bias other cluster representatives.

One of the earliest studies on distance-based outliers was due to Knorr and Ng [261], in which an outlier was defined as follows:

"An object O in a data set T is a $DB(p, D)$ outlier, if at least fraction p of the objects in T lies greater than distance D from O."

This definition is almost[2] identical to the k-nearest neighbor definition, by choosing the value of f to be $(N - k)/N$ for a data set containing N points. Since k is typically much less than N, the value of f needs to be very close to unity in order to obtain more reasonable results. Most distance-based algorithms therefore work with the parameter k, because it is simpler, and more intuitive to understand. Therefore, the discussion in this section will also use the parameter k rather than the fraction f, in order to maintain uniformity of presentation throughout the chapter. The afore-mentioned definition incorporates a distance *threshold* within the definition itself, and therefore returns a binary label, rather than an outlier score.

The simplest approach to the problem uses a *nested loop* approach. In the nested loop approach, two arrays are maintained– the first array contains the candidates for outlier data points, and the other array contains the points to which these candidates are compared in distance-based processing. Once more than k data points have been identified to lie within a distance of D from a point in the first array, that point is automatically marked as a non-outlier. Subsequently, no more time is spent on distance computations involving that data point. Such an approach may require $O(N^2)$ distance computations in the worst case. Since each distance computation may require $O(d)$ time, it follows that the overall running time is $O(N^2 \cdot d)$. Therefore, pruning methods are required in order to speed up the distance computations.

3.1   Cell-based Methods

A second approach which is exponential in the dimensionality but linear in the data points uses a *cell-based* technique. In the cell-based technique, the data space is divided into cells, the width of which is a

---

[2]This definition returns a binary label, whereas most k-nearest neighbor definitions return an outlier score.

L2	L2	L2	L2	L2	L2	L2
L2	L2	L2	L2	L2	L2	L2
L2	L2	L1	L1	L1	L2	L2
L2	L2	L1	★X	L1	L2	L2
L2	L2	L1	L1	L1	L2	L2
L2	L2	L2	L2	L2	L2	L2
L2	L2	L2	L2	L2	L2	L2

Figure 4.4.   Cell-based Partitions of Data Space

function of the threshold D, and the data dimensionality. Specifically, each dimension is divided into cells of width at most $\frac{D}{(2 \cdot \sqrt{d})}$. The presence of data points in a given cell, as well as in adjacent cells satisfies certain properties, which is exploited for more efficient processing. The approach is best explained in the two-dimensional case. Consider the 2-dimensional case, in which successive grid-points are at a distance of at most $D/(2 \cdot \sqrt{2})$. An important point to be kept in mind is that the number of grid-cells is based on a partitioning of the data *space*, and is independent of the number of data points. This is an important factor in the efficiency of the approach for low dimensional data, in which the number of grid-cells is likely to be modest. On the other hand, this approach is not suited to data of higher dimensionality.

For a given cell, its L_1 neighbors are defined to be the set of cells which are reachable from that cell by crossing at most 1 cell-to-cell boundary. Note that two cells touching at a corner are also L_1 neighbors. The L_2 neighbors are those cells which are obtained by crossing either 2 or 3 boundaries. A particular cell marked X, along with its set of L_1 and L_2 neighbors are illustrated in Figure 4.4. It is evident that an interior cell has 8 L_1 neighbors and 40 L_2-neighbors. Then, the following properties can be immediately observed.

1. The distance between a pair of points in a cell is at most $D/2$.

2. The distance between a point, and between a point in its L_1 neighbor is at most D.

3. The distance between a point, and a point in its L_r neighbor (where $r > 2$) is at least D.

The only cells for which immediate conclusions cannot be drawn, are those in L_2. This represents the region of uncertainty for the data points in a particular cell. For those cases, the distance computations need to be performed explicitly. At the same time, a number of rules can be defined in order to immediately declare some fraction of the data points as outliers or non-outliers. These are as follows:

1. If more than k data points are contained in a cell *together with* its L_1 neighbors, then *none* of these data points are outliers.

2. If less than k data points are contained in a cell A, and its L_1 and L_2 neighbors, then *all* points in cell A are outliers.

The first step in this process is to directly label data points as non-outliers, if their cells containing more than k points because of the first rule. Furthermore, all neighbor cells of such cells exclusively contain non-outliers. In order to obtain the full pruning power of the first rule, the sum of the points in each cell and its L_1 neighbors are determined. If the total number is greater than k, then all these points are labeled as non-outliers as well.

Next, the pruning power of the second rule is leveraged. For each cell A containing at least one data point, the sum of the number of points in it, and its L_1 and L_2 neighbors is computed. If this number is no more than k, then all points in cell A are labeled as outliers. At this point, many cells may been labeled as outliers or non-outliers. This provides major pruning gains.

The data points in cells which have not been labeled as either outlier or non-outlier need to have their k-nearest neighbor distance computed explicitly. Even for such data points, the computation of the k-nearest neighbor distances can be made faster with the use of the cell structure. Consider a cell A which has not been labeled as a pure outlier or pure non-outlier cell so far. Such cells may possibly contain a mixture of outliers and non-outliers. The main region of uncertainty for the data points in cell A are the set of points in the L_2 neighbors of this cell A. It cannot be known whether the points in the L_2 neighbors of A are within the threshold distance of D for the points in cell A. Explicit distance computations are required in order to determine the number of points within the threshold D for the data points in cell A. Those data

points for which no more than k points in L_1 and L_2 have distance less than D are declared outliers. Note that distance computations need to be explicitly performed only from points in cell A to the points in the L_2 neighbors of cell A. This is because all points in L_1 neighbors are already known to be at a distance less than D from any point in A, and all points in L_r for $r > 2$ are already known to be at least a distance of D from any point in A. Therefore, an additional level of savings is achieved in the distance computations.

The aforementioned description is for the 2-dimensional case. The approach can also be extended to higher dimensions. The main difference for the d-dimensional case is in terms of the width of a cell (which is now $D/(2 \cdot \sqrt{d})$), and the definition of L_2. In the case of 2-dimensional data, L_2 was defined as the set of cells which were at most 3 cells away, but not an immediate neighbor. In the general case of higher dimensions, L_2 is defined as the set of cells which are at most $\lceil 2 \cdot \sqrt{d} \rceil$ cells away, but not immediate neighbors. All other steps of the algorithm remain identical. However, for the high-dimensional case, this approach becomes increasingly expensive, because the number of cells increases exponentially with data dimensionality. Thus, this approach is generally suited to low-dimensional data.

In many cases, the data sets may not be available in main memory, but may be stored on disk. The data access efficiency therefore becomes a concern. It has been shown in [261] how this approach can be applied to disk-resident data sets with the use of clustered page reads. This algorithm requires at most three passes over the data. More details are available in [261].

3.2   Index-based Methods

The key issue in most distance-based schemes is the potentially large time which is required by the pairwise computations between data points. The cell-based scheme is a special kind of index which provides effective pruning of the distance computations in low dimensionality, with the use of grid-based localization. Indexing and clustering are two other common forms of data localization and access. Therefore, it is natural to explore, whether some of the traditional clustering methods or index structures can be used in order to improve the complexity of distance-based computations.

The work in [381] provides a first approach along this line of methods. The definition of outliers in this work is very similar to that proposed in [261]. The main difference is that instead of defining an *absolute threshold* D on the k-nearest neighbor distance, the points are *ranked*

in decreasing order of the k-nearest neighbor distance. The top n such data points are reported as outliers. Therefore, the threshold is on the distance *rank* rather than the distance *value.* The two definitions are almost equivalent, and are different only in terms of the parameter choice presented to the user. Many variations of this definition are used in the literature in order to report the outlier score. For example, the absolute *distance* to the k-nearest neighbor may be used as the outlier score (rather than the rank), or the *average* distance to *all* the k nearest neighbors may be used.

Let $\delta^k(\overline{X})$ be the k-nearest neighbor distance of the d-dimensional data point $\overline{X} = (x_1 \ldots x_d)$. Therefore, it is desired to determine the top n data points with the largest value of $\delta^k(\overline{X})$. A key technique used in this approach is to approximate sets of points is by their minimum bounding rectangles. This can be used in order to provide upper and lower bounds on the value of $\delta^k(\overline{X})$. Let R be a minimum bounding rectangle, where the lower and upper bounds along the ith dimension are denoted by $[r_i, r_i']$. Then, the minimum distance min_i of x_i along the ith dimension to any point in the minimum bounding rectangle R is potentially 0, if $x_i \in [r_i, r_i']$. Otherwise, the minimum distance is $\min\{|x_i - r_i|, |x_i - r_i'|\}$. Therefore, by computing this minimum value along each dimension, it is possible to estimate the total minimum bound to the entire rectangle R by $\sum_{i=1}^{d} min_i^2$. Similarly, the maximum distance max_i of $\overline{X}$ along the ith dimension to the bounding rectangle R is given by $\max\{|x_i - r_i|, |x_i - r_i'|\}$. The corresponding total maximum value can be estimated as $\sum_{i=1}^{d} max_i^2$. The afore-mentioned bounds can be used in conjunction with index structures such as the R^*-Tree [63] for estimating the k-nearest neighbor distance of data points. This is because such index structures use minimum bounding rectangles in order to represent the data at the nodes. In order to determine the outliers in the data set, the points are processed one by one in order to determine their k-nearest neighbor distances. The highest n such distances are maintained dynamically over the course of the algorithm. A branch-and-bound pruning technique is used on the index structure in order to determine the value of $\delta^k(\overline{X})$ efficiently. When the minimum distance estimate to a bounding rectangle is larger than the value of $\delta^k(\overline{X})$, then the bounding rectangle obviously does not contain any points which would be useful for updating the value of $\delta^k(\overline{X})$. Such subtrees of the R^*-Tree can be completely pruned from consideration.

Aside from the index-based pruning, individual data points can also be discarded from consideration early. A current estimate $Dmin$ on the minimum value of $\delta^k(\overline{X})$ among the best n outliers found so far is maintained. The estimate of $\delta^k(\overline{X})$ for a data point $\overline{X}$ is monotonically

decreasing with algorithm progression, as better nearest neighbors are found. When this estimate falls below $Dmin$, the point $\overline{X}$ can be discarded from consideration, and its k-nearest neighbor distance no longer needs to be estimated more accurately.

Many variations of this broad technique have been proposed in the literature for different data domains. Typically, such algorithms work with a nested loop structure, where the outlier scores of data points are computed one by one *in a heuristically ordered outer loop* which approximates a decreasing level of outlier score. For each point, the nearest neighbors are computed in the inner loop in a *heuristic ordering* which approximates increasing distance to the point. The inner loop can be abandoned, when its currently approximated nearest neighbor distance is less than the nth best outlier found so far ($\delta^k(\overline{X}) < Dmin$).

A good heuristic ordering in the outer and inner loops can ensure that the data point can be discarded from consideration early. The method for finding the heuristic ordering in the outer loop uses the complementarity of the clustering and outlier detection problem, and orders the data points on the basis of the cardinality of the clusters they are contained in. Data points in clusters containing very few (or one) point(s) are examined first. Typically, a very simple and efficient clustering process is used to create the outer loop ordering. The method for finding the heuristic ordering in the inner loop typically requires a fast approximation of the k-nearest neighbor ordering, and is dependent upon the specific data domain or application. An example of such an approach is that of proximity-based outlier detection in time-series [258]. This approach will be discussed in detail in Chapter 8.

3.2.1   Partition-based Speedup.

The approach discussed above may require the reasonably accurate computation of $\delta^k(\overline{X})$ for a large number of points, if the bound estimation process discussed above is not sufficiently robust. This can still be expensive in spite of pruning. In practice, the value of n is quite small, and many data points $\overline{X}$ can be excluded from consideration without estimating $\delta^k(\overline{X})$ explicitly. This is achieved by using clustering [381] in order to perform partitioning of the data space, and then analyzing the data at this level of granularity. A partition-based approach is used to prune away those data points which could not possibly be outliers in a computationally efficient way. This is because the partitioning represents a less granular representation of the data, which can be processed at lower computational costs. For each partition, a lower bound and an upper bound on the k-nearest neighbor distances of *all* included data points is computed. If the upper bound on the k-nearest neighbor distance estimate is less than a current

value of $Dmin$, then the *entire partition* can be pruned from consideration. The partition-based method also provides a more efficient way for approximating $Dmin$. First, the partitions are sorted by decreasing lower bound. The first l partitions containing at least n points are determined. The lower bound on the l-th partition provides an approximation for $Dmin$. The upper and lower bound for each partition is computed using the Minimum Bounding Rectangle of the index structure containing the points. More savings may be obtained by using the fact that the distances from each (unpruned) candidate data point X does not need to be computed to data points in partitions which are guaranteed to be further away than the current upper bound on the k-nearest neighbor distance of the point X (or its containing partition).

Thus, this analysis is performed at a less detailed level of granularity. This makes its efficiency closer to that of clustering-based methods. In fact, the partitions are themselves generated with the use of a clustering algorithm such as BIRCH [505]. Thus, this approach prunes many data points, and then works with a much smaller set of candidate partitions on which the analysis is performed. This greatly improves the efficiency of the approach. The exact details of computing the bounds on the partitions use the afore-mentioned estimations on the minimum and maximum distances to the bounding rectangles of different partitions, and are discussed in detail in [381]. Because of the close relationship between distance-based and clustering methods, it is natural to use clustering methods in order to improve the approximations on the k-nearest neighbor distance. A number of other techniques in the literature use clustering in order to achieve better pruning and speedups in distance-based algorithms [46, 185, 441].

3.3 Reverse Nearest Neighbor Approach

Most of the distance-based methods directly use the k-nearest neighbor distribution in order to define outliers. A different approach is to use the *number of reverse k-nearest neighbors* in order to define outliers [204]. Therefore, the concept of a reverse k-nearest neighbor is first defined.

DEFINITION 4.1 *A data point p is a reverse k-nearest neighbor of q, if and only if q is a k-nearest neighbor of p.*

Data points which have large k-nearest neighbor distances, will also have few reverse neighbors, because they will lie among the k-nearest neighbors of very few data points. Thus, an outlier is defined as a point for which the number of reverse k-nearest neighbors is less than a predefined user-threshold.

Player Name	Short-Handed Goals	Power-Play Goals	Game-Winning Goals	Game-Tying Goals	Games Played
Mario Lemieux	31	8	8	0	70
Jaromir Jagr	20	1	12	1	82
John Leclair	19	0	10	2	82
R. Brind'Amor	4	4	5	4	82

Table 4.1.   Example of Outliers in NHL Player Statistics [262]

The reverse nearest neighbor approach can also be easily understood in terms of the underlying k-nearest neighbor graph. Consider a graph in which the nodes correspond to the data points. A directed edge (p, q) is added to the graph if and only if q is among the k-nearest neighbors of p. Thus, every node has an outdegree of k in this graph. However, the in-degree of the nodes may vary, and is equal to the number of reverse k-nearest neighbors. The nodes with few reverse k-nearest neighbors are declared outliers. This approach also requires the determination of all the k-nearest neighbors of each node. Furthermore, distance-based pruning is no longer possible since the nearest neighbors of each node need to be determined explicitly. Thus, the approach may potentially require $O(N^2)$ time for construction of the k-nearest neighbor graph.

3.4   Intensional Knowledge of Distance-based Outliers

An important issue in outlier analysis is to retain a high level of interpretability for providing intuitive explanations and insights. This is very important in many application-driven scenarios. The concept of *intensional knowledge* of distance-based outliers was first proposed in [262]. The idea is to explain the outlier behavior of objects in terms of subsets of attributes. Thus, in this case, a *minimal* bounding box on the subsets of attributes is presented in order the explain the outlier behavior of the data points. For example, consider the case of NHL player statistics, which was first presented in [262]. An example set of statistics is illustrated in Table 4.1. The sample output from [262], which explains these outliers is as follows:

Mario Lemieux	An outlier in the 1-d space of power play goals An outlier in the 2-d space of short-handed goals and game-winning goals
R. Brind'Amor	An outlier in the 1-d space of game-tying goals.

Several notions are defined in [262] in order to understand the importance of an outlier:

1. Is a particular set of attributes the minimal set of attributes in which an outlier exists?

2. Is an outlier dominated by other outliers in the data?

The intensional knowledge can be directly characterized in terms of cells, which are the bounding rectangles along different attributes. The work in [262] proposes a number of roll-up and drill-down methods in order to define the interesting combinations of attributes for intensional knowledge. The concept of *strong* and *weak* outliers is also defined. Outliers which are defined by minimal combinations of attributes are generally considered stronger from an intensional perspective. It should be emphasized that this definition of strong and weak outliers is specific to an intensional knowledge-based approach, and is different from the more general form in which this book uses these terms (as the outlier tendency of an object).

3.5 Discussion of Distance-based Methods

Distance-based methods have a number of qualitative advantages over clustering-based techniques because of the more detailed granularity of the analysis. For example, distance-based algorithms can distinguish between noise and anomalies much better than cluster-based techniques. Furthermore, distance-based methods can also find isolated groups of outliers just like clustering methods. On the other hand, clustering methods have the advantage that they can provide insights about the *local* distributions of data points for defining distances. For example, in the case of Figure 4.1, the local cluster structure can be used in order to define a *locally sensitive* Mahalanobis distance, which is much more effective at identifying outliers, than a blind application of the euclidian metric. Surprisingly, virtually all the algorithms in the literature use unnormalized euclidian distances, which can lead to results which are not very insightful. While the density-based methods explained later in this chapter do incorporate some notions of locality, they are still unable to
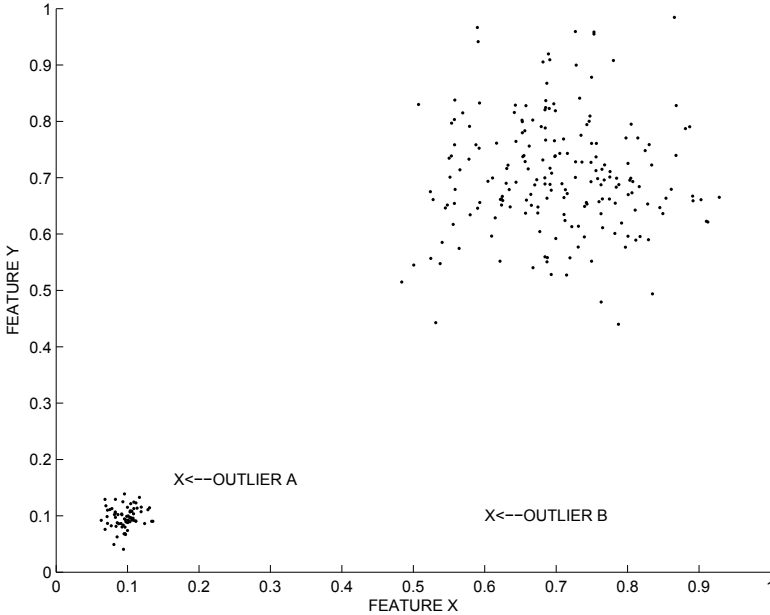
Figure 4.5.   Impact of local density on outliers

provide the detailed level of local insights that an effective combination of a clustering- and distance-based approach can provide. In this context, some recent research has incorporated local clustering insights into distance-based methods. Furthermore, the efficiency advantages of clustering methods should be incorporated into generalized distance-based methods in order to obtain the best results.

4.   Density-based Outliers

The sensitivity of distance-based outliers to data locality was first noticed in [78]. While the Figure 4.1 illustrates the general effect of data locality on both data density and cluster orientation, the work in [78, 79] specifically addresses the issue of varying local density. In order to understand this specific issue, consider the specific example of a data set with varying density in Figure 4.5. The figure contains two outliers labeled A and B. Furthermore, the figure contains two clusters, one of which is much sparser than the other. It is evident that the outlier A cannot be discovered by a distance-based algorithm unless a smaller distance-threshold is used by the algorithm. However, if a smaller distance threshold is used, then many data points in the sparser cluster may be incorrectly declared as outliers. This also means that the

ranking returned by a distance-based algorithm is incorrect when there is significant heterogeneity in the local distributions of the data. This observation was also noted more generally on the basis of the example in Figure 4.1, where it was shown that the outliers are sensitive to both the local cluster density and the orientation. However, much of the work in density-based clustering has generally focussed on issues of varying data density, rather than the varying shape and orientation of clusters. It should also be noted that the reverse-nearest neighbor approach discussed earlier in this chapter can also adjust well to local variations in the underlying data density. However, the issue was first raised explicitly in [78], and subsequently, the local outlier factor approach was proposed. This section contains a discussion of some of the more popular algorithms on density-based outlier analysis.

4.1 LOF: Local Outlier Factor

The *Local Outlier Factor (LOF)* is a quantification of the outlierness of the data points, which is able to adjust for the variations in the different densities. For a given data point $\overline{X}$, let $D^k(\overline{X})$ be its distance to the k-nearest neighbor of X, and let $L_k(\overline{X})$ be the set of points within the k-nearest neighbor distance of $\overline{X}$. Note that $L_k(\overline{X})$ will typically contain k points, but may sometimes contain more than k points because of ties in the k-nearest neighbor distance.

Then, the reachability distance $R_k(\overline{X}, \overline{Y})$ of object $\overline{X}$ with respect to $\overline{Y}$ is defined as the maximum of the distance $dist(\overline{X}, \overline{Y})$, between the pair $(\overline{X}, \overline{Y})$ and the k-nearest neighbor distance of $\overline{Y}$.

$$R_k(\overline{X}, \overline{Y}) = \max\{dist(\overline{X}, \overline{Y}), D^k(\overline{Y})\}$$

The reachability distance is not symmetric between $\overline{X}$ and $\overline{Y}$. Intuitively, when $\overline{Y}$ is in a dense region and the distance between $\overline{X}$ and $\overline{Y}$ is large, the reachability distance of $\overline{X}$ with respect to it is equal to the true distance $dist(\overline{X}, \overline{Y})$. On the other hand, when the distances between $\overline{X}$ and $\overline{Y}$ are small, then the reachability distance is smoothed out by the k-nearest neighbor distance of $\overline{Y}$. The larger the value of k, the greater the smoothing. Correspondingly, the reachability distances with respect to different points will also become more similar.

Then, the *average reachability distance* $AR_k(\overline{X})$ of data point $\overline{X}$ with respect to its neighborhood $L_k(\overline{X})$ is defined as the average of its reachability distances to all objects in its neighborhood.

$$AR_k(\overline{X}) = \text{MEAN}_{\overline{Y} \in L_k(\overline{X})} R_k(\overline{X}, \overline{Y})$$

Here the MEAN function simply represents the mean value over the entire set $L_k(\overline{X})$. The work in [78] also defines the reachability density

as the inverse of this value, though this particular presentation omits this step, since the LOF values can be expressed more simply and intuitively in terms of the average reachability distance $AR_k(\overline{X})$. The *Local Outlier Factor* is then simply equal to the mean ratio of $AR_k(\overline{X})$ to the corresponding values of all points in the k-neighborhood of $\overline{X}$.

$$LOF_k(\overline{X}) = \text{MEAN}_{\overline{Y} \in L_k(\overline{X})} \frac{AR_k(\overline{X})}{AR_k(\overline{Y})}$$

The use of distance ratios in the definition ensures that the local distance behavior is well accounted for in this definition. As a result, the LOF values for the objects in a cluster are often close to 1, when the data points in the cluster are homogeneously distributed. For example, in the case of Figure 4.5, the LOF values of data points in *both* clusters will be quite close to 1, even though the densities of the two clusters are different. On the other hand, the LOF values of *both* the outlying points will be much higher since they will be computed in terms of the ratios to the average neighbor reachability distances. In practice, the maximum value of $LOF_k(\overline{X})$ over a range of different values of k is used as the outlier score in order to rank the different objects [78].

One observation about the LOF method is that while it is popularly understood in the literature as a density-based approach, it can be more simply understood as a *relative* distance-based approach with smoothing. The relative distances are computed on the basis of the local distribution of reachability distances. The LOF method was originally presented in [78] as a density-based approach because of its ability to adjust to regions of varying density. The density is loosely defined as the inverse of the average of the smoothed reachability distances in a neighborhood according to [78]. This is of course not a precise definition of density, which is traditionally defined in terms of the number of data points within a specified area or volume. The presentation in this chapter omits this intermediate density variable, both for simplicity, and for a definition of LOF directly in terms of reachability distances. The real connection of LOF to data density lies in its insightful ability to *adjust* to varying data density with the use of *relative distances*. While this book has also classified this method as a density-based approach, it can be equivalently understood in terms of either a relaxed definition of density or distances.

4.2   LOCI: Local Correlation Integral

An interesting method proposed in [356] uses a local density-based method for outlier analysis. The LOCI method is truly a density-based method, since it defines the density $M(\overline{X}, \epsilon)$ of a data point $\overline{X}$ in terms

of the number of data points within a pre-specified radius ϵ around a point. This is referred to as the *counting neighborhood* of the data point $\overline{X}$. Correspondingly, the average density $AM(\overline{X}, \epsilon, \delta)$ in the δ-neighborhood of $\overline{X}$ is defined as the mean value of $M(\overline{X}, \epsilon)$ for all data points at a distance at most δ from $\overline{X}$. The value of δ is also referred to as the *sampling* neighborhood of $\overline{X}$, and is always larger than ϵ. Furthermore, the value of ϵ is always chosen as a constant fraction of δ, no matter what value of δ is used. The value of δ is a critical parameter in the analysis, and multiple values of this parameter are used in order to provide analytical insights at different levels of granularity. The average density is formally defined as follows:

$$AM(\overline{X}, \epsilon, \delta) = \text{MEAN}_{(\overline{Y}:dist(\overline{X}, \overline{Y}) \leq \delta)} M(\overline{Y}, \epsilon)$$

Correspondingly, the *multi-granularity deviation factor $MDEF(\overline{X}, \epsilon, \delta)$* at level δ is expressed in terms of the ratio of the densities at a point, and its neighborhood.

$$MDEF(\overline{X}, \epsilon, \delta) = 1 - \frac{M(\overline{X}, \epsilon)}{AM(\overline{X}, \epsilon, \delta)} \qquad (4.1)$$

Note the similarity to LOF in terms of using the local ratios while defining the outlier score of a data point. The larger the value of the MDEF, the greater the outlier score. In order to convert the MDEF score into a binary label, the deviation $\sigma(\overline{X}, \epsilon, \delta)$ of the different values of $M(\overline{X}, \epsilon)$ within the sampling neighborhood of $\overline{X}$ is computed.

$$\sigma(\overline{X}, \epsilon, \delta) = \frac{STD_{(\overline{Y}:dist(\overline{X}, \overline{Y}) \leq \delta)} M(\overline{Y}, \epsilon)}{AM(\overline{X}, \epsilon, \delta)}$$

Here the term STD corresponds to the standard-deviation function computed over the entire sampling neighborhood. The term in the denominator of the standard deviation accounts for the fact that the MDEF value of Equation 4.1 is scaled by the same value in the denominator.

The value of ϵ is always chosen to be half that of δ in order to enable fast approximate computation. Therefore, throughout this presentation, it is assumed that the value of ϵ is automatically decided by the choice of δ. Multiple values of δ are used in order to provide a multi-granularity approach for outlier analysis. These methods vary the sampling radius from a minimum radius containing at least 20 points to a maximum radius which spans most of the data. A data point is an outlier if its MDEF value is unusually large among *any* of the values computed at different granularity levels. Specifically, the value of the MDEF needs

to be at least $k \cdot \sigma(\overline{X}, \epsilon, \delta)$, where k is chosen to be 3. This choice of k is common in statistical analysis with the use of the normal distribution assumption.

The algorithm can be made much faster in practice with the use of several observations:

- Only a limited set of sampling neighborhoods need to be considered. In particular, if the sampling or counting neighborhoods do not change for small changes in δ, then those neighborhoods do not need to be considered.

- Fast ways to approximate the neighbor counts are also provided in [356]. This provides good approximations to MDEF, which are usually acceptable in practice. It has been shown in [356], that a box count of a grid-based division of the data provides a fast approximation, when L_∞ distances are used. This approximation is also referred to as the aLOCI algorithm.

4.2.1   LOCI Plot.   The LOCI plot compresses the information about a data point in a two dimensional representation, where the outlier behavior is visually interpretable from a multi-granular perspective. Since the value of $MDEF(\overline{X}, \epsilon, \delta)$ is constructed by examining the relative behavior of $M(\overline{X}, \epsilon)$ and $AM(\overline{X}, \epsilon, \delta)$ over different values of δ, it makes sense to visualize each of these quantities by separately plotting them against the sampling neighborhood δ. Therefore, the LOCI plot shows the value of δ on the X-axis, against each of the following two count-based quantities on the Y-axis:

- The value of $M(\overline{X}, \epsilon) = M(\overline{X}, \delta/2)$ is plotted on the Y-axis. This shows the actual density behavior of the data point $\overline{X}$ at different granularity levels.

- The values of $AM(\overline{X}, \epsilon, \delta) \pm STD_{(\overline{Y}:dist(\overline{X}, \overline{Y}) \leq \delta)} M(\overline{Y}, \epsilon)$ are plotted on the Y-axis. This shows the density behavior of the neighborhood of $\overline{X}$ (along with statistical ranges) for different granularity levels.

The LOCI plot provides a *visual* understanding of how the deviations of the data point relate to extreme values of the deviation at different granularity levels, and it *explains* why a particular data point may have a high MDEF value. The use of different granularity levels is certainly an advantage, because it can detect outliers in a very general way, which is data independent. For example, in the case of Figure 4.2, any distance-based or the LOF method would need to pick the value of k (for k-nearest

neighbor) very carefully in order to identify these data points as outliers. However, the LOCI method would always be able to find some level of granularity at which these data points are declared outliers. This would also show up in the LOCI plot at the corresponding granularity level.

4.3   Histogram-based Techniques

Histograms are a simple and intuitive construct, which are particularly suitable for density-based summarization of univariate data. In this case, the data is discretized into bins, and the frequency of each bin is estimated. Data points which lie in bins with very low frequency are reported as outliers. In the context of multivariate data, the approach can be generalized in two different ways:

- The outlier scores are computed separately for each dimension, and then the scores can be aggregated.

- The discretization along each dimension can be generated at the same time, and a grid structure can be constructed. The distribution of the points in the grid structure can be used in order to create a model of the sparse regions. The data points in these sparse regions are the outliers.

Let $f_1 \ldots f_k$ be the frequencies of the k univariate or multivariate bins which are constructed. Then, the mean and standard-deviation of these frequencies can be estimated. A student t-distribution or normal distribution can be used in order to model the bin frequencies. This model can be used to determine those bins which have unusually low frequency. The data points in the bins with unusually low frequency are declared outliers. If desired, the frequency of a bin is reduced by 1, in order to model the anomalousness of a data point, without including it in the count. This is because the inclusion of the data point itself in the count can mask its outlier score. The quality of the histogram profiles can be further improved by using a sequential ensemble method in which obvious outliers are moved from the data in a first step, and then the histograms are built on the pruned data containing fewer outliers. This model is more robust, since it discounts the impact of outliers on the model at least to a partial degree.

The major challenge with histogram-based techniques is that it is often hard to determine optimal histogram width well. Histograms which are too wide or too narrow will not model the frequency distribution at the level of granularity needed to optimally detect outliers. When the bins are too narrow, the normal data points falling in these bins will be declared outliers. On the other hand, when the bins are too wide,

anomalous data points may fall in high frequency bins, and will therefore not be declared outliers.

A second challenge with the use of histogram techniques is that they are too local in nature, and often do not take the global characteristics of the data into account. For example, for the case of Figure 4.2, a multivariate grid-based approach may not be able to classify an isolated group of data points as outliers, unless the resolution of the grid structure is calibrated carefully. This is because the density of the grid only depends on the data points inside it, and an isolated group of points may create an artificially dense grid cell, when the granularity of representation is high. Histogram methods do not work very well in higher dimensionality because of the sparsity of the grid structure with increasing dimensionality, unless the outlier score is computed with respect to carefully chosen lower dimensional projections. For example, a d-dimensional space will contain at least 2^d grid-cells, and therefore, the number of data points expected to populate each cell reduces exponentially with increasing dimensionality. Nevertheless, histogram-based techniques find wide applicability in intrusion-detection techniques, because such applications are naturally suited to modeling the normal data with the use of histogram-based profiles.

4.4   Kernel Density Estimation

Kernel-density estimation methods are similar to histogram techniques in terms of building density profiles, though the major differences is that a smoother version of the density profile is constructed. In kernel density estimation [409], a continuous estimate of the density is generated at a given point. The value of the density at a given point is estimated as the sum of the smoothed values of kernel functions $K_h'(\cdot)$ associated with each point in the data set. Each kernel function is associated with a kernel width h 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:

$$\overline{f}(\overline{x}) = (1/N) \cdot \sum_{i=1}^{N} K_h'(\overline{x} - \overline{X_i}) \tag{4.2}$$

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.

$$K_h'(x - X_i) = (1/\sqrt{2\pi} \cdot h) \cdot e^{-(x-X_i)^2/(2h^2)}$$

The estimation error is defined by the kernel width h which is chosen in a data driven manner. It has been shown [409] that for most smooth functions $K'_h(\cdot)$, when the number of data points goes to infinity, the estimator $\overline{f}(x)$ asymptotically converges to the true density function $f(x)$, provided that the width h is chosen appropriately. For the d-dimensional case, the kernel function is chosen to be the product of d identical kernels $K_i(\cdot)$, each with its own smoothing parameter h_i.

As before, the mean and standard deviations of the data density at each of the points can be constructed. The data points with unusually low density are declared outliers with the use of a t-distribution or normal distribution assumption. If desired, the density-contribution of the data point itself can be excluded in order to determine its outlier score. As in the case of histogram-based methods, sequential ensembles can be used in order to improve the robustness of the model.

Density-based methods have similar challenges as histogram-techniques. In particular, the use of a global bandwidth in order to estimate density may not work very well in cases where there are wide variations in local density such as Figures 4.2 and 4.5. Furthermore, these methods are not very effective for higher dimensionality, because the accuracy of the density estimation process degrades with increasing dimensionality.

5.   Limitations of Proximity-based Detection

Most proximity-based methods use distances in order to define outliers at varying levels of granularity. Typically, higher levels of granularity are required for greater accuracy. In particular, methods which abstract the data by various forms of summarization do not distinguish well between true anomalies and noisy regions of low density. Furthermore, these methods need to combine global and local analysis carefully in order to find the true outliers in the data. A fully global analysis may miss important outliers as indicated in Figures 4.1 and 4.5, whereas a fully local analysis may miss small clustered groups of outliers as illustrated in Figure 4.2. At the same time, increasing the granularity of analysis can make the algorithms inefficient. In the worst-case, a distance-based algorithm with full granularity can require $O(N^2)$ distance computations in a data set containing N records. While indexing methods can be used in order to incorporate pruning into the outlier search, the effectiveness of pruning methods reduces with increasing dimensionality because of data sparsity.

An even more fundamental limitation in the context of high dimensional data is not one of *efficiency*, but that of the *quality* of the outliers found. In the high-dimensional case, all points become almost equidis-

tant from one another, and therefore the contrast in the distances is lost [22, 215]. This is also referred to as the curse of dimensionality, which arises from data sparsity, and it negatively impacts many high dimensional applications [8]. With increasing dimensionality, most of the features are not informative for outlier detection, and the noise effects of these dimensions will impact proximity-based methods in a very negative way. In such cases, the outliers can be masked by the noise in the features, unless the relevant dimensions can be explicitly discovered by an outlier detection method. Since proximity-based methods are naturally designed to use all the features in the data, their quality will naturally degrade with increasing dimensionality. Some methods do exist for improving the effectiveness of such methods in increasing dimensionality with subspace methods. These methods will be discussed in Chapter 5.

6.   Conclusions and Summary

This chapter provides an overview of the key proximity-based techniques for outlier analysis. All these methods determine the outliers in the data with the use of proximity information between data points. These methods are closely related to clustering techniques in a complementary sense; while the former finds outlier points in sparse data localities, the latter tries to determine dense data localities. Therefore, clustering is itself a common method used in proximity-based outlier analysis.

Proximity-based methods enjoy wide popularity in the literature because of ease of implementation and interpretability. A major challenge in using such methods is that they are typically computationally intensive, and most of the high-quality methods require $O(N^2)$ distance computations in the worst-case. Furthermore, these methods may not work very effectively, when all the dimensions are used for the analysis. This is because the anomalies in the data are often lost in the noise of full dimensional analysis. Methods for finding such outliers will be discussed in detail in the next chapter.

7.   Bibliographic Survey

The traditional definition of multivariate outliers was often understood in the context of side-products of clustering algorithms. Outliers were therefore defined as data points which do not naturally fit into any cluster. However, the non-membership of a data point in a cluster is not able to distinguish between noise and anomalies. A detailed discussion of different clustering algorithms, and their use for outlier analysis may

be found in [232, 255]. Many of the clustering algorithms explicitly label points which do not fit within clusters as outliers [492]. However, in some sparse high-dimensional domains such as transactional data, (subspace)-clustering may be the only approach which can be used for determining outliers [209]. This method is discussed in detail in the chapter on categorical data.

In order to improve the accuracy of a clustering approach further, one can use the distance of data points to cluster centroids, rather than using only the membership of data points in clusters. The work in [412] investigates a number of deterministic and probabilistic methods for clustering in order to detect anomalies. These techniques were designed in the context of intrusion detection. One challenge in such methods is to prevent the clustering methods from quality-degradation by noise and anomalies, which are already present in the data. This is because if the clusters which are found are already biased by noise and anomalies, it will also prevent outliers from being found effectively. Such techniques have been used often in the context of intrusion-detection applications [55, 412]. The work in [55] uses a first phase in which the normal data is identified, by using data points matching frequent patterns in the data. Subsequently this normal data is used in order to perform robust clustering. The outliers are then determined as points which lie at a significant distance to these clusters. These methods can be considered a kind of sequential ensemble approach.

A number of outlier detection methods have also been proposed for cases where the anomalies may lie in small clusters [155, 371, 372, 351, 208, 239]. Many of these techniques work by using distance-thresholding in order to regulate the creation of new clusters. When a data point does not lie within a specified threshold of the nearest cluster centroid, a new cluster is created containing a single instance. This results in clusters of varying size, since some of the newly created clusters do not get a sufficient number of points added to them. Then, the outlierness of a data point may be decided both by the number of points in its cluster, and the distance of its cluster to the other clusters. A number of indexing techniques have also been proposed in order to speed to the partitioning of the data points into clusters [98, 428]. Biased sampling [265] has also been shown to be an effective and efficient method for clustering-based outlier detection.

Distance-based methods have been extremely popular in the literature because of their ability to perform the analysis at a higher level of granularity than clustering methods. Furthermore, such methods are intuitive and extremely easy to understand and implement. The first distance-based method was proposed in [261]. The ideas in this work

were extended to finding intensional knowledge in [262]. Subsequently, indexing methods were designed to improve the efficiency of this method in [381]. Subsequently, a significant number of distance-based algorithms have been developed for different scenarios. The work in [46] uses linearization in which the multidimensional space is populated with Hilbert space-filling curves. This 1-d representation has the advantage that the k-nearest neighbors can be determined very *fast* by examining the predecessors and successors of a data point on the space-filling curve. The sum of the k-nearest neighbor distances on the linearized representation is used in order to generate the outlier score of a data object. While the use of the *sum* of the k-nearest neighbor distances has some advantages over the k-nearest neighbor distance in differentiating between sparsely populated data and clustered data, it has the disadvantage of (sometimes) not being able to detect groups of isolated anomalies as illustrated in Figure 4.2. One challenge with the use of space filling curves is that since they map the data into a hypercube in d-dimensions, the number of corners of this hypercube increase exponentially with dimensionality. In such cases, the sparsity of the data in high-dimensions may result in a degradation of the locality behavior of the space-filling curve with increasing dimensionality. In order to address this issue, the work in [46] uses data shifting techniques in order to improve locality. An iterative technique was designed, which requires $d + 1$ scans of the data set.

The work in [60] designs a simple pruning technique in order to improve the efficiency of a k-nearest neighbor based outlier detection technique. The core idea is similar to the pruning rule used in [381]. The idea is that if the outlier score for an object is less than the k-nearest neighbor distance of the n-th outlier, then that data point cannot possibly be an outlier and is pruned from further consideration. This simple pruning rule has been shown in [60] to work well with randomized data. The randomization itself can be done in linear time by using a disk-based shuffling technique. The work in [471] performs the nearest neighbor computations on a smaller sample of the data set in order to improve the efficiency. Theoretical guarantees are provided in order to bound the loss in accuracy resulting from the sampling process.

The effectiveness of pruning methods is clearly dependent on the ability to generate a good bound on the k-nearest neighbor distances in an efficient way. Therefore, the work in [185] partitions the data into small clusters. The k-nearest neighbor distance of a data point within a cluster is used in order to generate an upper bound on the k-nearest neighbor distance of that point. If this upper bound is less than the scores of the set of outliers already found, then the point can be pruned from consid-

eration. The work in [381] also uses clustering techniques for pruning, though the method in [185] uses recursive hierarchical partitioning in order to ensure that each cluster is assigned a similar number of data points. The ordering of the data points along the principal component of largest variance is used in order to provide a quick estimate of the k-nearest neighbor distance.

A resolution-based method was proposed in [157]. According to this method, whether a point belongs to a cluster or whether it is an outlier depends on the distance threshold. At the highest resolution level, all points are in individual clusters of their own and are therefore outliers. As the resolution is slowly reduced, more and more data points join clusters. In each step each point changes from being an outlier to a cluster. Based on this, a *Resolution Outlier Factor (ROF)* value was defined in [157]. This was shown to provide effective results for outlier analysis.

Most of the distance-based algorithms are designed with the use of euclidian distances. In practice, the euclidian function may not be optimal for finding the outliers. In fact, for many other domains of data, the distance functions are often defined in a fairly complex way, and many of the pruning techniques designed for euclidian spaces will not work well in arbitrary spaces. In this context, an efficient algorithm was designed for outlier detection in arbitrary metric spaces [441], which requires at most three passes over the data.

A method to improve the efficiency of distance-based algorithms with the use of reference points was proposed in [359]. The core idea in this work is to rank the data points on the basis of their relative degree of density with respect to a fixed set of R reference points. Each data point is transformed to a 1-dimensional space in R possible ways on the basis of their distance to a reference point. For each of these R 1-dimensional data sets, the relative degree of density of each data point with respect to the corresponding reference point is computed. The overall relative degree of density of a data point is defined as the minimum relative degree of density over all the reference points. This relative degree of density provides a way to rank the different data points. Distributed algorithms for speeding up outlier detection are proposed in [66].

Scalability is a significant issue in the context of the data streams. Typically, in the case of data streams, a past window of history is used in order to determine outliers. Data points whose k-nearest neighbor values are large in a specific sliding window history are declared outliers [48, 266]. Stream clustering methods such as those in [25] can also be used in order to speed up the outlier analysis process. Such an approach has been discussed in [266].

The issue of local density in the context of outlier analysis was first addressed in [78, 79]. We note that the reverse nearest neighbor approach [81, 204] presented in this chapter shares some similarities to LOF in terms of adjusting to local densities with the use of a reverse nearest-neighbor approach. The variations in the local density may result in poor ranking of outliers by global distance-based methods. Therefore, the concept of *Local Outlier Factor (LOF)* was proposed in [78]. These methods adjust the outlier score of an object on the basis of the local density. It should be mentioned that the concept of density is really loosely defined in LOF as an inverse of averaged distances. A true definition of density should really count the number of data points in a specific volume. Data points in local regions of high density are given a higher outlier score, even if they are slightly isolated from the other points in their locality. Many different variants of the broad LOF approach were subsequently proposed. For example, the work in [241] proposed the concept of top-n local outliers, where the top-n outliers were determined on the basis of the density. Pruning techniques were used to improve the running time. This was achieved by partitioning the data into clusters, and computing bounds on the LOF values of the points in each cluster. Thus, entire clusters can be pruned if they are guaranteed to contain only points, which have lower LOF values than the weakest of the current top-n outliers. Other methods for improving the effectiveness of top-n local outlier detection with the use of cluster-pruning were proposed in [117].

One issue with LOF techniques is that they can sometimes be ineffective, when regions of different density are not clearly separated. Therefore, the INFLO technique of [242] takes the symmetric neighbor relationship into account while defining the local outliers. The concept of *connectivity-based outlier factor (COF)* was proposed in [442], which is also able to find outliers in low density or arbitrarily shaped regions effectively. The main difference from LOF is the way in which the neighborhood of a data point is defined. Specifically, the neighborhood is defined incrementally by adding the closest point to the current neighborhood set. This can define neighborhoods effectively when the points are distributed on arbitrary lower dimensional manifolds of the data. The LOF approach has also been combined with other clustering techniques. For example, the work in [208, 210] defines a score called *Cluster-Based Local Outlier Factor (CBLOF)* in which anomalies are defined as a combination of local distances to nearby clusters and the size of the clusters to which the data point belongs. Data points in small clusters, which are at a large distance to nearby clusters are flagged as outliers.

The LOF scheme has also been extended to the case of spatial data with non-spatial attributes [433]. For example, sea surface temperatures can be considered as a kind of non-spatial attribute in the context of spatial location. Such data are known to exhibit *spatial auto-correlations*, in which the value of an element is affected by its immediate neighbors (eg. spatial temperature locality). Furthermore, the data shows *spatial heteroscedasticity*, in which the variance of a data point is based on its location. For example, "normal" temperature variations are clearly based on geographical location. We note that spatial data shares some similarities with temporal data from the perspective of *spatial continuity*, which is analogous to *temporal continuity*. Correspondingly, the work in [433] defines a local outlier measure, known as the *Spatial Local Outlier Measure (SLOM)*, which is specially suited to spatial outlier detection. The generalization of the LOF method to the streaming scenario is discussed in [369].

The LOCI method [356] is also a locally sensitive method, which uses the number of points in a circular neighborhood around a point, rather than the inverse of the k-nearest neighbor distances for local density computation. Thus, it is truly a density based method from an intuitive perspective. Furthermore, the approach is tested over different levels of granularity in order to reduce the parameter choices, and remove the need for *some* of the input parameters during the outlier detection process. An approximate version of the algorithm can be implemented in almost linear time. An interesting contribution of this work is the introduction of LOCI plots, which provide an intuitive understanding of the outliers in the data with a visual plot. The LOCI plot provides an understanding of how different sizes of neighborhoods may correspond to the outlier score of a data point.

The traditional methods for density-based outlier analysis involve the use of discretization, grid-based methods, and kernel-density based methods. The first two belong in the general category of histogram-based methods [236]. The main challenge in histogram-based methods is that the bucket-size along each dimension can sometimes be hard to pick correctly. Ideally, histograms should be built only with normal data (without anomalies), in order to obtain the best results. However, they can also be used with a mixture of normal and anomalous instances. In such cases, a given point should be removed from contention while evaluating the statistical frequency of the histogram bin that it belongs to. These methods are also hard to use in the high dimensional case, because the grids can become increasingly sparse with increasing dimensionality. A closely related method is that of kernel-density estimation [409]. Kernel density-estimation is a continuous variation of grid-based methods, in

which a smooth kernel function is used for the estimation process. Such methods also become increasingly susceptible in greater dimensionality, because density estimation cannot be robustly performed as the dimensionality of the data increases. This is consistent with the behavior of all proximity-based methods, because the concept of proximity is poorly defined with increasing dimensionality [215].

8.   Exercises

1. Consider a data set with the following observations: { (1, 3), (1.01, 3.01), (0.99, 3.01), (0.99, 3), (0.99, 2.99), (3, 1) }.

   - What are the results of linear modeling on this data set onto 1-dimensions (PCA), for finding outliers?
   - How well does a 1-NN technique work for finding outliers in this case? How do the absolute values of the outlier scores compare in the two cases?

2. Consider a data set containing a single cluster with the points { (1, 1), (0, 0), (2, 2.1), (3, 3.1), (4, 4), (5.1, 5) }.

   - Consider the two points $(6.5, 6.5)$, and $(1, 2.1)$. Draw the points on a piece of paper. Which of the two data points seems more like an outlier?
   - Which point does a 1-NN algorithm set as the highest outlier score with the euclidian metric?
   - Which point does a 1-NN algorithm set as the lowest outlier score with the euclidian metric?
   - Which data point does a PCA-based algorithm set at the highest outlier rank, when the residuals are the outlier scores?
   - Would you recommend changing the distance function from the euclidian metric? How?

3. Download the Ionosphere data set from the UCI machine learning repository [169].

   - Rank the data points based on their residual scores in a PCA approach, when only the top 3 eigenvectors are used.
   - Rank the data points based on their k-nearest neighbor scores, for values of k ranging from 1 through 5.
   - Normalize the data, so that the variance along each dimension is 1. Rank the data points based on their k-nearest neighbor scores, for values of k ranging from 1 through 5.

- How many data points are common among the top 5 ranked outliers using different methods?

- Now use a voting scheme, which adds up the ranks of the outliers in different schemes. Which are the top 5 outliers now?

- Does this ensemble approach provide more robust outliers?

4. Repeat Exercise 3 with the network intrusion data set from the UCI machine learning repository.

5. A manufacturing company produces 2-dimensional square widgets, which are normally distributed with a length of 1 meter on each side, and a standard deviation of 0.01 meters.

- Generate a data set with 100,000 widgets from this distribution.

- The company produced 5 anomalous widgets, due a defect in the manufacturing process. Each such widget had a square length of 0.1 meters, and standard deviation of 0.001 meters. Generate these 5 anomalous points using the normal distribution assumption.

- Does a 1-NN approach find the anomalous widgets?

- Does a 10-NN approach find the anomalous widgets?

6. Apply a k-means clustering approach to the data set in Exercise 5, where 5 cluster centroids are used. As a post-processing step, remove any clusters with 10 or less data points. Score the data points by their distance to their closest cluster centroids. Which data points have the highest outlier scores?

7. Apply the reverse 1-NN algorithm to the case of Exercise 5. Which data points have the highest outlier scores? Which data points have the highest outlier scores with the reverse 10-NN algorithm? With the reverse 100-NN algorithm?

8. Repeat Exercises 3 and 4 with the use of the LOF method and determine the ranking of the outliers. Are the outliers same in this case as those found in Exercises 3 and 4?

9. Repeat Exercise 8 with the use of the LOCI method. Are the outliers found to be the same?

Chapter 5

HIGH-DIMENSIONAL OUTLIER DETECTION: THE SUBSPACE METHOD

"In view of all that we have said in the foregoing sections, the many obstacles we appear to have surmounted, what casts the pall over our victory celebration? It is the curse of dimensionality, a malediction that has plagued the scientist from the earliest days."– Richard Bellman

1.   Introduction

Many real data sets are very high dimensional. In some scenarios, real data sets may contain hundreds or thousands of dimensions. With increasing dimensionality, many of the conventional outlier detection methods do not work very effectively. This is an artifact of the well known *curse of dimensionality*. In high-dimensional space, the data becomes sparse, and the true outliers become masked by the noise effects of multiple dimensions, when analyzed in *full dimensionality*.

A main cause of the dimensionality curse is the difficulty in defining locality for the high dimensional case. For example, proximity-based methods define locality with the use of distance functions. On the other hand, it has been shown in [65, 215], that all pairs of points are almost equidistant in high-dimensional space. This is referred to as *data sparsity*. Since outliers are defined as data points in sparse regions, this results in a poorly discriminative situation where all data points are situated in an almost equally sparse regions in full dimensionality. The challenges arising from the dimensionality curse are not specific to outlier detection. It is well known that many problems such as clustering and similarity search experience qualitative challenges with increasing

dimensionality [5, 7, 95, 215]. In fact, it has been suggested that almost any algorithm which is based on the notion of proximity would degrade qualitatively in higher dimensional space, and would therefore need to re-defined in a more meaningful way [8]. The impact of the dimensionality curse on the outlier detection problem was first noted in [4].

In order to further explain the causes of the ineffectiveness of full dimensional outlier analysis algorithms, a motivating example will be presented. In Figure 5.1, four different 2-dimensional views of a hypothetical data set have been illustrated. Each of these views corresponds to a disjoint set of dimensions. It is evident that point A is exposed as an outlier in the first view of the data set, whereas point B is exposed as an outlier in the fourth view of the data set. However, neither of the data points A and B are exposed as outliers in the second and third views of the data set. These views are therefore *noisy* from the perspective of measuring the outlierness of A and B. In this case, three of the four views are quite non-informative and noisy for exposing any *particular* outlier A or B. In such cases, the outliers are lost in the random distributions within these views, when the distance measurements are performed in *full* dimensionality. This situation is often naturally magnified with increasing dimensionality. For data sets of very high dimensionality, it is possible that only a very small fraction of the views may be informative for the outlier analysis process.

What does the aforementioned pictorial illustration tell us about the issue of locally relevant dimensions? The physical interpretation of this situation is quite intuitive in practical scenarios. An object may have several measured quantities, and significantly abnormal behavior of this object may be reflected only in a small subset of these quantities. For example, in an airplane mechanical fault detection scenario, the results of thousands of different airframe tests on the same plane may mostly be normal, with some noisy variations, which are not significant. On the other hand, some deviations in a small subset of tests may be significant enough to be indicative of anomalous behavior. When the data from the tests are represented in full dimensionality, the anomalous data points will not appear significant in virtually all views of the data, except for a very small fraction of the dimensions. Therefore, aggregate proximity measures are unlikely to expose the outliers, since the noisy variations of the vast number of normal tests will mask the outliers. Furthermore, when different objects (instances of different airframes) are tested, then different tests (subsets of dimensions) may be relevant to finding the outliers, which emphasizes the *local* nature of the relevance.

(a) View 1
Point *A* is outlier

(b) View 2
No outliers

(c) View 3
No outliers

(d) View 4
Point *B* is outlier
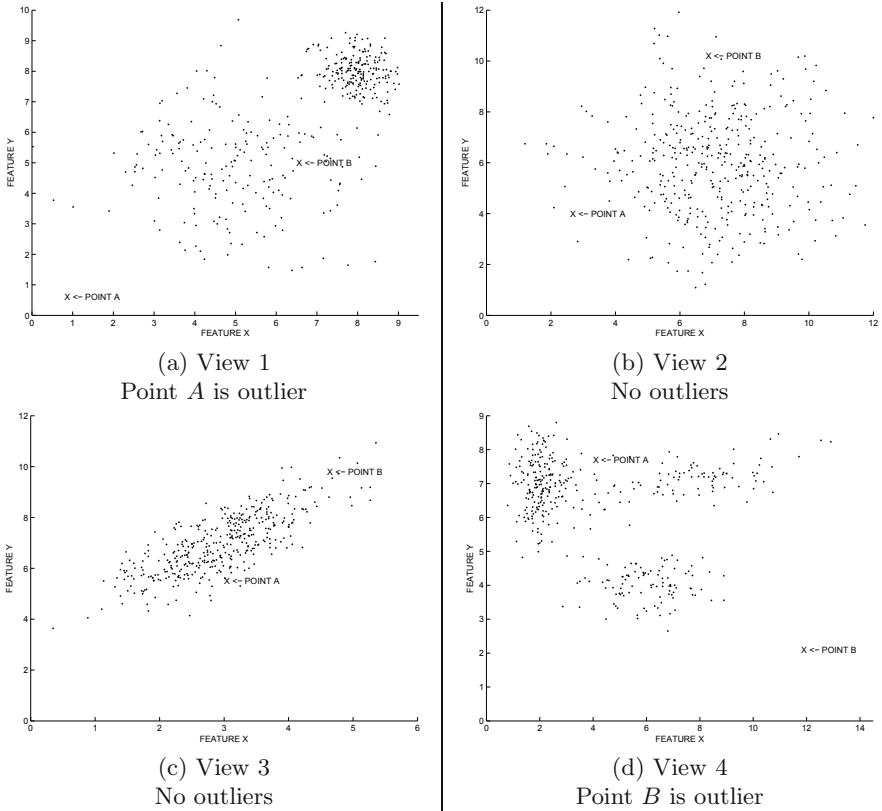
Figure 5.1. The outlier behavior may be lost in a majority of randomly chosen subspaces in the high dimensional case.

What does this mean for full-dimensional analysis in such scenarios? When full-dimensional distances are used in order to measure deviations, the dilution effects of the vast number of "normally noisy" dimensions will make the detection of outliers difficult. In most cases, this will show up as concentration effects in the distances, from the noise in the other dimensions. This may make the computations more erroneous. Furthermore, the additive effects of the noise present in the large number of different dimensions will interfere with the detection of actual deviations. Simply speaking, *outliers are lost in low-dimensional subspaces, when full-dimensional analysis is used, because of the masking and dilution effects of the noise in full dimensional computations* [4].

Similar effects are also experienced for other distance-based methods such as clustering and similarity search. For these problems, it has been shown [5, 7, 215] that by examining the behavior of the data in subspaces, it is possible to design more meaningful clusters which are specific to the particular subspace in question. This broad observation is generally true of the outlier detection problem as well. Since the outliers may only be discovered in low dimensional subspaces of the data, it makes sense to explore the lower dimensional subspaces for deviations of interest. Such an approach filters out the additive noise effects of the large number of dimensions, and results in more robust outliers.

Such a problem is very challenging to address effectively. This is because the number of possible projections of high dimensional data is exponentially related to the dimensionality of the data. The problem of outlier detection is like finding a needle in a haystack, *even when we know* the relevant dimensions of interest. Being forced to determine the relevant subsets of dimensions *in addition to this challenge* is equivalent to suggesting that even the haystack of interest is hidden in an exponential number of possible haystacks. An important observation is that subspace analysis in the context of the outlier detection problem is generally more difficult than in the case for problems such as clustering, which are based on aggregate behavior. This is because outliers, by definition, are rare, and therefore statistical aggregates on individual dimensions in a given locality often provide *very weak* hints for the subspace exploration process as compared to aggregation-based methods such as clustering. When such weak hints result in the omission of relevant dimensions, the effects can be much more drastic than the inclusion of irrelevant dimensions, especially in the interesting cases when the number of locally relevant dimensions is a small fraction of the full data dimensionality. A common mistake is to assume that the complementarity relationship between clustering and outlier analysis can be extended to the problem of local subspace selection. In particular, blind adaptations of dimension

selection methods from earlier subspace clustering methods, which are unaware of the nuances of subspace analysis principles across different problems, may sometimes miss important outliers. In this context, it is also crucial to recognize the difficulty in identifying relevant subspaces for outlier analysis, and use robust methods which combine the results from different subspaces.

An effective outlier detection method would need to search the data points and dimensions in *an integrated way*, so as to reveal the most relevant outliers. This is because different subsets of dimensions may be relevant to different outliers, as is evident from the example in Figure 5.1. The integration of point and subspace exploration leads to a further expansion in the number of possibilities which need to be examined for outlier analysis. This chapter will focus on subspace exploration methods, which attempt to find the relevant outliers by sifting through different subsets of dimensions in the data in an ordered way. This is accomplished simultaneously with a data-specific evaluation process, so that relevant data points are reported as outliers without having to explore all the subspaces in an exhaustive way. The idea is to determine the relevant subsets of dimensions in which the most important outliers are revealed as quickly as possible. This model is referred to as *projected outlier detection* [4]. Correspondingly, this chapter will present a number of algorithms, which achieve this goal.

Several classes of methods are commonly used in order to discover the relevant subspaces:

- **Rarity-based:** These methods attempt to discover the subspaces based on rarity of the underlying distribution. The major challenge here is computational, since the number of rare subspaces is far larger than the number of dense subspaces in high dimensionality.

- **Unbiased:** In these methods, the subspaces are sampled in an unbiased way, and scores are combined across different subspaces.

- **Aggregation-based:** In these methods, aggregate statistics such as cluster statistics, variance statistics, or non-uniformity statistics of local or global subsets of the data are used in order to determine the relevance of subspaces. Note that the difference from rarity-based statistics, is that instead of trying to determine the *number of data points* in a pre-specified local subspace, these methods typically analyze the statistical distributions of pre-specified local or global reference sets of points. Since such methods use statistics over local or global *subsets* of the data, it provides some *hints* for relevant subspaces for exploration. However, since such hints

are weak, and are not guaranteed to be the correct ones, multiple subspace sampling is crucial.

This chapter is organized as follows. Evolutionary algorithms for outlier detection are discussed in section 2. These algorithms are based on a grid-based approach for defining outliers. Distance-based methods for subspace outlier detection are studied in section 3. Methods for using and combining multiple subspaces in order to determine relevant outliers are discussed in section 4. The problem of determining outliers in generalized subspaces is discussed in section 5. The limitations of subspace analysis are discussed in section 6. The conclusions and summary are presented in section 7.

2.   Projected Outliers with Grids

A first approach to projected outlier detection was presented in [4]. Projected outliers are determined by finding *localized regions of the data in low dimensional space*, which have abnormally low density. Thus, the first step is to identify and mine those *localized* patterns which contain data points, but have abnormally low density. Thus, the goal is to determine interesting anomalies, rather than the noise in the data. Once such localized regions have been identified, then the outliers are defined as those records which have such patterns present in them. An interesting observation is that such lower dimensional projections can be determined even in data sets with missing attribute values. This is quite useful for many real applications, in which feature extraction is a difficult process and full feature descriptions often do not exist. For example, in the airframe fault detection scenario introduced earlier in this chapter, it is possible that only a subset of tests may have been applied, and therefore the values in only a subset of the dimensions may be available for outlier analysis.

2.1   Defining Abnormal Lower Dimensional Projections

In order to find such abnormal lower dimensional projections, it is important to provide a proper statistical definition of an abnormal lower dimensional projection. An abnormal lower dimensional projection is one in which the density of the data is exceptionally lower than average. In this context, the methods for extreme value analysis introduced in Chapter 2 are useful.

A grid-based approach is used in order to determine projections of interest. The first step is to perform a grid discretization of the data. Each attribute of the data is divided into ϕ ranges. These ranges are

created on an equi-depth basis. Thus, each range contains a fraction $f = 1/\phi$ of the records. The reason for using equi-depth ranges as opposed to equi-width ranges is that different localities of the data have different densities. Therefore, such an approach partially adjusts for the local variations in data density during the initial phase. These ranges form the units of locality which are used in order to define low dimensional projections which have unreasonably sparse regions.

Consider a k-dimensional cube which is created by picking grid ranges from k different dimensions. The expected fraction of the records in that region is equal to f^k, if the attributes were statistically independent. Of course, the data is far from statistically independent and therefore the actual distribution of points in a cube would differ significantly from average behavior. Many of the local regions may contain very few data points, if any. It is precisely these abnormally sparse regions, which are useful for the purpose of outlier detection.

It is assumed that the total number of points in the database is denoted by N. Under the afore-mentioned independence assumption, the presence or absence of any point in a k-dimensional cube is a bernoulli random variable with probability f^k. Then, the expected fraction and standard deviation of the points in a a k-dimensional cube is given by $N \cdot f^k$ and $\sqrt{N \cdot f^k \cdot (1 - f^k)}$. Furthermore, if the number of data points N is large, then the central limit theorem can be used to *approximate* the number of points in a cube by a normal distribution. Let $n(\mathcal{D})$ be the number of points in a k-dimensional cube $\mathcal{D}$. The sparsity coefficient $S(\mathcal{D})$ of the data set $\mathcal{D}$ can be computed as follows:

$$S(\mathcal{D}) = \frac{n(D) - N \cdot f^k}{\sqrt{N \cdot f^k \cdot (1 - f^k)}}$$

Only sparsity coefficients which are negative are indicative of local projected regions, in which the presence of the points is significantly lower than expected. Since $n(\mathcal{D})$ is assumed to fit a normal distribution, the normal distribution tables can be used to quantify the probabilistic level of significance of its deviation. Of course, while the independence assumption is almost never completely true, it provides a good heuristic for determining the level of abnormality of the underlying data points in practice.

2.2   Evolutionary Algorithms for Outlier Detection

It is evident from the discussion in the introduction, that an exhaustive search of all the subspaces in the data for outliers is unlikely to be

fruitful, because of high computational complexity. Therefore, an ordered search method is required, which prunes off most of the subspaces automatically during the exploration process. Since the search space is noisy and unstructured in this case, this is a natural candidate for the use of evolutionary algorithms.

The nature of this problem is such that there are no upward or downward-closed properties on the grid-based subspaces satisfying the sparsity condition.[1] Unlike problems such as frequent pattern mining [28] where one is looking for large aggregate patterns, the problem of finding subsets of dimensions which are sparsely populated has the flavor of finding a needle in haystack. Furthermore, it may often be the case that even though particular regions may be well populated on certain sets of dimensions, they may be very sparsely populated when such dimensions are combined together. For example, in a given data set, there may be a large number of individuals clustered at the age of 20 (low local variance), and a modest number of individuals with varying levels of diabetes (modest local variance). However, *very rare* individuals would satisfy both criteria, because the disease does not affect young individuals. From the perspective of outlier detection, a 20-year old with diabetes is a very interesting record. However, the interestingness of the pattern is not even hinted at by its lower dimensional projections, or the relative variances in these individual projections. Therefore, the best projections are often created by an unknown combination of dimensions, whose lower dimensional projections may contain very few hints for proper subspace exploration. One solution is to change the measure in order to force better closure or pruning properties; however this can worsen the quality of the solution substantially by forcing the choice of the measure to be driven by algorithmic considerations. In general, it is not possible to predict the behavior of the data when two sets of dimensions are combined. Therefore, a natural option is to develop search methods which can identify such hidden combinations of dimensions. In order to search the exponentially increasing space of possible projections, the work in [4] borrows ideas from a class of evolutionary search methods in order to reduce the size of the search space.

Evolutionary Algorithms [223] are methods which imitate the process of organic evolution [125] in order to solve parameter optimization problems. In evolutionary methods, every solution to an optimization problem can be disguised as an individual in an evolutionary system. The

---

[1]An upward closed pattern is one in which all supersets of the pattern are also valid patterns. A downward closed set of patterns is one in which all subsets of the pattern are also members of the set.

measure of fitness of this "individual" is equal to the objective function value of the corresponding solution, and the other species which this individual has to compete with are a group of other solutions to the problems. Appropriate operations are defined in order to imitate the recombination and mutation processes as well, and the simulation is complete. Each feasible solution is encoded in the form of a string and is the chromosome representation of the solution. The process of conversion of feasible solutions of the problem into strings which the algorithm can use is referred to as its *encoding*. The measure of fitness of a string is evaluated by the *fitness function*. This is equivalent to the objective function value of the solution. The better the objective function value, the better the fitness value. As the process of evolution progresses, all the individuals in the population typically improve in fitness and also become more similar to each other. Dejong [134] defined convergence of a particular position in the string, as the the stage at which 95% of the population had the same value for that gene. The population is said to have converged when all positions in the string representation have converged.

The relevant localized subspace patterns can be easily represented as strings. Let us assume that the grid range for the ith dimension is denoted by m_i. Then, the value of m_i can take on any of the values 1 through ϕ, or it can take on the value $*$, which denotes a "don't care". Thus, there are a total of $\phi + 1$ values that the dimension m_i can take on. Thus, consider a 4-dimensional problem with $\phi = 10$. Then, one possible example of a solution to the problem is given by *3*9. In this case, the ranges for the second and fourth dimension are identified, whereas the first and third are left as "don't cares". The evolutionary algorithm uses the dimensionality of the projection k as an input parameter. Therefore, for a d-dimensional data set, the string of length d will contain k specified position and $(d - k)$ "don't care" positions. The fitness for the corresponding solution may be computed using the sparsity coefficient discussed earlier. The evolutionary search technique starts with a population of p random solutions and iteratively used the processes of selection, crossover and mutation in order to perform a combination of hill climbing, solution recombination and random search over the space of possible projections. The process is continued until the population converges to a global optimum according to the *Dejong convergence criterion*[134]. At each stage of the algorithm, the m best projection solutions (most negative sparsity coefficients) are kept track of. At the end of the algorithm, these solutions are reported as the best projections in the data. The following operators are defined for selection, crossover and mutation:

- **Selection:** The copies of a solution are replicated by ordering them by rank and biasing them in the population in the favor of higher ranked solutions. This is referred to as *rank selection.*

- **Crossover:** The crossover technique is key to the success of the algorithm, since it implicitly defines the subspace exploration process. One solution is to use a uniform two-point crossover in order to create the recombinant children strings. The two-point crossover mechanism works by determining a point in the string at random called the crossover point, and exchanging the segments to the right of this point. However, such a blind recombination process may create poor solutions too often. Therefore, an optimized crossover mechanism is defined. In this case, it is guaranteed that both children solutions correspond to a k-dimensional projection as the parents, and the children typically have high fitness values. This is achieved by examining a subset of the different possibilities for recombination and picking the best among them.

- **Mutation:** In this case, random positions in the string are flipped with a predefined mutation probability. Care must be taken to ensure that the dimensionality of the projection does not change after the flipping process.

At termination, the algorithm is followed by a postprocessing phase. In the postprocessing phase, all data points containing the abnormal projections are reported by the algorithm as the outliers. The approach also provides the relevant projections which provide the *causality* (or intensional knowledge) for the outlier behavior of a data point. Thus, this approach also has a high degree of interpretability in terms of providing the reasoning for *why* a data point should be considered an outlier.

3.   Distance-based Subspace Outlier Detection

In these methods, distance-based models are used in lower dimensional subspaces of the data in order to determine the relevant outliers. There are two major variations to the common task.

- In one class of models, the outliers are determined by exploring relevant subspaces.

- In another class of methods, the relevant outlying subspaces for a given data point are determined. This is more useful for providing *intensional knowledge*, for illustrating *why* a specific data point is an outlier.

The second class of methods shares similarities with the approach used in [262] for finding intensional knowledge from distance-based outliers. Both classes of methods will be discussed in subsequent sections.

3.1   Subspace Outlier Degree

A distance-based method for finding outliers in lower dimensional projections of the data is proposed in [273]. In this approach, instead of trying to find local subspaces of abnormally low density over the whole data, a local analysis is provided specific to each data point. For each data point $\overline{X}$, a set of reference points $S(\overline{X})$ are determined, which represent the proximity of the current data point being examined.

Once this reference set $S(\overline{X})$ has been determined, the relevant subspace for $S(\overline{X})$ is determined as the set $Q(\overline{X})$ of dimensions in which the variance is small. The specific threshold is picked as a user-specified fraction of the average dimension-specific variance of the data points in $S(\overline{X})$. Thus, this approach analyzes the statistics of individual dimensions independently of one another during the crucial step of subspace selection, though this may sometimes not be helpful for picking the best subspace projections. The approach of analyzing the distance behavior of individual dimensions for picking the subspace set $Q(\overline{X})$ is a rather naive generalization derived from subspace clustering methods. Unlike data clustering, the effectiveness of subspace outlier methods is almost entirely dependent upon the identification of dimensions containing rare points rather than dimensions with specific kinds of aggregate statistics. In outlier analysis, aggregate data measures such as the dimension-specific variance tell us very little about the subspace behavior of the rare points, and which choices of subspaces are likely to be most relevant for identification of these very unusual points. In some cases such as the example of the young diabetes patient discussed earlier, the unusual behavior is manifested in combinations of dimensions rather than the variances of the individual dimensions. If the absolute variance of a particular dimension such as the diabetes level is not deemed to be sufficiently low, it will not selected in the projection.

In the interesting cases, where the number of relevant dimensions is limited, the negative effects of removing a single relevant dimension can be even more drastic than keeping many irrelevant dimensions. The particularly problematic factor here is that if a mistake is made in subspace selection, there is virtually no chance of recovering from the mistake, when a single subspace is picked for analysis. As we will discuss later, other more insightful techniques in [256, 337] mitigate these impacts by using multiple subspaces for outlier analysis.

The euclidian distance of $\overline{X}$ is computed to the mean of the reference set $S(\overline{X})$ *in the subspace defined by* $Q(\overline{X})$. This is denoted by $G(\overline{X})$. The value of $G(\overline{X})$ is affected by the number of dimensions in $Q(\overline{X})$. The *subspace outlier degree* $SOD(\overline{X})$ of a data point is defined by normalizing this distance $G(\overline{X})$ by the number of dimensions in $Q(\overline{X})$.

$$SOD(\overline{X}) = \frac{G(\overline{X})}{|Q(\overline{X})|}$$

It remains to explain how the reference set $S(\overline{X})$ is generated with the use of distances. This may sometimes turn out to be a challenge, since the concept of proximity is itself hard to define in full dimensional space. Therefore, there is a circularity in using full dimensional distances to pick the reference set. The work [273] uses a shared nearest neighbor approach in order to compute this locality.

This work tries to find the outliers in a *single* subspace of the data, on the basis of local analysis. In practice, the deviations may be hidden in unusual subspaces which are not evident from the 1-d variance statistics of the reference set. Therefore, if the wrong subspace is selected by aggregate analysis, it is quite likely that many outliers may be missed. Furthermore, since the different dimensions in the data may combine to provide unusual results, it is sometimes more helpful to evaluate the locality of a data point in a subspace by examining the data distribution in the entire subspace, rather than examining the different dimensions independently from one another.

3.2   Finding Distance-based Outlying Subspaces

Most of the methods for outlier detection attempt to search for relevant subspaces in order to find outliers. However, some recent methods [499–501] are designed for finding the outlying subspaces *for a given data point*. Thus, the causality in this case is the other way around, where subspaces are determined from points.

A system called *HOS-Miner* was presented in [499]. According to this work, the definition of the outlying subspace for a given data point $\overline{X}$ is as follows:

DEFINITION 5.1 *For a given data point* $\overline{X}$, *determine the set of subspaces such that the sum of its k-nearest neighbor distances in that subspace is at least* δ.

This approach does not normalize the distances with the number of dimensions. Therefore, a subspace becomes more likely to be outlying with increasing dimensionality. This definition also exhibits closure

properties in which any subspace of a non-outlying subspace is also not outlying. Similarly, every superset of an outlying subspace is also outlying. Clearly, only *minimal* subspaces which are outliers are interesting. The method in [499] uses both downward- and upward-closure properties to prune off subspaces which are either not relevant or not interesting. An X-Tree is used in order to perform the indexing for performing the k-nearest neighbor queries in different subspaces efficiently. It should be noted that while the closure properties result in better efficiency and algorithmic convenience, they do not necessarily imply greater effectiveness. As the earlier example with the young diabetes patient illustrated, true outliers are often hidden in subspaces of the data, which cannot be inferred from their lower or higher dimensional projections.

In order to further improve the efficiency of the learning process, the work in [499] uses a random sample of the data in order to learn about the subspaces before starting the subspace exploration process. This is achieved by estimating a quantity called the *Total Savings Factor (TSF)* of the outlying subspaces. These are used to regulate the search process for specific query points and prune the different subspaces in an ordered way. Furthermore, the TSF values of different subspaces are dynamically updated as the search proceeds. It has been shown in [499] that such an approach can be used in order to determine the outlying subspaces of specific data points efficiently. Numerous methods for using different kinds of pruning properties and genetic algorithms for finding outlying subspaces are presented in [500, 501].

4. Combining Outliers from Multiple Subspaces

One of the major challenges of subspace analysis is that a given data point may show very different behavior in terms of its outlier degree in different subspaces. This also corresponds to the fact that the *outlier scores* from different subspaces may all be very different. These need to be combined into a unified outlier score. This principle is generally related to that of ensemble-analysis, which was discussed in Chapter 1. A variety of methods have been proposed for examining different subspaces for outlier ranking.

4.1 Random Subspace Sampling

The simplest method for combining outliers from multiple subspaces is the use of random subspace sampling. In the work in [289], an approach called *feature bagging* is used, which is analogous to the ensemble technique often used in data classification. This approach also falls in the class of *independent ensembles* introduced in Chapter 1.

The broad approach is to repeatedly apply the following two steps:

- Randomly select between $(d/2)$ and d features from the underlying data set in iteration t in order to create a data set D_t in the tth iteration.

- Apply the outlier detection algorithm O_t on the data set D_t in order to create score vectors S_t.

In principle, the outlier detection algorithm O_t used for the tth iteration could be different. However, the work in [289] uses the LOF algorithm for all the iterations.

At the end of the process, the outlier scores from the different algorithms need to be combined. There are two distinct methods which are used in order to combine the different subspaces:

- *Breadth-first Approach:* In this approach, the ranking of the algorithms is used for combination purposes. The top-ranked outliers over all the different executions are ranked first, followed by the second-ranked outliers (with repetitions removed), and so on. Minor variations could exist because of tie-breaking between the outliers within a particular rank.

- *Cumulative Sum Approach:* The outlier scores over the different algorithm executions are summed up. The top ranked outliers are reported on this basis.

It was shown in [289] by synthetic data analysis, that combining methods are important when some of the features are noisy. In such cases, full-dimensional algorithms are unable to distinguish the true outliers from the normal data, because of the additional noise. Improvements over the base LOF-approach were also observed with the use of real-data analysis. At first sight, it would seem that random subspace sampling [289] does not attempt to optimize the discovery of subspaces to finding rare instances at all. Nevertheless, it does have the paradoxical merit that it is relatively efficient to sample subspaces, and therefore a large number of subspaces can be sampled in order to improve robustness. The robustness resulting from multiple subspace sampling is clearly a very desirable quality, as long as the combination function at the end recognizes the differential behavior of different subspace samples for a given data point. In a sense, this approach implicitly recognizes the difficulty of detecting relevant and rare subspaces for the outlier detection problem, and therefore approaches the problem by sampling as many subspaces as possible in order to reveal the rare behavior. From a conceptual perspective, this approach is similar to that of harnessing the

power of many weak learners to create a single strong learner in classification problems. The approach has been shown to show consistent performance improvement over full dimensional methods for many real data sets in [289]. This approach may also be referred to as the *feature bagging method* or *random subspace ensemble method*. This approach is likely to have significant potential for improving subspace analysis, by experimenting with different choices of combination functions.

The work in [310] designs the concept of *isolation forest*, which derives its motivation from another ensemble technique known as *random forests*, which are commonly used in classification. In this case, the data is recursively partitioned by axis-parallel cuts along randomly selected attributes, so as to isolate different kinds of instances from one another. In such cases, the tree branches containing outliers are noticeably less deep, because these data points are quite different from the normal data. Thus, data points which have noticeably shorter paths in the branches of different trees are more likely to be outliers. The different branches correspond to different local subspace regions of the data, depending on how the attributes are selected for splitting purposes. The smaller path methods correspond to lower dimensionality of the subspaces in which the outliers have been isolated. The final combination step is performed by using the path lengths of the data points in the different samples. One major challenge of using such an approach is that when the dimensionality of the data increases, an incorrect choice of attribute for splitting at the higher levels of the tree is more likely to mislead the detection approach. Nevertheless, the approach is efficient in determining each subspace sample, and the use of multiple subspace samples is a desirable quality of the approach.

4.2   Selecting High Contrast Subspaces

The subspace ensemble method [289] discussed in the last section randomly samples subspaces. If many dimensions are noisy, at least a few of them are likely to be included in each subspace sample. This implies that a larger number of subspace samples will be required in order to obtain more robust results. Therefore, it is natural to ask whether it is possible to perform a pre-processing in which a smaller number of *high-contrast* subspaces are selected.

In the work proposed in [256], the outliers are found only in these high-contrast subspaces, and the corresponding scores are combined together. Thus, this approach decouples the subspace search as a a generalized pre-processing approach from the outlier ranking of the individual data points. The approach discussed in [256] is quite interesting because

of its pre-processing approach to finding relevant subspaces in order to reduce the irrelevant subspace exploration. While the high contrast subspaces are obtained using aggregation-based methods, the aggregation behavior is only used as hints in order to identify multiple subspaces for greater robustness. The assumption here is that rare events are *statistically more likely* to occur in subspaces where there is significant non-uniformity and contrast. The final outlier score combines the results over different subspaces. The insight in the work of [256] is to combine subspace selection and multiple subspaces analysis in order to determine the relevant outlier scores. Therefore, the risk of not picking the correct subspace is reduced. This approach has been shown to work well in [256] over the random subspace sampling method.

The conditional probability for an an attribute value along any particular dimension $P(x_1|x_2 \ldots x_d)$ is the same as its unconditional probability $P(x_1)$ for the case of uncorrelated data. High-contrast subspaces are likely to violate this assumption because of non-uniformity in data distribution. In our earlier example of the young diabetes patients, this corresponds to the unexpected rarity of the *combination* of youth and the disease. The idea is that subspaces with such unexpected non-uniformity are more *likely* to contain outliers, though it is treated only as a weak hint for pre-selection of one of multiple subspaces.

A variety of tests based on the student's t-distribution can be used in order to measure the deviation of this sample from the basic hypothesis of independence. This provides a measure of the non-uniformity of the subspace, and therefore provides a way to measure the quality of the subspaces in terms of their propensity to contain outliers. A bottom-up *Apriori* style [29] approach was proposed in order to determine the relevant projections. In this approach the subspaces are continuously extended to higher dimensions for testing. Details of the approach are available in [256].

4.3   Local Selection of Subspace Projections

The work in [337] uses *local* statistical selection of relevant subspace projections in order to determine outliers. In other words, the selection of the subspace projections is optimized to specific data points, and therefore the locality of a given data point matters in the selection process. For each data point $\overline{X}$, a set of subspaces is identified, which are considered *high contrast* subspaces from the perspective of outlier detection. However, this exploration process uses the high contrast behavior as statistical *hints* in order to explore *multiple* subspaces for robustness, since a single subspace may often miss the true projection.

Algorithm $OUTRES$(Data Point: $\overline{X}$
       Subspace: S);
begin
  for each attribute i not in S
  if $S_i = S \cup \{i\}$ passes non-uniformity test **then**
  begin
   Compute $OS(S_i, \overline{X})$;
   $O(\overline{X}) = OS(S_i, \overline{X}) \cdot O(\overline{X})$;
   $OUTRES(\overline{X}, S_i)$;
  end
end

Figure 5.2. The OUTRES Algorithm

The $OUTRES$ method [337] examines the density of lower dimensional subspaces in order to identify relevant projections. The basic hypothesis, is that for a given data point $\overline{X}$ it is desirable to determine subspaces in which the data is sufficiently non-uniformly distributed in its locality. In order to characterize the distribution of the locality of a data point, the work in [337] computes the density of the locality of data point $\overline{X}$ in subspaces S as follows:

$$den(S, \overline{X}) = |\mathcal{N}(\overline{X}, S)| = |\{\overline{Y} : dist(\overline{X}, \overline{Y} \leq \epsilon\}|$$

This is the simplest possible definition of the density, though other more sophisticated methods such as kernel density estimation [409] are used in $OUTRES$ in order to obtain more refined results. Kernel density estimation is also discussed in Chapter 4. A major challenge here is in comparing the subspaces of varying dimensionality. This is because the density of the underlying subspaces reduces with increasing dimensionality. It has been shown in [337], that it is possible to obtain comparable density estimates across different subspaces of different dimensionalities, by selecting the bandwidth of the density estimation process according to the dimensionality of the subspace.

Furthermore, the work in [337] uses statistical techniques in order to meaningfully compare different subspaces. For example, if the data is uniformly distributed, then the number of data points lying within a distance ϵ of the data point should be regulated by the fractional volume of the data in that subspace. Specifically, the fractional parameter defines a binomial distribution characterizing the number of points in that volume, if that data were to be uniformly distributed. Of course, one is really interested in subspaces which deviate significantly from this

behavior. The (local) relevance of the subspace for a particular data point $\overline{X}$ is computed using statistical testing. The two hypothesis are as follows:

- Hypothesis H_0: The local subspace neighborhood $\mathcal{N}(\overline{X}, S)$ is uniformly distributed.

- Hypothesis H_1: The local subspace neighborhood $\mathcal{N}(\overline{X}, S)$ is not uniformly distributed.

The Kolmogorov-Smirnoff goodness of fit test [424] is used to determine which of the afore-mentioned hypothesis are true. It is important to note that this process provides an idea of the *usefulness* of a subspace, and is used in order to enable a *filtering condition* for removing irrelevant subspaces from the process of computing the outlier score of a specific data point. A subspace is defined as relevant, if it passes the hypothesis condition H_1. In other words, outlier scores are computed using a combination of subspaces which *must* satisfy this relevance criterion.

In order to combine the scores which are obtained from multiple *relevant* subspaces, the work in [337] uses the product of the outlier scores obtained from different subspaces. Thus, if $S_1 \ldots S_k$ be the different abnormal subspaces found for data point $\overline{X}$, and if $O(S_i, \overline{X})$ be the outlier score from subspace S_i, then the overall outlier score $OS(\overline{X})$ is defined as follows:

$$OS(\overline{X}) = \prod_i O(S_i, \overline{X})$$

It is evident that *low scores* represent a greater tendency to be an outlier. The advantage of using the product over the sum, is that the latter is dominated by the high scores, as a result of which a few subspaces containing normal behavior will dominate the sum. On the other hand, in the case of the product, the outlier behavior in a small number of subspaces will be greatly magnified. This is particularly appropriate for the problem of outlier detection. So far, it has not been discussed, how the actual subspaces $S_1 \ldots S_k$ are determined. This will be achieved with a careful subspace exploration.

In order to actually define the outlier score, subspaces are considered significant for particular objects only if their density is at least two standard deviations less than the mean value. This is essentially a filter condition for that subspace to be considered deviant. Thus, the deviation $dev(\overline{X}, S_i)$ of the data point $\overline{X}$ in subspace S_i is defined as the ratio of the deviation of the density of the object from the mean density, divided by two standard deviations.

$$dev(S_i, \overline{X}) = \frac{\mu - den(S_i, \overline{X})}{2 \cdot \sigma}$$

The outlier score of a data point in a subspace is the ratio of the density of the point in the space to its deviation, if it satisfies the filter condition of the density being at least two standard deviations less than the mean. Otherwise the outlier score is considered to be 1, and it does not affect the overall outlier score in the product function defined earlier for combining different subspaces. Thus, for the points satisfying the filter condition, the outlier score $OS(S_i, \overline{X})$ is defined as follows:

$$O(S_i, \overline{X}) = \frac{den(S_i, \overline{X})}{dev(S_i, \overline{X})}$$

An observation in [337] is that subspaces which are either very low dimensional (eg. 1-d subspaces) or very high dimensional are not very informative from an outlier detection perspective. A recursive exploration of the subspaces is performed, where an additional attribute is included in the subspace for statistical testing. Therefore, the work in [337] uses recursive processing in which the subspaces are built in recursive fashion. When an attribute is added to the current subspace S_i, the non-uniformity test is utilized to determine whether or not that subspace should be used. Otherwise, this subspace is discarded.

The overall algorithm uses a recursive subspace exploration procedure in order to measure the outlierness of any particular object. Note that the entire recursive algorithm uses the data point $\overline{X}$ as input, and therefore the procedure needs to be applied separately *for each data point.* For any given subspace, an attribute is incrementally added. Then, the non-uniformity test is applied to determine if it is relevant. If it is not relevant, then the subspace is discarded. Otherwise, the outlier score $O(S_i, \overline{X})$ in that subspace is computed for the data point, it is multiplied with the current value of $OS(\overline{X})$. Since the outlier scores of subspaces, which do not meet the filter condition are set to 1, they do not affect the density computation in this multiplicative approach. The procedure is then recursively called in order to explore the next subspace. Thus, such a procedure potentially explores an exponential number of subspaces, though the real number is likely to be much smaller in practice. This is because of the non-uniformity test, which prunes off large parts of the recursion tree during the exploration. The overall algorithm for subspace exploration for a given data point $\overline{X}$ is illustrated in .

5.   Generalized Subspaces

A significant amount of success has been achieved for finding outliers in axis-parallel subspaces in recent work. While these methods are effective for finding outliers in cases where the outliers naturally deviate in
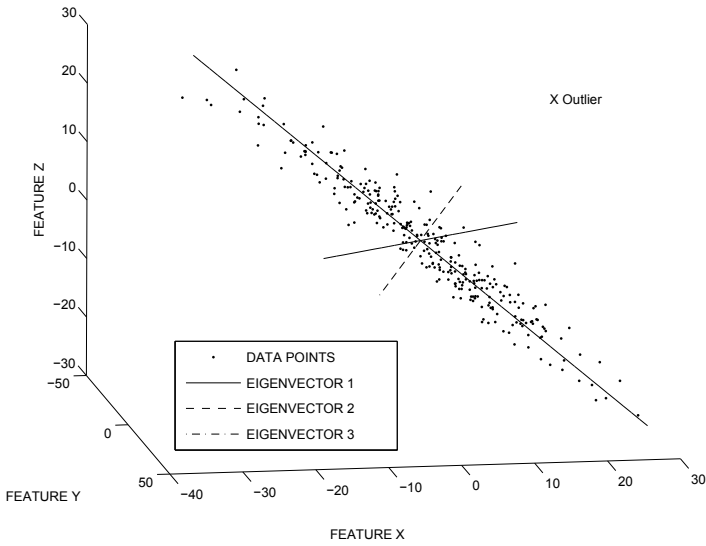
Figure 5.3. The example of Figure 3.4 re-visited: Global PCA can discover outliers in cases, where the entire data is aligned along lower dimensional manifolds.
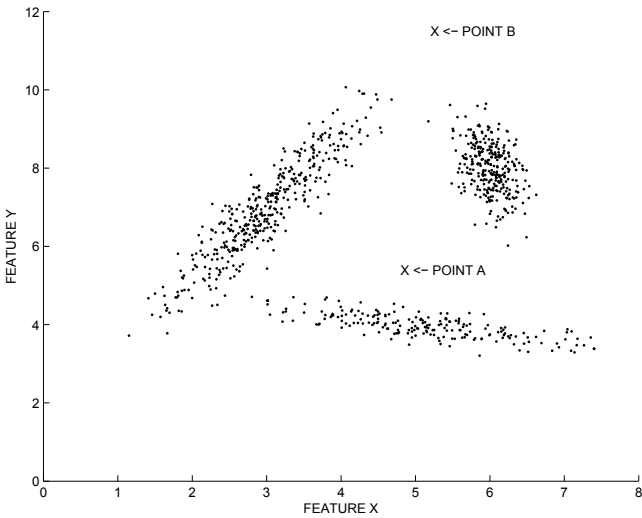
Figure 5.4. The example of Figure 2.7 revisited: Outliers are best discovered by determining deviations from local PCA-based clusters. Neither axis-parallel subspace outliers nor global-PCA can capture such clusters.

specific subspaces from the clusters, they are not very useful for finding clusters in cases where the points are aligned along lower-dimensional manifolds of the data. For example, in the case of Figure 5.4, no 1-dimensional subspace analysis from the 2-dimensional data can find the outliers. On the other hand, it is possible to find *localized* 1-dimensional correlated subspaces so that most of the data aligns along these localized 1-dimensional subspaces, and the remaining deviants can be classified as outliers.

These algorithms are generalizations of the following two classes of algorithms:

- The PCA-based linear models discussed in Chapter 3 find the *global* regions of correlation in the data. For example, in the case of Figure 5.3, the outliers can be effectively identified by determining these global directions of correlation. However, no such *global* directions of correlation exist in the case of Figure 5.4.

- The axis-parallel subspace outliers discussed earlier in this chapter can find deviants, when the data is naturally aligned along low dimensional axis-parallel subspace clusters. However, this is not the case in Figure 5.4, where the data is aligned along arbitrary directions of correlation.

This problem can be partially addressed with the use of generalized projected clustering methods, where the clusters are determined in arbitrarily aligned subspaces of the data [7]. The method discussed in [7] has a built-in mechanism in order to determine the outliers *in addition to* the clusters. Such outliers are naturally data points which do not align with the clusters. However, the approach is not particularly optimized for finding the outliers, because the primary purpose of the method is to determine the clusters. The outliers are discovered as a side-product of the clustering algorithm, rather than as the primary goal. Therefore, the approach may discover the weaker outliers, which correspond to the noise in the data. Similarly, the approach in [132] is focussed on determining the noise in the data for improving mixture modeling of probabilistic PCA algorithms. In order to determine the outliers which are optimized to the locality of a particular data point, it is critical to determine localized subspaces which are optimized to the data point $\overline{X}$, which is being evaluated for its outlier score. The determination of such subspaces is non-trivial, since it often cannot be inferred from locally aggregate properties of the data, for detecting the behavior of *rare* instances.

Another method was recently proposed in [274] for finding outliers in generalized subspaces of the data. The main difference from earlier gen-

eralized subspace clustering methods is that local reference sets are used for local correlation analysis. For a given data point $\overline{X}$, this method finds the full-dimensional k-nearest neighbors of $\overline{X}$. This provides a reference set S with mean vector $\overline{\mu}$. The PCA approach of Chapter 3 is applied to the covariance matrix $\Sigma(S)$ of the *local* reference set S in order to determine the key eigenvectors $\overline{e_1} \ldots \overline{e_d}$, in increasing order of variance, with corresponding eigenvalues $\lambda_1 \leq \lambda_2 \ldots \leq \lambda_d$. The discussion in section 3 of Chapter 3 performs these same steps [406] except that they are performed on a *global* basis, rather than on a local reference set S. Even if all d dimensions are included, it is possible to create a normalized outlier score of a data point $\overline{X}$, to the centroid $\overline{\mu}$ of the data with the use of local eigenvalue scaling, as discussed in Chapter 3:

$$Score(\overline{X}) = \sum_{j=1}^{d} \frac{|(\overline{X} - \overline{\mu}) \cdot \overline{e_j}|^2}{\lambda_j} \qquad (5.1)$$

As discussed in section 2.2.2 of Chapter 2, this can be approximately modeled as a χ^2 distribution with d degrees of freedom for each data point, and the outlier scores of the different data points can be reasonably compared to one another. Such an approach is used in [406] in the context of global data analysis. The survey paper of Chandola et al. [107] provides a simpler exposition. The work in [274] uses a similar approach with the use of a local reference set, selected with the use of full dimensional k-nearest neighbor distances.

Eigenvectors with large values of λ_i will usually not contribute much to the score, though as discussed below, this may not always be the case. Such directions are pruned from the score. The δ eigenvectors[2] with the smallest eigenvalues are picked for the computations above. Correspondingly, the pruned score is defined on the basis of the first $\delta \leq d$ eigenvectors only with the smallest eigenvalues.

$$Score(\overline{X}, \delta) = \sum_{j=1}^{\delta} \frac{|(\overline{X} - \overline{\mu}) \cdot \overline{e_j}|^2}{\lambda_j} \qquad (5.2)$$

How should the value of δ be determined for a particular data point $\overline{X}$? The score is a χ^2-distribution with δ-degrees of freedom. It was observed in [274] that the value of δ can be parameterized, by treating the χ^2 distribution as a special case of the Γ distribution.

$$Score(\overline{X}, \delta) \sim \Gamma(\delta/2, 2)$$

---

[2]The work in [274] uses δ as the number of *longest* eigenvectors, which is only a notational difference, but is noted here to avoid confusion.
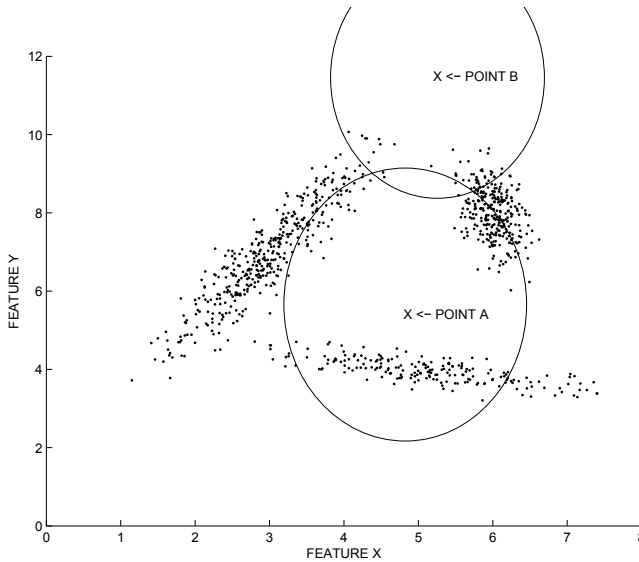
Figure 5.5. Local reference set may sometimes contain points from multiple generating mechanisms

The optimal value of δ is picked specifically for each data point, by picking the value of δ in order to determine the maximal unlikely deviation based on this model. This is done by using the cumulative density function of the aforementioned distribution. While this value can be directly used as an outlier score, it was also shown in [274], how this score may be converted into a more intuitive probability value.

This approach has several issues:

- A *single subspace* has been used by this approach for finding the outliers with the use of the local reference set S. If the local reference set S is not accurately determined, then this will not provide the proper directions of local correlation. The use of a single subspace is risky, especially with the use of weak aggregation-based hints, because it is often possible to unintentionally remove relevant subspaces. This can have drastic effects. The use of multiple subspaces may be much more relevant in such scenarios, such as the methods proposed in [289, 256, 337, 341].

- There is an inherent circularity in identifying the reference set with the use of full dimensional k-nearest neighbor distances, especially if the distances are not meaningfully defined in full dimensionality. The choice of points in the reference set and the choice of the subspace clearly impact each other in a circular way. This is a classical

"chicken and egg" problem in subspace analysis, which was first pointed out in [5]. The analysis in such cases needs to be *simultaneous* rather than *sequential*. As is well known, the most robust techniques for handling circularity in virtually all problem domains (eg. the EM algorithm and many projected clustering methods) use iterative methods, so that the point-specific and dimension-specific aspects of the problem are able to interact with one another. This is however, not the case in [274], where a sequential analysis is used.

In particular, it may happen that many locally irrelevant features may be used during the determination of the local reference set, when full dimensional distances are used. This set could therefore contain data points from multiple generating mechanisms, as illustrated in Figure 5.5. When the number of irrelevant features is unknown, a specific number of points in the reference set will not be able to avoid this problem. The use of a smaller reference set size can reduce the chance of this happening to some extent, but can never guarantee it, especially when many irrelevant features are used. On the other hand, reducing the reference set size can also result in a correlation hyperplane, whose eigenvalue statistics overfit an artificially small set of reference points.

- An interesting question arises, as to whether it is necessary to select a particular set of dimensions in a hard way, since the eigenvalues in the denominator of Equation 5.1 already provide a soft weighting to the importance (or relevance) of the different dimensions. For example, if for a large value of λ_i, a data point shows even larger deviations along that direction, such an outlier would either be missed by dimension pre-selection, or would include other less relevant dimensions. An example is the outlier B in Figure 5.5, which is aligned along the longer eigenvector, and therefore the longest eigenvector is the *most informative* about its outlier behavior. In particular, the method of picking the δ smallest eigenvectors implicitly assumes that the relevance of the attributes are ordered by eigenvalue magnitude. While this may generally be true for aggregation-based clustering algorithms, it is very often not true in outlier analysis because of the unusual nature of outliers. The possibility of outliers aligning along long eigenvectors is not uncommon at all, since two highly correlated attributes may often show highly deviant behavior of a similarly correlated nature. This example also shows, how *brittle* the rare nature of outlier analysis is to aggregation-based measures. This is because of the varying

causes of rarity, which cannot be fully captured in aggregation statistics. This is relevant to our discussion in the introduction section, that straightforward generalizations of subspace selection methods from clustering (based on aggregates), are often not appropriate or optimized for (the rare nature of) outlier analysis. One advantage of using all the dimensions is that it reduces to a local Mahalanobis distance with the same dimensionality, and allows better comparability in the scores across different outliers. In such cases, intuitive probability values may be derived more simply from the $\chi^2(d)$ distribution.

The high dimensional case is an extremely difficult one, and it is understandable that no given method will be able to solve these problems perfectly. It should also be pointed out that the iterative EM algorithm discussed in Chapter 2 will be able to discover the local directions of correlation along with outliers which have low fit value to the model. These may sometimes include weak outliers, which are not always interesting. Given that direct discovery of optimal subspaces in a given locality is much more difficult in outlier analysis, a possible line of work would be to use a two-phase approach of first finding the weak outliers, and then determining the strong ones among them by more detailed analysis. For example, it may be possible to use this pre-filtered set of weak outliers for intensive ensemble-based subspace exploration. Combining pre-filtered data points with pre-filtered high-contrast subspaces may provide an interesting direction of future exploration. A significant scope still exists for further improvement of the techniques designed in this area.

6.   Discussion of Subspace Analysis

While subspace outlier analysis seems to be the only meaningful method for high dimensional outlier detection, the approach faces a number of challenges, a lot of which are computational in nature. In the high-dimensional case, a small number of deviant subspaces may remain hidden out of a large number of possibilities. This can create unprecedented challenges for outlier analysis. The combinatorial nature of the problem necessitates the design of more efficient algorithms which can perform an ordered exploration of these spaces. In spite of the recent advances in the literature, the design of efficient algorithms for the high dimensional subspace exploration scenario remains a challenge. This is of course an inherent property of high-dimensional data, in which the curse of dimensionality impacts the results both from a qualitative and efficiency perspective.

The second challenge arises from the fact that a subspace exploration technique reports a number of different possibilities for the projections. In such cases, it remains a challenge to combine the results from these deviant subspaces, and rank the resulting outliers effectively. This is of course an opportunity as well, since the results from multiple subspaces may provide more robust outliers. Therefore, significant advancements are required in *ensemble analysis* for outlier detection.

It has been claimed in [514] as an apparently new insight, that the major reason for difficulty in high dimensional outlier analysis is not the concentration of distances, but the masking effects of the locally noisy and irrelevant nature of some of the dimensions, and that the literature has failed to discuss the impact of locally relevant dimensions. This is an incorrect assertion, since both the aspects of local feature selection (relevance) and distance concentration have been studied extensively in the literature. While it is true that noisy and irrelevant attributes mask the outliers, the observation is certainly not new, and the two factors of distance concentration and local feature relevance are closely related. The original work in [4] (and virtually every other subsequent work [289, 256, 337] on this topic) provides a pictorial illustration and a fairly detailed discussion of how (locally) irrelevant attributes mask outliers in different feature-specific views of the data. As stated in [4]: "*... by using full dimensional distance measures it would be difficult to determine outliers effectively because of the averaging behavior of the noisy and irrelevant dimensions. Furthermore, it is impossible to prune off specific features a-priori, since different points may show different kinds of abnormal patterns, each of which use different features or views.*" The ineffectiveness of *global* feature selection in high dimensional data in fact forms the motivating reason for subspace analysis, which can be considered a *local* feature selection method, or a *local* dimensionality reduction method [7, 95]. These connections of local subspace analysis to the ineffectiveness of global feature selection in high dimensional data were explicitly discussed in detail in the motivational discussion of one of the earliest works on subspace analysis [5]. At this point, these results are well known and established[3] wisdom. While it is possible to reduce the distance concentration effects by carefully calibrating the fraction of informative dimensions, such cases are (usually) not interesting for subspace analysis.

---

[3]Some of the earliest methods even refer to these classes of techniques as local dimensionality reduction [95] in order to emphasize the enhanced and differential local feature selection effect, which arises as a result of different generating mechanisms.

Distance concentration and (too many) irrelevant attributes are closely related. The interesting cases for subspace analysis (typically) show some levels of both properties. Even limited levels of distance concentration impact the effectiveness of full dimensional distance-based algorithms, and this impact is therefore important to examine in outlier analysis. It should be noted that noisy and irrelevant attributes are more likely to lead to concentration of distances. For example, for the case of uniformly distributed data, where all attributes are noisy, the concentration effect is extreme, and an outlier deviating along *a relatively small number of dimensions* will be hard to discover by full dimensional methods. In such cases, from a full dimensional distance-based or density-based perspective, all data points have almost equally good outlier scores, and this can be equivalently understood in terms of *either* locally irrelevant features or distance concentration effects. Of course, real data sets are not uniformly distributed, but *both* irrelevant features and concentration effects are present to varying degrees in different data sets. The general assumption for subspace analysis is that the addition of more dimensions often does not add *proportionally* more information for a particular outlier. The challenging outliers are often defined by the behavior of a small number of dimensions, and when the point-specific information does not increase substantially with data dimensionality, even modest concentration effects will have a negative impact on full dimensional algorithms. The more the number of irrelevant attributes, the more erroneous the computations for full-dimensional distance-based methods. An extreme example at the other end of the spectrum is where an outlier shows informative and deviant behavior in every dimension, and therefore outlier characteristics grow *stronger* with increasing dimensionality. However, in this rather uninteresting case, since the outlier shows *both* many relevant features *and* also typically does not conform to the distance concentration behavior of the remaining data, a trivial full dimensional distance-based algorithm would find it easily in most cases. In general, cases where the informative dimensions also increase significantly with data dimensionality, are not as interesting for subspace analysis because the full dimensional masking behavior becomes less prominent in this easier case. Subspace analysis does not exclude the possibility that the more obvious deviants may also be found by full dimensional analysis.

Outliers, by their very rare nature, may often be hidden in small combinations of dimensions in a high dimensional data set. Subspace analysis is interesting for such scenarios. On the other hand, when more dimensions do add (significantly) more information, then this becomes an easy case for analysis, which no longer remains interesting. In the

former case, the vast majority of noisy dimensions make all data points appear as outliers from a density-based or data sparsity perspective.

To summarize, subspace outlier analysis is one of the most challenging problems because of the rare and unusual nature of outliers. In order to design meaningful algorithms, the following principles need to be kept in mind.

- Aggregation-based methods for subspace analysis only provide very weak hints for outlier analysis as compared to clustering algorithms. A direct exploration of rare regions is possible, though it is computationally challenging because of combinatorial explosion [4]. As a result, it becomes necessary to use heuristic methods.

- Aggregation-based methods may be usable, if caution is utilized in recognizing the fact that a given subspace derived from such methods may not always include the relevant dimensions. Exclusion of relevant dimensions has more drastic effects than inclusion of many irrelevant dimensions. Where possible, subspace ensembles should be used in order to combine the weak hints derived from the different subspaces, if aggregation-based measures are used.

- The individual component of an ensemble should be designed with efficiency considerations. This is because the ability to execute the individual component more number of times within a fixed time frame, eventually provides more robustness.

7.   Conclusions and Summary

Subspace methods for outlier detection are used in cases, where the outlier tendency of a data point is diluted by the noise effects of a large number of locally non-informative dimensions. In such cases, the outlier analysis process can be sharpened significantly by searching for subspaces in which the data points deviate significantly from the normal behavior. The earliest work on subspace outlier detection used evolutionary search methods in order to determine abnormal lower dimensional projections of the data. A number of subsequent methods have also been designed for determining multiple relevant subspaces for a candidate outlier, and then combining the results from different subspaces in order to create a more robust ensemble-based ranking. It is also possible to determine the outliers in arbitrarily oriented subspaces of the data. Such methods are able to exploit the local correlations in the data in order to determine relevant outliers.

Outlier analysis is the most difficult problem among all classes of subspace analysis problems. This difficulty arises out of the rare nature

of outliers, which makes direct statistical analysis more difficult. Since subspace analysis and local feature selection are related, it is noteworthy that even for global feature selection, there are few known methods for outlier analysis, as compared to clustering and classification algorithms. The reason is simple: enough statistical evidence is often not available for the analysis of rare characteristics. Robust statistics is all about *more* data, and outliers are all about *less* data and statistical non-conformity with most of the data! Regions and subspaces containing statistical conformity tell us very little about the complementary regions of non-conformity in the particular case of high-dimensional subspace analysis, since the *potential* domain of the latter is much larger than the former. In particular, a local subspace region of the greatest aggregate conformity does not necessarily reveal anything about the rare point with the greatest statistical non-conformity.

While it is doubtful that the more difficult variations of the problem will ever be fully solved, or will work completely in all situations, it may be possible to design methods which work in many important scenarios. There are many merits in being able to design such methods, because of the numerous insights they can provide in terms of identifying the causes of abnormality. The main challenge is that outlier analysis is so brittle, that it is often impossible to make confident assertions about inferences drawn from aggregate data analysis. The issue of efficiency seems to be closely related to that of effectiveness in high dimensional outlier analysis. This is because the search process for outliers is likely to require exploration of multiple local subspaces of the data in order to ensure robustness. With increasing advances in the computational power of modern computers, there is as yet hope that this area will become increasingly tractable for analysis.

8.   Bibliographic Survey

In the context of high-dimensional data, there are two distinct lines of research, one of which investigates the *efficiency* of high dimensional outlier detection [46, 185, 467], and the other investigates the more fundamental issue of the *effectiveness* of high dimensional outlier detection [4, 273]. Unfortunately, the distinction between these two lines of work is sometimes blurred in the literature, even though these are clearly different lines of work with very different motivations. It should be noted that the methods discussed in [46, 185, 467] are all *full dimensional methods*, because outliers are defined on the basis of their full dimensional deviation. While the method of [467] uses projections for indexing, this is

used only as an approximation to improve the efficiency of the outlier detection process.

In the high-dimensional case, the efficiency of (full dimensional) outlier detection also becomes a concern, because most outlier detection methods require repeated similarity search in high dimensions in order to determine the nearest neighbors. The efficiency of these methods degrades because of two factors: (i) the computations now use a larger number of dimensions, and (ii) the effectiveness of pruning methods and indexing methods degrades with increasing dimensionality. The solution to these issues still remains unresolved in the vast similarity search literature. Therefore, it is unlikely that *significantly* more efficient similarity computations could be achieved in the context of high dimensional outlier detection, though some success has been claimed for improving the efficiency of high dimensional outlier detection in methods proposed in [46, 185, 467]. On the whole, it is unclear how these methods would compare to the vast array of techniques available in the similarity search literature for indexing high dimensional data. This chapter does *not* investigate the efficiency issue at all, because the efficiency of a *full dimensional* outlier detection technique is not important, if it does not even provide meaningful outliers. Therefore, the focus of the chapter is on methods which *re-define* the outlier detection problem in the context of lower dimensional projections. It is also noted that an angle-based outlier detection for high-dimensional data has been proposed in [269], though this method has been discussed in the chapter on extreme value analysis (Chapter 2), since this method is not a subspace exploration technique. It is also designed to find specific kinds of outliers which lie at the boundaries of the multivariate data, and is much closer in principle to other multivariate extreme value analysis methods such as depth-based and deviation-based methods.

The problem of subspace outlier detection was first proposed in [4]. In this paper, an evolutionary algorithm was proposed to discover the lower dimensional subspaces in which the outliers may exist. The method for distance-based outlier detection with subspace outlier degree was proposed in [273]. Another distance-based method for subspace outlier detection was proposed in [346]. Some methods have also been proposed for outlier analysis by randomly sampling subspaces and combining the scores from different subspaces [289, 310]. In particular, the work in [289] attempts to combine the results from these different subspaces in order to provide a more robust evaluation of the outliers. These are essentially *ensemble-based* methods, which attempt to improve detection robustness by bagging the results from analyzing different sets of features. The major challenge of these methods is that random sampling may not

work very well, when the outliers are hidden in specific subspaces of the data. The work in [256] can be considered a generalization of the broad approach in [289], where only high contrast subspaces are selected for the problem of outlier detection.

The reverse problem of finding outlying subspaces *from* specific points was studied in [499–501]. In these methods, a variety of pruning and evolutionary methods were proposed in order to speed up the search process for outlying subspaces. The work in [47] also defines the exceptional properties of outlying objects both with respect to the entire population (global properties), and also with respect to particular sub-populations to which it belongs (local properties). Both these methods provide different but meaningful insights about the underlying data. A genetic algorithm for finding the outlying subspaces in high dimensional data is provided in [500]. In order to speed up the fitness function evaluation, methods are proposed to speed up the computation of the k-nearest neighbor distance with the use of bounding strategies. A broader framework for finding outlying subspaces in high dimensional data is provided in [501]. A method which uses two-way search for finding outlying subspaces is proposed in [482]. In this method, full dimensional methods are first used to determine the outliers. Subsequently, the key outlying subspaces from these outlier points are detected and reported. A method for using rules in order to explain the context of outlier objects is proposed in [340].

A number of ranking methods for subspace outlier exploration have been proposed in [337–339]. In these methods, outliers are determined in multiple subspaces of the data. Different subspaces may either provide information about different outliers, or about the same outliers. Therefore, the goal is to combine the information from these different subspaces in a robust way in order to report the final set of outliers. The *OUTRES* algorithm proposed in [337] uses recursive subspace exploration in order to determine all the subspaces relevant to a particular data point. The outlier scores from these different subspaces are combined in order to provide a final value. A tool-kit for ranking subspace outliers has been presented in [338]. A more recent method for using multiple views of the data for subspace outlier detection is proposed in [341]. Methods for subspace outlier detection in multimedia databases were proposed in [51].

Most of the methods for subspace outlier detection perform the exploration in axis-parallel subspaces of the data. This is based on the complementary assumption that the dense regions or clusters are hidden in axis-parallel subspaces of the data. However, it has been shown in recent work that the dense regions may often be located in arbitrarily ori-

ented subspaces of the data [7]. While it has been shown in earlier work that the removal of noise (or weak outliers) improves the effectiveness of generalized subspaces clustering algorithms [7], specific techniques are also required in order to determine outliers in a way which is optimized to the data correlations. Another work in [274] provides an arbitrarily oriented solution for the generalized outlier analysis problem, which extends the correlation-analysis approach proposed in [7] to a method based on local reference sets rather than clusters.

Recently, the problem of outlier detection has also been studied in the context of dynamic data and data streams. The SPOT method was proposed in [498], which is able to determine projected outliers from high dimensional data streams. Thus approach employs a window-based time model and decaying cell summaries to capture statistics from the data stream. A set of top sparse subspaces are obtained by a variety of supervised and unsupervised learning processes. These are used in order detect the projected outliers. A multi-objective genetic algorithm is employed for finding outlying subspaces from training data.

The problem of high dimensional outlier detection has also been extended to other application-specific scenarios such as astronomical data [213], uncertain data [23], transaction data [210] and supervised data [513]. In the uncertain scenario, high dimensional data is especially challenging, because the noise in the uncertain scenario greatly increases the sparsity of the underlying data. Furthermore, the level of uncertainty in the different attributes is available. This helps decide the importance of different attributes for outlier detection purposes. Subspace methods for outlier detection in uncertain data are proposed in [23]. Supervised methods for high-dimensional outlier detection are proposed in [513]. In this case, a small number of examples are presented to user of the outliers. These are then used in order to learn the critical projections which are relevant to the outlierness of an object. The learned information is then leveraged in order to determine the relevant outliers in the underlying data.

9.   Exercises

1. Which of the following data points is an outlier in some well chosen two-dimensional projection: { $(1, 8, 7)$, $(2, 8, 8)$, $(5, 1, 2)$, $(4, 1, 1)$, $(3, 1, 8)$ }

2. Download the *Arrythmia* data set from the UCI Machine Learning Repository [169]. Write a computer program to determine all distance-based outliers in different 2-dimension projections. Are the outliers the same in different projections?

3. In the *Arrythmia* data set mentioned in the previous exercise, examine the *Age*, *Height* and *Weight* attributes of the *Arrythmia data* set both independently and in combination. Draw a scatter plot of each of the 1-dimensional distributions and different 2-dimensional combinations. Can you visually see any outliers?

4. Write a computer program to determine the subspace outlier degree of each data point in the *Arrythmia* data set for all 1-dimensional projections and 2-dimensional projections. Which data points are declared outliers?

5. Write a computer program to perform subspace sampling of the *Arrythmia* data set, using the approach of [289] by sampling 2-dimensional projections. How many subspaces need to be sampled in order to robustly identify the outliers found in Exercise 2 over different executions of your computer program.

6. Consider a data set with d-dimensions, in which exactly 3 specific dimensions behave in an abnormal way with respect to an observation. How many minimum number of random subspaces of dimensionality $(d/2)$ will be required in order to include all 3 dimensions in the subspace with probability at least 0.99? Plot the number of required samples for different values of $d > 6$.

Chapter 6

SUPERVISED OUTLIER DETECTION

"True, a little learning is a dangerous thing, but it still beats total ignorance." – Abigail van Buren

1.   Introduction

The discussion in the previous chapters focussed on the problem of unsupervised outlier detection in which no prior information is available about the abnormalities in the data. In such scenarios, many of the anomalies found correspond to noise, and may not be of any interest to an analyst. It has been observed [284, 315, 440] in diverse applications such as system anomaly detection, financial fraud, and web robot detection that *the nature of the anomalies is often highly specific to particular kinds of abnormal activity in the underlying application.* In such cases, unsupervised outlier detection methods may often discover noise, which may not be specific to that activity, and therefore may not also be of any interest to an analyst. The goal of supervised outlier detection is to incorporate application-specific knowledge into the outlier analysis process, so as to obtain more meaningful anomalies with the use of learning methods. Because of the rare nature of anomalies, such data is often limited, and it is hard to create robust and generalized models on this basis. Nevertheless, the general observation has been that the incorporation of learning methods can significantly improve the robustness of the outlier analysis process. *The general recommendation for outlier analysis is to always use supervision where possible.*

In most real data domains, some examples of normal or abnormal data may be available. This is referred to as *training data*, and can be used to create a *classification model*, which distinguishes between normal and anomalous instances. The problem of classification has been widely

studied in its own right, and numerous algorithms are available in the literature [146] for creating supervised models from training data. In many cases, different kinds of abnormal instances may be available, in which case the classification model may be able to distinguish between them. For example, in an intrusion scenario, different kinds of intrusion anomalies are possible, and it may be desirable to distinguish among them.

So how is the supervised outlier detection problem different from classification? The supervised outlier detection problem may be considered a very difficult special case (or variation) of the classification problem, depending upon the following possibilities, which may be present either in isolation or in combination.

- *Class Imbalance:* Since outliers are defined as rare instances in the data, it is natural that the distribution between the normal and rare class will be very skewed. From a practical perspective, this implies that the optimization of classification accuracy may not be meaningful, especially since the misclassification of positive (outlier) instances is less desirable than the misclassification of negative (normal) instances. In other words, false positives are more acceptable than false negatives. This leads to cost-sensitive variations of the classification problem, in which the objective function for classification is changed.

- *Contaminated Normal Class Examples (Positive-Unlabeled Class Problem):* In many real scenarios, the data may originally be present in unlabeled form, and manual labeling is performed for annotation purposes. In such cases, only the positive class is labeled, and the remaining "normal" data contains some abnormalities. This is natural in large scale applications such as the web and social networks, in which the sheer volume of the underlying data makes contamination of the normal class more likely. For example, consider a social networking application, in which it is desirable to determine spam in the social network feed. A small percentage of the documents may be spam. In such cases, it may be possible to recognize and label some of the documents as spam, but many spam documents may remain in the examples of the normal class. Therefore, the "normal" class may also be considered an unlabeled class. In practice however, the unlabeled class is predominantly the normal class, and the anomalies in it may be treated as contaminants. The classification models need to be built to account for this. Technically, this case can be considered a form of partial supervision [306], though it can also be treated as a

difficult special case of full supervision, in which the normal class is more noisy and contaminated. Standard classifiers can be used on the positive-unlabeled version of the classification problem, as long as the relative frequency of contaminants is not extreme. In cases where the unlabeled class does not properly reflect the distribution in the test instances, the use of such unlabeled classes can actually *harm* classification accuracy [301].

A different flavor of incomplete supervision refers to missing training data about an *entire* class, rather than imperfect or noisy labels. This case is discussed below.

- *Partial Training Information (Semi-supervision or novel class detection):* In many applications, examples of one or more of the anomalous classes may not be available. For example, in an intrusion detection application, one may have examples of the normal class, and *some* of the intrusion classes, as new kinds of intrusions arise with time. In some cases, examples of one or more normal classes are available. A particularly commonly studied case is the *one class variation*, in which only examples of the normal class are available. The only difference between this extreme case and the unsupervised scenario is that the examples of the normal class are typically guaranteed to be free of outliers. In many applications, this is a very natural consequence of the extreme rarity of the outlier. For example, in a bio-terrorist attack scenario, no examples of anomalous classes may be available, since no such event may have occurred in the first place. Correspondingly, the examples of the training class are also guaranteed to be free of outliers. This particular special case, in which the training data contains only normal classes, is much closer to the unsupervised version of the outlier detection problem. This will be evident from the subsequent discussion in the chapter.

It is evident that most of the above cases are either a special case, or a variant of the classification problem, which provides different challenges. Furthermore, it is possible for some of these conditions to be present in combination. For example, in an intrusion detection application, labeled data may be available for some of the intrusions, but no labeled information may be available for other kinds of intrusions. Thus, this scenario requires the determination of both rare classes and novel classes. In some cases, rare class scenarios can be reduced to partially supervised scenarios, when only the rare class is used for training purposes. Therefore, the boundaries between these scenarios are often blurred in real applications. Nevertheless, since the techniques for the different scenar-

ios can usually be combined with one another, it is easier to discuss each of these challenges separately. Therefore, a section will be devoted to each of the aforementioned variations of the supervised outlier detection problem.

The discussion in this chapter will be focussed on more generic forms of multidimensional data, rather than specific kinds of data such as temporal and spatial data. This is because the general principles of supervised outlier detection are often independent of specific data type and can be easily generalized to more complex data domains. The goal of this chapter is to provide an understanding of how classification methods need to be *modified* in order to address the challenges of supervised outlier analysis. Therefore, a working knowledge of the classification problem is assumed [146] for the purposes of this chapter. Furthermore, a section on supervised methods will be included in many of the chapters which address the more complex data types.

A particular form of supervision is *active learning*, when human experts may intervene during the outlier detection process in order to identify relevant instances. Very often, active learning may be accomplished by providing an expert with candidates for outliers, which are followed by the expert explicitly labeling these pre-filtered examples. In such cases, *label acquisition* is combined with *model construction* in order to progressively incorporate more human knowledge into the outlier analysis process. Such a human-computer cooperative approach can sometimes provide more effective results than automated techniques.

Paucity of training data is a common problem, when the class distribution is imbalanced. Even in a modestly large training data set, only a small number of rare instances may be available. Typically, it may be expensive to acquire examples of the rare class. Imbalanced class distributions could easily lead to training algorithms which show differentially overfitting behavior. In other words, the algorithm may behave robustly for the normal class, but may overfit the rare class. Therefore, it is important to design the training algorithms, so that overfitting is avoided.

This chapter is organized as follows. The next section will discuss the problem of rare-class detection in the fully supervised scenario. The semi-supervised case of classification with positive and unlabeled data will be studied in section 3. Section 4 will discuss the problem of novel class detection. This is also a form of semi-supervision, though it is of a different kind. Methods for outlier detection with human supervision are addressed in section 5. The conclusions and summary are presented in section 6.

2.   The Fully Supervised Scenario: Rare Class Detection

The problem of rare-class detection or *class imbalance* is a common one in the context of supervised outlier detection. The straightforward use of evaluation metrics and classifiers which are not cognizant of this class imbalance may lead to very surprising results. For example, consider a medical application in which it is desirable to identify tumors from medical scans. In such cases, 99% of the instances may be normal, and only 1% are abnormal.

Consider the trivial classification algorithm, in which every instance is labeled as normal without even examining the feature space. Such a classifier would have a very high absolute accuracy of 99%, but would not be very useful in the context of a real application. In fact, many forms of classifiers (which are optimized for absolute accuracy) may show a degradation to the trivial classifier. For example, consider a k-nearest neighbor classifier, in which the majority class label in the neighborhood is reported as the relevant class label. Because of the inherent bias in the class distribution, the majority class may very often be normal even for abnormal test instances. Such an approach fails because it does not account for the *relative* behavior of the test instances with respect to the original class distribution. For example, if 49% of the training instances among the k-nearest neighbors of a test instance are anomalous, then that instance is *much* more likely to be anomalous *relative* to its original class distribution. By allowing changes to the classification criterion, such as reporting non-majority anomalous classes as the relevant label, it is possible to improve the classification accuracy of anomalous classes. However, the *overall* classification accuracy may degrade. Of course, the question arises whether the use of measures such as overall classification accuracy is meaningful in the first place. Therefore, the issue of *evaluation* and *model construction* are closely related in the supervised scenario. The first step is to identify how the rare class distribution relates to the objective function of a classification algorithm, and the algorithmic changes required in order to incorporate the modifications to the modeling assumptions.

There are two primary classes of algorithms which are used for handling class imbalance:

■ *Cost Sensitive Learning:* The objective function of the classification algorithm is modified in order to weight the errors in classification differently for different classes. Classes with greater rarity have higher costs. Typically, this approach requires algorithm-

specific changes to different classifier models in order to account for costs.

- *Adaptive Re-sampling:* The data is re-sampled so as to magnify the *relative proportion* of the rare classes. Such an approach can be considered an *indirect form* of cost-sensitive learning, since data re-sampling is equivalent to implicitly assuming higher costs for misclassification of rare classes.

Both these methodologies will be discussed in this section. For the case of the cost-sensitive problem, it will also be discussed how classification techniques can be heuristically modified in order to approximately reflect costs. A working knowledge of classification methods is assumed in order to understand the material in this section. The reader is also referred to [146] for a description of the different types of classifiers.

For the discussion in this section, it is assumed that the training data set is denoted by $\mathcal{D}$, and the labels are denoted by $L = \{1, \ldots k\}$. Without loss of generality, it can be assumed that the normal class is indexed by 1. The ith record is denoted by $\overline{X_i}$, and its label l_i is drawn from L. The number of records belonging to the ith class are denoted by N_i, and $\sum_{i=1}^{k} N_i = N$. The class imbalance assumption implies that $N_1 >> N - N_1$. While imbalances may exist between other anomalous classes too, the major imbalance occurs between the normal and the anomalous classes.

2.1   Cost Sensitive Learning

In cost sensitive learning, the goal is to learn a classifier, which maximizes the weighted accuracy over the different classes. The *misclassification cost* of the ith class is denoted by c_i. Some models [145] use a $O(k \times k)$ cost matrix to represent the full spectrum of misclassification behavior. In such models, the cost is dependent not only on the class identity of the misclassified instance, but is also dependent on the specific class label *to which* it is misclassified. A simpler model is introduced here, which is more relevant to the rare class detection problem. Here the cost only depends on the origin class, and not on a combination of the origin and destination class. The goal of the classifier is to learn a training model which minimizes the *weighted misclassification rate*.

The choice of c_i is picked in an application specific manner, though numerous heuristics exist to pick the costs in an automated way. The work in [497] proposes methods to learn the costs directly in a data driven manner. Other simpler heuristic rules are used often in many practical scenarios. For example, by choosing the value of c_i to be proportional to $1/N_i$, the *aggregate* impact of the instances of each class on

the weighted misclassification rate is the same, in spite of the imbalance between the classes. Such methods are at best rule-of-thumb techniques for addressing imbalance, though more principled methods also exist in the literature. Many such methods will be discussed in this chapter.

2.1.1 MetaCost: A Relabeling Approach.

A general framework known as MetaCost [145] uses a *relabeling* approach to classification. This is a *meta-algorithm*, which can be applied to any classification algorithm. In this method, the idea is to relabel some of the training instances in the data, by using the costs, so that normal training instances, which have a reasonable probability of classifying to the rare class are relabeled to that rare class. Of course, rare classes may also be relabeled to a normal class, but the cost-based approach is *intended to* make this less likely. Subsequently, a classifier can be used on this more balanced training data set. The idea is to use the costs in order to move the decision boundaries in a cost-sensitive way, so that normal instances have a greater chance of misclassification than rare instances, and the *expected misclassification cost* is minimized.

In order to perform the relabeling, the classifier is applied to each instance of the training data and its classification prediction is combined with costs for re-labeling. Then, if a classifier predicts class label i with probability $p_i(\overline{X})$ for the data instance $\overline{X}$, then the expected misclassification cost of the prediction of $\overline{X}$, *under the hypothesis that it truly belonged to* r, is given by $\sum_{i \neq r} c_i \cdot p_i(\overline{X})$. Clearly, one would like to minimize the expected misclassification cost of the prediction. Therefore, the *MetaCost* approach tries different hypothetical classes for the training instance, and relabels it to the class which minimizes the expected misclassification cost. A key question arises as to how the probability $p_i(\overline{X})$ may be estimated from a classifier. This probability clearly depends upon the specific classifier which is being used. While some classifiers explicitly provide a probability score, not all classifiers provide such probabilities. The work in [145] proposes a bagging approach [80] in which the training data is sampled with replacement (bootstrapping), and a model is repeatedly constructed on this basis. The training instances are repeatedly classified with the use of such a bootstrap sample. The fraction of predictions (or votes) for a particular class across different training data samples are used as the classification probabilities.

The challenge of such an approach is that relabeling training data is always somewhat risky, especially if the bagged classification probabilities do not reflect intrinsic classification probabilities. In fact, each bagged classification model-based prediction is *highly correlated* to the

others (since they share common training instances), and therefore the aggregate estimate is not a true probability.

In practice, the estimated probabilities are likely to be *very* skewed towards one of the classes, which is typically the normal class. For example, consider a scenario in which a rare class instance (with global class distribution of 1%) is present in a local region with 15% concentration of rare class instances. Clearly, this rare instance shows informative behavior in terms of *relative* concentration of the rare class in the locality of the instance. A vanilla 20-nearest neighbor classifier will virtually *always*[1] classify this instance to a normal class in a large bootstrapped sample. This situation is not specific to the nearest neighbor classifier, and is likely to occur in many classifiers, when the class distribution is very skewed. For example, an unmodified Bayes classifier will usually assign a lower probability to the rare class, because of its much lower *a-priori* probability, which is factored into the classification. Consider a situation, where a hypothetically perfect Bayes classifier has a prior probability of 1% and a posterior probability of 30% for the correct classification of a rare class instance. Such a classifier will typically assign far fewer than 30% of the votes to the rare class in a bagged prediction, especially[2] when large bootstrap samples are used. In such cases, the normal class will win every time in the bagging because of the prior skew. This means that the bagged classification probabilities can sometimes be close to 1 for the normal class in a skewed class distribution.

This suggests that the effect of cost weighting can sometimes be overwhelmed by the erroneous skews in the probability estimation attained by bagging. In this particular example, even with a cost ratio of 100 : 1, the rare class instance will be wrongly relabeled to a normal class. This moves the classification boundaries in the opposite direction of what is desired. In fact, in cases where the unmodified classifier degrades to a trivial classifier of always classifying to the normal class, the expected misclassification cost criterion of [145] will result in relabeling all rare class instances to the normal class, rather than the intended goal of selective relabeling in the other direction. In other words, relabeling

---

[1] The probability can be (approximately) computed from a binomial distribution to be at least equal to $\sum_{i=0}^{9} \binom{20}{i} \cdot 0.15^i \cdot 0.85^{20-i}$ and is greater than 0.999.

[2] The original idea of bagging was not designed to yield class probabilities [80]. Rather, it was designed to perform robust prediction for instances, where either class is an almost equally good fit. In cases, where one of the classes has a "reasonably" higher (absolute) probability of prediction, the bagging approach will simply boost that probability to almost 1, when counted in terms of the number of votes. In the rare class scenario, it is expected for unmodified classifiers to misclassify rare classes to normal classes with "reasonably" higher probability.

may result in a further *magnification* of the errors arising from class skew. This leads to degradation of classification accuracy, *even from a cost-weighted perspective.*

In the previous example, if the *fraction* of the 20-nearest neighbors belonging to a class are used as its probability estimate for relabeling, then much more robust results can be obtained with *MetaCost*. Therefore, the effectiveness of *MetaCost* depends on the quality of the probability estimate used for re-labeling. Of course, if good probability estimates are directly available from the training model in the first place, then a test instance may be directly predicted using the expected misclassification cost, rather than using the indirect approach of trying to "correct" the training data by re-labeling. This is the idea behind weighting methods, which will be discussed in the next section.

2.1.2 Weighting Methods.

Most classification algorithms can be modified in natural ways to account for costs with some simple modifications. The primary driving force behind these modifications is to implicitly treat each training instance with a weight, where the weight of the instance corresponds to its misclassification cost. This leads to a number of simple modifications to the underlying classification algorithms. In most cases, the weight is not used explicitly, but the underlying classification model is changed to reflect such an implicit assumption. Some methods have also been proposed in the literature [496] in order to incorporate the weights explicitly into the learning process. In the following, a discussion is provided about the natural modifications to the more common classification algorithms.

Bayes Classifier The modification of the Bayes classifier provides the simplest case for cost-sensitive learning. In this case, changing the weight of the example only changes the a-priori probability of the class, and all other terms within the Bayes estimation remain the same. Therefore, this is equivalent to multiplying the Bayes probability in the unweighted case with the cost, and picking the largest one. Note that this is the same criterion that is used in *MetaCost*, though the latter uses this criterion for *relabeling* training instances, rather than predicting test instances. When good probability estimates are available from the Bayes classifier, the test instance can be directly predicted in a cost-sensitive way.

Proximity-based Classifiers In nearest neighbor classifiers, the classification label of a test instance is defined to be the majority class from its k nearest neighbors. In the context of *cost-sensitive* classification, the *weighted* majority label is reported as the relevant one, where the

weight of an instance from class i is denoted by c_i. Thus, fewer examples of the rare class need to be present in a neighborhood of a test instance, in order for it to be reported as the relevant one. In a practical implementation, the number of k-nearest neighbors for each class can be multiplied with the corresponding cost for that class. The majority class is picked *after* the weighting process. A discussion of methods for k-nearest neighbor classification in the context of data classification may be found in [506].

Rule-based Classifiers   In rule-based classifiers, frequent pattern mining algorithms may be adapted to determine the relevant rules at a given level of support and confidence. A rule relates a condition in the data (eg. ranges on numeric attributes) to a class label. The support of a rule is defined as the number of training instances which are relevant to that rule. The confidence of a rule is the fractional probability that the training instance belongs to the class on the right hand side, if it satisfies the conditions on the left-hand side. Typically, the data is first discretized, and all the relevant rules are mined from the data, at pre-specified levels of support and confidence. These rules are then prioritized based on the underlying confidence (and sometimes also the support). For a given test instances, all the relevant rules are determined, and the results from different rules can be combined in a variety of ways (eg. majority class from relevant rules, top matching rule etc.) in order to yield the final class label.

Such an approach is not difficult to adapt to the cost-sensitive case. The main adaptation is that the weights on the different training examples need to be used during the computation of measures such as the support or the confidence. Clearly, when rare examples are weighted more heavily, the confidence of a rule will be much higher, when its right hand side corresponds to a rare class because of the weighting. This will result in the selective emphasis of rules corresponding to rare instances. Some methods for using rule-based methods in imbalanced data classification are proposed in [245, 247].

2.1.3   Decision Trees.   In decision trees, the training data is recursively partitioned, so that the instances of different classes are successively separated out at lower levels of the tree. The partitioning is performed by using conditions on one or more features in the data. Typically, the split criterion uses the various entropy measures such as the gini-index for deciding the choice of attribute and the position of the split. For a node containing a fraction of instances of different classes denoted by $p_1 \dots p_k$, its gini-index is denote by $1 - \sum_{i=1}^{k} p_i^2$.
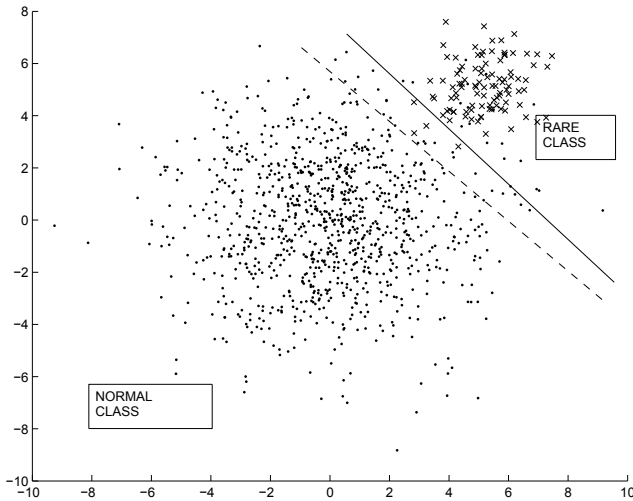
Figure 6.1. Optimal hyperplanes will change because of weighting of examples

Better separations of different classes lead to lower gini-index. The split attribute and partition point is decided as one which minimizes the gini-index of the children nodes. By using costs as weights for the instances, the computation of the gini-index will be impacted so as to selectively determine regions of the data containing higher proportions of the rare class. Some examples of cost-sensitive decision trees are discussed in [450, 463].

2.1.4 SVM Classifier. SVM classifiers work by learning hyperplanes, which optimally separate the two classes in order to minimize the expected error. Thus, SVM classifiers can be modeled as an optimization problem, where the goal is to learn the coefficients of the underlying hyperplane. For example, a two-class example has been illustrated in Figure 6.1. The optimal separator hyperplane for the two classes is illustrated in the same figure with the solid line. However, it is possible to change the optimization model by incorporating weights (or costs) into the optimization problem. This shifts the decision boundary, so as to allow erroneous classification of a larger number of normal instances, while correctly classifying more rare instances. The result would be a reduction in the overall classification accuracy, but an increase in the cost-sensitive accuracy. For example, in the case of Figure 6.1, the optimal separator hyperplane would move from the solid line to the dotted line in the figure. The issue of class-boundary re-alignment for SVMs in the context of imbalanced data sets has been explored in detail in

[443, 470]. While these models are not designed with the use of example re-weighting, they achieve similar goals by using class-biased penalties during the SVM model creation.

2.2   Adaptive Re-sampling

In adaptive re-sampling, the different classes are differentially sampled in order to enhance the impact of the rare class on the classification model. Sampling can be performed either with or without replacement. Either the rare class can be oversampled, or the normal class can be under-sampled, or both. The classification model is learned on the re-sampled data. The sampling probabilities are typically chosen in proportion to their misclassification costs. This enhances the proportion of the rare costs in the sample used for learning. It has generally been observed [143], that under-sampling has a number of advantages over over-sampling. When under-sampling is used, the sampled training data is much smaller than the original data set. In some variations, all instances of the rare class are used in combination with a small sample of the normal class [106, 278]. This is also referred to as *one-sided selection*. Under-sampling also has the advantage of being efficient without losing too much information, because:

- The model construction phase for a smaller training data set requires much less time.

- The normal class is less important for modeling purposes, and most of the rare class is included for modeling. Therefore, the discarded instances do not take away too much from the modeling effectiveness.

2.2.1   Relation between weighting and sampling.   Since cost-sensitive learning can be logically understood as methods which weigh examples differently, a question arises as how these methods relate to one another. Adaptive re-sampling methods can be understood as methods which sample the data in proportion to their weights, and then treat all examples equally. From a practical perspective, this may often lead to similar models in the two cases, though sampling methods may throw away some of the relevant data. It should also be evident that a direct weight-based technique retains more information about the data, and is therefore likely to be more accurate. This seems to be the case from many practical experiences with real data [102]. On the other hand, adaptive re-sampling has distinct *efficiency* advantages because it works with a much smaller data set. For example, for a data set containing 1% of labeled anomalies, it is possible for a re-sampling technique to work

effectively with 2% of the original data, when the data is re-sampled into an equal mixture of the normal and anomalous classes. This translates to a performance improvement of a factor of 50.

2.2.2   Synthetic Over-sampling: SMOTE.   Over-sampling methods are also used in the literature, though less frequently so than under-sampling. One of the problems of over-sampling the minority class is that a larger number of samples with replacement leads to repeated samples of the same record. This could lead to over-fitting, and does not necessarily help the effectiveness of the classifier. In other to address this issue, it was suggested [103] that synthetic over-sampling could be used to create the over-sampled examples in a way which provides better effectiveness. The *SMOTE* approach works as follows. For each minority instance, its k nearest neighbors are found. Then, depending upon the level of over-sampling required, a fraction of them are chosen randomly. A synthetic data example is generated on the line segment connecting that minority example to its nearest neighbor. The exact position of the example is chosen uniformly at random along the line segment. The *SMOTE* algorithm has been shown to provide more robust over-sampling than a vanilla over-sampling approach. This approach forces the decision region of the re-sampled data to become more general than one in which only members from the rare classes in the *original* training data are over-sampled.

2.2.3   One Class Learning with Positive Class.   It is possible to take adaptive re-sampling to its logical extreme by not including any examples of the normal class. This artificially transforms the problem to the semi-supervised scenario, though the nature of the semi-supervision is quite different from naturally occurring scenarios. In most natural forms of semi-supervision, the positive class is missing, and copious examples of the normal class may be available. Here the normal class examples are removed from the data. This problem is also different from the positive-unlabeled classification problem. Such a problem may sometimes occur naturally in scenarios where the background class is too diverse or noisy to be sampled in a meaningful way.

In such cases, unsupervised models can be constructed on the subset of the data corresponding to the positive class. The major difference is that *higher fit* of the data to the positive class corresponds to greater outlier scores. This is the reverse of what is normally performed in outlier detection. The assumption is that the representative data contains only anomalies, and therefore outliers are more likely to be similar to this data. Proximity-based classifiers are very natural to construct in the

one-class scenario, since the propensity of a test instance to belong to a class can be naturally modeled in terms of distances.

In the case of SVM classifiers, it is possible to create a two-class distribution by using the origin as one of the classes [396]. Typically, a kernel function is used in order to transform the data into a new space in which the dot product corresponds to the value of the kernel function. In such a case, an SVM classifier will naturally create a hyperplane which separates out the combination of features which describe the one class in the data. However, the strategy of using the origin as the second class in combination with a feature transformation is not necessarily generic and may not work well in all data domains. This differential behavior across different data sets has already been observed in the literature. In some cases, the performance of vanilla one-class SVM methods is quite poor, without careful changes to the model [382]. Other one-class methods for SVM classification are discussed in [250, 323, 382, 445].

2.2.4   Ensemble Techniques.   A major challenge of undersampling is the loss of the training data, which can have a detrimental effect on the quality of the classifier. A natural method to improve the quality of the prediction is to use ensemble techniques, in which the data instances are repeatedly classified with different samples, and then the majority vote is used for predictive purposes. In many of these methods, all instances from the rare class are used, but the majority class is under-sampled [106, 312]. Therefore, the advantages of selective sampling may be retained without a significant amount of information loss from the sampling process. In addition, a special kind of ensemble known as the *sequential ensemble* has also been proposed in [312]. In the sequential ensemble, the choice of the majority class instances picked in a given iteration depends upon the behavior of the classifier during previous iterations. Specifically, only majority instances which are correctly classified by the classifier in a given iteration are not included in future iterations. The idea is to reduce the redundancy in the learning process, and improve the overall robustness of the ensemble. Note that this is a *supervised* sequential ensemble, and is exactly analogous to the sequential ensemble method introduced in Chapter 1 for general-purpose outlier analysis.

2.3   Boosting Methods

Boosting methods are commonly used in classification in order to improve the classification performance on difficult instances of the data. The well known *Adaboost* algorithm [394] works by associating each training example with a *weight*, which is updated in each iteration,

depending upon the results of the classification in the last iteration. Specifically, instances which are misclassified, are given higher weights in successive iterations. The idea is to give higher weights to "difficult" instances which may lie on the decision boundaries of the classification process. The overall classification results are computed as a combination of the results from different rounds. In the tth round, the weight of the ith instance is $D_t(i)$. The algorithm starts off with equal weight of $1/N$ for each of the N instances, and updates them in each iteration. In practice, it is always assumed that the weights are normalized in order to sum to 1, though the approach will be described below in terms of (unscaled) relative weights for notational simplicity. In the event that the ith iteration is misclassified, then its (relative) weight is increased to $D_{t+1}(i) = D_t(i) \cdot e^{\alpha_t}$, whereas in the case of a correct classification, the weight is decreased to $D_{t+1}(i) = D_t(i) \cdot e^{-\alpha_t}$. Here α_t is chosen as the function $(1/2) \cdot \ln((1 - \epsilon_t)/\epsilon_t)$, where ϵ_t is the fraction of incorrectly predicted instances on a weighted basis. The final result for the classification of a test instance is a weighted prediction over the different rounds, where α_t is used as the weight for the tth iteration.

In the imbalanced and cost-sensitive scenario, the *AdaCost* method has been proposed [158], which can update the weights based on the cost of the instances. In this method, instead of updating the misclassified weights for instance i by the factor e^{α_t}, they are instead updated by $e^{\beta_-(c_i) \cdot \alpha_t}$, where c_i is the cost of the ith instance. Note that $\beta_-(c_i)$ is a function of the cost of the ith instance and serves as the "adjustment" factor, which accounts for the weights. For the case of correctly classified instances, the weights are updated by the factor $e^{-\beta_+(c_i) \cdot \alpha_t}$. Note that the adjustment factor is different depending upon whether the instance is correctly classified. This is because for the case of costly instances, it is desirable to increase weights more than less costly instances in case of misclassification. On the other hand, in cases of correct classification, it is desirable to reduce weights less for more costly instances. In either case, the adjustment is such that costly instances get relatively higher weight in later iterations. Therefore $\beta_-(c_i)$ is a non-decreasing function with cost, whereas $\beta_+(c_i)$ is a non-increasing function with cost. A different way to perform the adjustment would be to use the same exponential factor for weight updates as the original *Adaboost* algorithm, but this weight is further multiplied with the cost c_i [158], or other non-decreasing function of the cost. Such an approach would also provide higher weights to instances with larger costs. The use of boosting in weight updates has been shown to significantly improve the effectiveness of the imbalanced classification algorithms.

Boosting methods can also be combined with synthetic oversampling techniques. An example of this is the *SMOTEBoost* algorithm, which combines synthetic oversampling with a boosting approach. A number of interesting comparisons of boosting algorithms are presented in [246, 248]. In particular, an interesting observation in [248] is that the effectiveness of the boosting strategy is dependent upon the quality of the learner that it works with. When the boosting algorithm starts off with a weaker algorithm to begin with, the final (boosted) results are also not as good as those derived by boosting a stronger algorithm.

3.   The Semi-Supervised Scenario: Positive and Unlabeled Data

In many data domains, the positive class may be easily identifiable, though examples of the negative class may be much harder to model simply because of their diversity and inexact modeling definition. Consider for example, a scenario where it is desirable to classify or collect all documents which belong to a rare class. In many scenarios, such as the case of web documents, the types of the documents available are too diverse, and it is hard to define a representative negative sample of documents from the web.

This leads to numerous challenges at the *data acquisition stage*, where it is unknown, what kinds of negative examples one might collect for contrast purposes. The problem is that the universe of instances in the negative class is rather large and diverse, and the collection of a representative sample may be difficult. For very large scale collections such as the web and social networks [493], this scenario is quite common. A number of methods are possible for negative data collection, none of which are completely satisfactory in terms of being *truly representative* of what one might encounter in a real application. For example, for web document classification, one simple option would be to simply crawl a random subset of documents off the web. Nevertheless, such a sample would contain contaminants which do belong to the positive class, and it may be hard to create a purely negative sample, unless a significant amount of effort is invested in creating a clean sample. The amount of human effort involved in human labeling in rare class scenarios is especially high because the vast majority of examples are negative, and a manual process of filtering out the positive examples would be too slow and tedious. Therefore, a simple solution is to use the sampled background collection as the unlabeled class for training, but this may contain positive contaminants. This could lead to two different levels of challenges:

- The contaminants in the negative class can reduce the effectiveness of a classifier, though it is still better to use the contaminated training examples rather than completely discard them.

- The collected training instances for the unlabeled class may not reflect the true distribution of documents. In such cases, the classification accuracy may actually be *harmed* by using the negative class [301].

A number of methods have been proposed in the literature for this variant of the classification problem, which can address the aforementioned issues.

While some methods in the literature treat this as a new problem which is distinct from the fully supervised classification problem [306], other methods [152] recognize this problem as a noisy variant of the classification problem, to which traditional classifiers can be applied with some modifications. An interesting and fundamental result proposed in [152] is that the accuracy of a classifier trained on this scenario differs by only a constant factor from the true conditional probabilities of being positive. The underlying assumption is that the labeled examples in the positive class are picked randomly from the positive examples in the combination of the two classes. These results provides strong support for the view that learning from positive and unlabeled examples is essentially equivalent to learning from positive and negative examples.

There are two broad classes of methods which can be used in order to address this problem. In the first class of methods, heuristics are used in order to identify training examples which are negative. Subsequently, a classifier is trained on the positive examples, together with the examples, which have already been identified to be negative. A less common approach is to assign weights to the unlabeled training examples [293, 306]. The second case is a special one of the first, in which each weight is chosen to be binary. It has been shown in the literature [307], that the second approach is superior. An SVM approach is used in order to learn the weights. The work in [507] uses the weight vector in order to provide robust classification estimates.

3.1 Difficult Cases and One-Class Learning

While the use of the unlabeled class provides some advantage to classification in most cases, this is not always true. In some scenarios, the unlabeled class in the training data reflects the behavior of the negative class in the test data very poorly. In such cases, it has been shown, that the use of the negative class actually *degrades* the effectiveness of classifiers. In such cases, it has been shown in [301] that the use of one-

class learners provides more effective results than the use of a standard classifier. Thus, in such situations, it may be better to simply discard the training class examples, which do not truly reflect the behavior of the test data. Most of the one-class SVM classifiers discussed in the previous section can be used in this scenario.

4. The Semi-Supervised Scenario: Novel Class Detection

The previous section discussed cases, where it is difficult to obtain a clean sample of normal data, when the background data is too diverse or contaminated. A more common situation in the context of outlier detection is one in which no training data is available about one or more of the anomalous classes. Such situations can arise, when the anomalous class is so rare that it may be difficult to collect concrete examples of its occurrence, even when it is recognized as a concrete possibility. Some examples of such scenarios are as follows:

- In a bio-terrorist attack application, it may be easy to collect normal examples of environmental variables, but no explicit examples of anomalies may be available, if an attack has never occurred.

- In an intrusion or viral attack scenario, many examples of normal data and previous intrusions or attacks may be available, but new forms of intrusion may arise over time.

This is truly a semi-supervised version of the problem, since training data is available about some portions of the data, but not others. Therefore, such scenarios are best addressed with a combination of supervised and unsupervised techniques. It is also important to distinguish this problem from one-class classification, in which instances of the *positive class* are available. In the one-class classification problem, it is desirable to determine other examples, which are as *similar* as possible to the training data, whereas in the novel class problem, it is desirable to determine examples, which are as *different* as possible from the training data.

In cases, where only examples of the normal class are available, the only difference from the unsupervised scenario is that the training data is guaranteed to be free of outliers. The specification of normal portions of the data makes the determination of further outliers easier, because this data can be used in order to construct a model of what the normal data looks like. Another distinction between unsupervised outlier detection and one-class novelty detection, is that novelties are often defined in a temporal context, and eventually become a normal part of the data.

4.1   One Class Novelty Detection

Since the novel-class detection problem is closely related to the one-class problem in cases where only the normal class is specified, it is natural to question whether it is possible to adapt some of the one-class detection algorithms to this scenario. The major difference in this case is that it is desirable to determine classes which are as *different* as possible from the specified training class. This is a more difficult problem, because a data point may be different from the training class in several ways. If the training model is not exhaustive in describing the corresponding class, it is easy for mistakes to occur.

For example, nearest neighbor models are easy to adapt to the one class scenario. In the one-class models discussed in the previous section, it is desirable to determine data points which are as *close* as possible to the training data. In this case, the opposite is desired, where it is desirable to determine data points which are as *different* as possible from the specified training data. This is of course no different from the unsupervised methods for creating proximity-based outlier detection methods. *In fact, any of the unsupervised models for outlier detection can be used in this case.* The major difference is that the training data is guaranteed to contain only the normal class, and therefore the outlier analysis methods are likely to be more robust. Strictly speaking, when only examples of the normal class are available, the problem is hard to distinguish from the unsupervised version of the problem, at least from a methodological point of view. From a *formulation* point of view, the training and test records are not distinguished from one another in the unsupervised case (any record can be normal or an anomaly), whereas the training (only normal) and test records (either normal or anomaly) are distinguished from one another in the semi-supervised case.

One class SVM methods have also been adapted to novelty detection [397]. The main difference from *positive example training-based* one-class detection is that the class of interest lies on the *opposite* side of the separator as the training data. Some of the one-class methods such as SVMs are unlikely to work quite as well in this case. This is because a one-class SVM may really only be able to model the class present in the training data (the normal class) well, and may not easily be able to design the best separator for the class which is most *different* from the normal class. Typically, one-class SVMs use a kernel-based transformation along with reference points such as the origin in order to determine a synthetic reference point for the other class, so that a separator can be defined. If the transformation and the reference point is not chosen properly, the one-class SVM is unlikely to provide robust results in terms of identify-

ing the outlier. One issue with the one-class SVM is that the anomalous points (of interest) and the training data now need to lie on *opposite* sides of the separator. This is a more difficult case than one in which the anomalous points (of interest) and the training data need to lie on the same side of the separator (as was discussed in a previous section on positive-only SVMs). The key difference here is that the examples of interest are not available on the *interesting* side of the separator, which is poorly modeled.

It has been pointed out that the use of the origin as a prior for the anomalous class [91] can lead to incorrect results, since the precise nature of the anomaly is unknown a-priori. Therefore, the work in [91] attempts to determine a linear or non-linear decision surface which wrap around the surfaces of the normal class. Points which lie outside this decision surface are anomalies. It is important to note that this model essentially uses an indirect approach such as SVM to model the dense regions in the data. Virtually all unsupervised outlier detection methods attempt to model the normal behavior of the data, and can be used for novel class detection, especially when the only class in the training data is the normal class. *Therefore the distinction between normal-class only variations of the novel class detection problem and the unsupervised version of the problem are limited and artificial, especially when other labeled anomalous classes do not form a part of the training data.* Numerous analogues of unsupervised methods have also been developed for novelty detection, such as extreme value methods [383], direct density ratio estimation [214], and kernel-based PCA methods [220]. This is not surprising, given that the two problems are different only at a rather superficial level. In spite of this, the semi-supervised version of the (normal-class only) problem seems to have a distinct literature of its own. This is somewhat unnecessary, since any of the unsupervised algorithms can be applied to this case. The main difference is that the training and test data are distinguished from one another, and the outlier score is computed for a test instance with respect to the training data. Novelty detection can be better distinguished from the unsupervised case in temporal scenarios, where novelties are defined *continuously* based on the past behavior of the data. This will be discussed in more detail in Chapter 8 on temporal outlier detection, though a brief introduction is provided in the following subsections.

4.2   Combining Novel Class Detection with Rare Class Detection

A more challenging scenario arises, when labeled rare classes are present in the training data, but novel classes may also need to be detected. Such scenarios can arise quite often in many applications such as intrusion detection, where partial knowledge is available about *some* of the anomalies, but others may need to be modeled in an unsupervised way. Furthermore, it is important to distinguish different kinds of anomalies from one another, whether they are found in a supervised or unsupervised way. The labeled rare classes already provides important information about *some* of the outliers in the data. This can be used to determine different kinds of outliers in the underlying data, and distinguish them from one another. This is important in applications, where it is not only desirable to determine outliers, but also obtain an understanding of the kind of outlier which is discovered. The main challenge in these methods is to seamlessly combine unsupervised outlier detection methods with fully supervised rare class detection methods. For a given test data point two decisions need to be made, in the following order:

1. Is the test point a natural fit for a model of the training data? This model also includes the currently occurring rare classes. A variety of unsupervised models such as clustering can be used for thus purpose. If not, it is immediately flagged as an outlier, or a novelty.

2. If the test point is a fit for the training data, then a classifier model is used to determine whether it belongs to one of the rare classes. Any cost-sensitive model (or an ensemble of them) can be used for this purpose.

Thus, this model requires a combination of unsupervised and supervised methods in order to determine the outliers in the data. This situation arises more commonly in online and streaming scenarios, which will be discussed in the next section.

4.3   Online Novelty Detection

The most common scenario for novel class detection occurs in the context of *online* scenarios in concept drifting data streams. In fact, novel class detection usually has an implicit assumption of *temporal* data, since classes can be defined as novel only in terms of what has already been seen in the *past*. In many of the batch-algorithms discussed above, this temporal aspect is not fully explored, since a single snapshot of

training data is assumed. Many applications such as intrusion detection are naturally focussed on a streaming scenario. In such cases, novel classes may appear at any point in the data stream, and it may be desirable to distinguish different kinds of novel classes from one another [328, 329, 36]. Furthermore, when new classes are discovered, these kinds of anomalies may recur over time, albeit quite rarely. In such cases, the effectiveness of the model can be improved by keeping a memory of the *rarely recurring classes*. This case is particularly challenging because aside from the temporal aspects of modeling, it is desirable to perform the training and testing in an online manner, in which only one pass is allowed over the incoming data stream. This scenario is a true amalgamation of supervised and unsupervised methods for anomaly detection, and is discussed in detail in section 4.3 of Chapter 8.

In the streaming scenario containing only unlabeled data, unsupervised clustering methods [25, 26] can be used in order to identify significant novelties in the stream. In these methods, *novelties occur as emerging clusters in the data, which eventually become a part of the normal clustering structure of the data*. Both the methods in [25, 26] have statistical tests to identify, when a newly incoming instance in the stream should be considered a novelty. Thus, the output of these methods provides an understanding of the natural complementary relationship between the clusters (normal unsupervised models) and novelties (temporal abnormalities) in the underlying data. This issue will be discussed in some more detail in Chapter 8 on temporal outlier detection.

5.   Human Supervision

A natural form of supervision in outlier detection is one in which a human expert may intervene in the outlier detection process in order to further improve the effectiveness of the underlying algorithms. One of the major challenges in outlier detection is that the anomalies found by an algorithm which is either purely unsupervised or only partially supervised may not be very useful. This is because unsupervised algorithms (or even supervised methods with a small amount of training data) may not be able to effectively distinguish between *useless noise* and *useful outliers*. In such cases, it may be valuable to add human supervision to outlier analysis in order to detect more meaningful outliers. The incorporation of human supervision can augment the limited knowledge of outlier analysis algorithms. Specifically, the augmentation may be done in several ways:

- An unsupervised or supervised outlier detection algorithm may present pre-filtered results to a user, and the user can provide
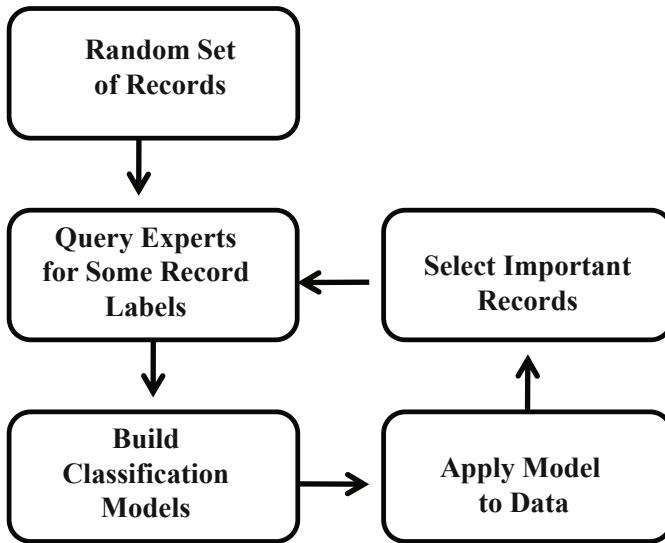
Figure 6.2. The overall procedure for active learning

their feedback on this small number of pre-filtered examples. This process would not be possible to perform manually on the original data set, which may large, and in which the vast majority of examples are normal [360].

■ The user-provided examples can be combined with the results from an unsupervised algorithm to learn which outliers determined by the unsupervised algorithm are relevant. The combined results can then be used in order to train a traditional rare class detection model, as discussed earlier in this chapter. For example, an SVM approach was used in [512].

Each of the aforementioned methodologies are discussed in detail below.

5.1   Active Learning

An interesting procedure for active learning from unlabeled data is proposed in [360]. An iterative procedure is used in order to label some of the examples in each iteration. In each iteration, a number of interesting instances are identified, for which the addition of labels would be helpful for further classification. These are considered the "important" instances. The human expert provides labels for these examples. These are then used in order to classify the data set with the augmented labels.

The first iteration is special, in which a purely unsupervised approach is used for learning. These procedure is performed iteratively until the addition of further examples is no longer deemed helpful for further classification. The overall procedure is illustrated in Figure 6.2. It should be noted that this approach can also be used in scenarios in which a small number of positive examples are available to begin with.

A key question arises as to *which* examples should be presented to the user for the purposes of labeling. It is clear that examples which are very obviously positive or negative (based on current models) are not particularly useful to present to the user. Rather, it is the examples with the greatest *uncertainty* or *ambiguity*, which should be presented to the user in order to gain the greatest knowledge about the decision boundaries between the different classes. It is expected that the selected examples should lie on the decision boundaries, in order to maximize the learning of the contours separating different classes, with the use of least amount of expert supervision, which can be expensive in many scenarios.

A common approach to achieve this goal in active learning is the principle of *query by committee* [400]. . In these methods, an ensemble of classifiers is learned, and the greatest *disagreement* among them is used to select data points which lie on the decision boundary. A variety of such criteria based on ensemble learning are discussed in [331]. It is also possible to use the model characteristics directly in order to select such points. For example, two primary criteria which can be used for selection, are as follows [360]:

- *Low Likelihood:* These are data points which have low fit to the model describing the data. For example, if an EM algorithm is used for modeling, then these are points which have low fit to the underlying model.

- *High Uncertainty:* These are points which have the greatest uncertainty in terms of the component of the model to which they belong. In other words, in an EM model, such a data point would show relatively even soft probabilities for different components of the mixture.

All data points are ranked on the basis of the two aforementioned criteria. The lists are merged by alternating between them, and adding the next point in the list, which has not already been added to the merged list. Details of other relevant methods such as *interleaving* are discussed in [360].

5.2   Outlier by Example

The outlier by example method follows the same principle of learning from (user-provided) positive and unlabeled examples, which was discussed earlier in this chapter, except for the difference that an unsupervised approach is utilize to perform feature transformations on the examples, and augment the user provided examples by comparing the deviations of the objects with those of the user-provided examples. The algorithm proceeds in the following steps:

- *Feature Extraction:* In this step, all the objects are transformed to their MDEF-based representations as discussed in the section on the LOCI method in Chapter 4. This is done by using the LOCI method discussed in Chapter 4, except that different *sampling neighborhoods* are used in order to create a vector of deviations for different sampling neighborhoods. Thus, this approach transforms the objects into a vector representation of MDEF values.

- *Example Augmentation:* A major challenge with all supervised learning methods is the paucity of training examples for effective training. Therefore, the user-provided examples are augmented in order to increase the number of positive examples. Two kinds of outliers are added. The first kind are outliers for which any component of the MDEF vector is greater than a user-specified threshold. These are referred to as *outstanding outliers.* The second kind of examples are *artificially generated* from the user specified outliers, by creating MDEF values which lie between their current maximum MDEF value and the threshold K. Depending upon the number of outliers which need to be generated, equally spaced intervals between the MDEF value and the threshold are generated. Note that the representation of the data is still in the form of MDEF vectors, and the artificially generated data is also represented in this form.

- *Final Classification:* The augmented training data is used to learn an SVM classifier, which distinguishes the unlabeled examples from the positive examples.

An interesting observation about the technique above is that the additional labeling and augmentation is done with the use of *automated* techniques. This is different from the method of [360] in which labeling is done by human experts. In both methods, human experts and automated methods are involved, but in different parts of the process.

6.   Conclusions and Summary

This chapter discusses the problem of supervised outlier analysis. In many real scenarios, training data is available, which can be used in order to greatly enhance the effectiveness of the outlier detection process. Many of the standard classification algorithms in the literature can be adapted to this problem, especially when full supervision is available. The major challenge of using the standard classification algorithms is that they may not work very well in scenarios where the distribution of classes is imbalanced. In order to address this issue, sampling and re-weighting can be used quite effectively.

The partially supervised variations of the problem are diverse. Some of these methods do not provide any labels on the normal class. This corresponds to the fact that the normal class may be contaminated with an unknown number of outlier examples. Furthermore, in some cases, the distribution of the normal class may be very different in the training and test data. One-class methods can sometimes be effective in addressing such issues.

Another form of partial supervision is the identification of novel classes in the training data. Novel classes correspond to scenarios in which the labels for some of the classes are completely missing from the training data. In such cases, a combination of unsupervised and supervised methods need to be used for the detection process. In cases where examples of a single normal class are available, the scenario becomes almost equivalent to the unsupervised version of the problem.

Supervised methods are closely related to active learning in which human experts may intervene in order to add more knowledge to the outlier detection process. Such combinations of automated filtering with human interaction can provide insightful results. The use of human intervention sometimes provides the more insightful results, because the human is involved in the entire process of label acquisition and final outlier detection.

7.   Bibliographic Survey

Supervision can be incorporated in a variety of ways, starting from partial supervision to complete supervision. In the case of complete supervision, the main challenges arise in the context of class imbalance and cost-sensitive learning [102, 105, 151]. The issue of evaluation is critical in cost-sensitive learning because of the inability to model the effectiveness with measures such as the absolute accuracy. Methods for interpreting ROC curves and classification accuracy in the presence of costs and class imbalance are discussed in [144, 159, 249, 376, 377]. The

impact of class imbalance is relevant even for feature selection [335, 511], because it is more desirable to select features which are more indicative of the rare class.

A variety of general methods have been proposed for cost-sensitive learning such as *MetaCost* [145], weighting [496], and sampling [106, 102, 143, 278, 496]. Weighting methods are generally quite effective, but may sometimes be unnecessarily inefficient, when most of the training data corresponds to the background distribution. In this context, sampling methods can significantly improve the efficiency. Numerous cost-sensitive variations of different classifiers have been proposed along the lines of weighting, and include the Bayes classifier [496], nearest neighbor classifier [506], decision trees [450, 463], rule-based classifiers [245, 247] and SVM classifiers [443, 470].

Ensemble methods for improving the robustness of sampling are proposed in [106, 312]. Since the under-sampling process reduces the number of negative examples, it is natural to use an ensemble of classifiers which combine the results of classifiers trained on different samples. This provides more robust results, and ameliorates the instability which arises from under-sampling. The major problem in over-sampling of the minority class is the over-fitting obtained by re-sampling duplicate instances. Therefore, a method known as *SMOTE* creates synthetic data instances in the neighborhood of the rare instances [103].

The earliest work on boosting rare classes was proposed in [252]. This technique is designed for imbalanced data sets, and the intuition is to boost the positive training instances (rare classes) faster than the negatives. Thus, it increases the weight of false negatives more the false positives. However, it is not cost-sensitive, and it also decreases the weight of true positives more than true negatives, which is not desirable. The *AdaCost* algorithm proposed in this chapter was proposed in [158]. Boosting techniques can also be combined with sampling methods, as in the case of the *SMOTEBoost* algorithm [104]. An evaluation of boosting algorithms for rare class detection is provided in [246]. Two new algorithms for boosting are also proposed in the same paper. The effect of the base learner on the final results of the boosting algorithm are investigated in [248]. It has been shown that the final result from the boosted algorithm is highly dependent on the quality of the base learner.

A particular case which is commonly encountered is one in which the instances of the positive class are specified, whereas the other class is unlabeled [152, 301, 293, 306, 307, 493, 507]. Since the unlabeled class is pre-dominantly a negative class with contaminants, it is essentially equivalent to a fully supervised problem, with some loss in accu-

racy which can be quantified [152]. In some cases, when the collection mechanisms for the negative class are not reflective of what would be encountered in test instances, the use of such instances may harm the performance of the classifier. In such cases, it may be desirable to discard the negative class entirely and treat the problem as a one-class problem [301]. However, as long as the training and test distributions are not too different, it is generally desirable to also use the instances from the negative class.

The one-class version of the problem is an extreme variation in which only positive instances of the class are used for training purpose. SVM methods are particularly popular for one-class classification [250, 323, 382, 396, 445]. Methods for one-class SVM methods for scene classification are proposed in [480]. It has been shown that the SVM method is particularly sensitive to the data set used [382].

An important class of semi-supervised algorithms is known as *novelty detection*, in which no training data is available about some of the anomalous classes. This is common in many scenarios such as intrusion detection, in which the patterns in the data may change over time, and may therefore lead to novel anomalies (or intrusions). These problems are combination of the supervised and unsupervised case, and numerous methods have been designed for the streaming scenario [328, 329, 36]. The special case, where only the normal class is available is not very different from the unsupervised scenario, other than the fact that it may have an underlying temporal component. Numerous methods have been designed for this case such as single-class SVMs [397, 91], minimax probability machines [282], kernel-based PCA methods [383], direct density ratio estimation [214], and extreme value analysis [220]. Single class novelty detection has also been studied extensively in the context of the first story detection in text streams [515], and will be discussed in detail in Chapter 7. The methods for the text streaming scenario are most highly unsupervised, and use standard clustering or nearest neighbor models. In fact, a variety of stream clustering methods [25, 26] discover newly forming clusters (or emerging novelties) as part of their output of the overall clustering process. A detailed survey of novelty detection methods may be found in [325, 326].

Human supervision is a natural goal in anomaly detection, since most of the anomalous instances are not interesting, and it is only by incorporating user feedback that the interesting examples can be separated from noisy anomalies. Methods for augmenting user-specified examples with automated methods are discussed in [512, 513]. These methods also add artificially generated examples to the training data, in order to increase the number of positive examples for the learning process. Other

methods are designed for *selectively* presenting examples to a user, so that only the relevant ones are labeled [360]. A nearest-neighbor method for active learning is proposed in [207]. The effectiveness of active learning methods for selecting good examples to present to the user is critical in ensuring minimal human effort. Such points should lie on the decision boundaries separating two classes [121]. Methods which use query by committee to select such points with ensembles are discussed in [331, 400]. A selective sampling method which uses active learning in the context of outlier detection is proposed in [1]. A method has also been proposed in [309] as to how unsupervised outlier detection algorithms can be leveraged in conjunction with limited human effort in order to create a labeled training data set.

8. Exercises

1. Download the *Arrythmia* data set from the *UCI Machine Learning Repository*.

   - Implement a 20-nearest neighbor classifier which classifies the majority class as the primary label. Use a $3 : 1$ ratio of costs between the normal class, and any other minority cost. Determine the overall accuracy and the cost-weighted accuracy.

   - Implement the same algorithm as above, except that each data point is given a weight, which is proportional to its cost. Determine the overall accuracy and the cost-weighted accuracy.

2. Repeat the exercise above for the quantitative attributes of the *KDD CUP 1999 Network Intrusion* data set of the *UCI Machine Learning Repository*.

3. Repeat each of the exercises above with the use of the *MetaCost* classifier, in which 100 different bagged executions are utilized in order to estimate the probability of relabeling. An unweighted 10-nearest neighbor classifier is used as the base learner. For each bagged execution, use a 50% sample of the data set. Determine the overall accuracy and the cost-weighted accuracy.

4. Repeat Exercises 1 and 2 by sampling one-thirds the examples from the normal class, and including all examples from the other classes. An unweighted 20-nearest neighbor classifier is used as the base learner. Determine the overall accuracy and the cost-weighted accuracy.

5. Repeat Exercise 4, by using an ensemble of five classifiers, and using the majority vote.

6. Repeat Exercises 1 and 2 with the use of cost-sensitive boosting. An unweighted 10-nearest neighbor classifier is used as the base learner.

Chapter 7

OUTLIER DETECTION IN CATEGORICAL, TEXT AND MIXED ATTRIBUTE DATA

"We become not a melting pot, but a mosaic. Different people, different beliefs, different yearnings, different hopes, different dreams."–Jimmy Carter

1. Introduction

A significant number of attributes in real data sets are not numerical, but have categorical values. For example, while demographic data may contain quantitative attributes such as the age, many other attributes such as sex and zip code are categorical. Data collected from surveys may often contain responses to multiple-choice questions, which are categorical. Similarly, many kinds of data such as the names of people and entities, IP-addresses and URLs are inherently discrete in nature. In many cases, categorical and numerical data are found in the same data set, as different attributes. This is referred to as *mixed-attribute data.* Mixed data is quite challenging to address because of the difficulties in appropriately weighting the importance of the different attributes.

Categorical data is inherently unordered. Unlike numerical data, it is often hard to assign similarity between different values on the same attribute. This creates numerous challenges in terms of generalizing algorithms for numerical data to the categorical domain. These challenges are as follows:

- Extreme value analysis and statistical algorithms are dependent on the use of statistical averages of different record values. In the cases of categorical data, statistics can be meaningfully computed

only on the frequencies of individual attribute values. It is no longer meaningful to evaluate averages over different values of an attribute.

- Linear models such as PCA are highly dependent on continuity in attribute values. Since such continuity does not exist in categorical data, a direct application of PCA is not possible, unless modifications are made to the basic approach.

- All proximity-based algorithms are dependent on some notion of similarity. However, notions of similarity such as the euclidian distance are no longer applicable to categorical data. Therefore, similarity functions need to be re-defined for categorical data in order to enable the use of proximity-based algorithms. Even in those cases, most of the index structures and efficiency-driven pruning techniques for numerical data cannot be easily generalized to categorical data.

- Many density-based models such as kernel-based methods and volume-based methods cannot be easily utilized for categorical data, because these methods are implicitly dependent on the notion of distances. Furthermore, density-based techniques also need to be adapted for categorical data.

Among the different classes of methods, proximity- and density-based techniques are the most promising, as along as suitable distance-functions and density-profiling methods can be developed for categorical data. However, all classes of algorithms can be adapted for categorical data with suitable modifications. Because of the importance of similarity computations in outlier detection, this chapter will also review the similarity measures for categorical data. In particular, recent work [75] has studied many of these measures specifically in the context of the outlier detection problem.

An important observation about categorical data is that it can be transformed to binary data, by treating each value of the categorical attribute as a binary attribute. This allows the use of many algorithms which are designed for domains such as text data. Furthermore, since binary data is a (rudimentary) special case of numerical data, the existing algorithms for numerical data can also be applied to this case. However in practice, such an approach is rather expensive, when the number of possible *values* of a categorical attribute is large. This is because a separate binary field needs to be dedicated to each *distinct* value of the categorical attribute. Such cases arise quite often in many real scenarios. For example, for a field such as the zip-code, thousands of

values may correspond to the attribute. Furthermore, since these values are typically disjoint of one another, it would seem rather wasteful to use the binary representation for processing purposes.

For ease in further discussion, the notations for categorical data are introduced here. It is assumed that the d-dimensional data set contains N records denoted by $\mathcal{D}$. When the data set is purely categorical, is assumed that the ith attribute contains n_i possible distinct values. On the other hand, for a mixed attribute data set, the first d_c attributes are assumed to be categorical, and the remaining are assumed to numerical.

This chapter is organized as follows. The various sections will discuss how the different classes of algorithms can be generalized to categorical data. Section 2 discusses the generalization of probabilistic models to categorical data. Section 3 discusses the extension of linear models to categorical data. The extension of proximity-based models to categorical data is studied in section 4. A special case of categorical data is binary data, which often corresponds to many kinds of transaction databases. Outlier detection methods for such data are discussed in section 5. The case of text data is studied in section 6. Section 7 discusses the conclusions and summary.

2.   Extending Probabilistic Models to Categorical Data

Outlier detection methods can easily be generalized to the case of categorical data [478] with the use of an appropriate generative models for categorical values. Generative models have the advantage that the specific complexity of the data domain is captured with a pre-defined probability distribution describing the behavior of each mixture component. Once this distribution has been defined, the other steps of the algorithm are analogous to the numerical case discussed in Chapter 2, with many steps being virtually identical. The main differences arise in terms of the methodology for parameter estimation, which is clearly domain data distribution-specific. Even those aspects are analogously defined to the numerical case. Furthermore, such an approach can be easily extended to mixed-attribute data by defining appropriate probability distributions for the categorical and numerical attributes, and combining them in product form.

As before, it is assumed that the mixture contains k components denoted by $\mathcal{G}_1 \ldots \mathcal{G}_k$:

- Pick a data distribution with probability α_i, where $i \in \{1 \ldots k\}$, in order to pick one of the k distributions. Let us assume that the rth one is picked.

■ Generate a data point from $\mathcal{G}_r$.

The values of α_i denote the prior probabilities, and may either be pre-specified to equal values of $1/k$ as a simplification, or may be learned from the data. The main difference from the numerical case is in the mathematical form of the generative model for the mth cluster (or mixture component) $\mathcal{G}_m$. In the bernoulli[1] model, it is assumed that the jth value of ith attribute is generated by cluster m with probability p_{ijm}. All model parameters are denoted by the notation Θ.

Consider a record $\overline{X}$ containing the attribute values $j_1 \ldots j_d$, where the rth attribute takes on the value of j_r. Then, the value of the generation probability $g^{m,\Theta}(\overline{X})$ from cluster m, is given by the following expression:

$$g^{m,\Theta}(\overline{X}) = \prod_{r=1}^{d} p_{rj_rk} \tag{7.1}$$

Note that the value of $g^{m,\Theta}(\cdot)$ is a discrete probability, unlike the continuous density function $f^{m,\Theta}(\cdot)$ discussed in Chapter 2. Nevertheless, this is the analogous quantification for the cluster-specific generation probability. Correspondingly, the assignment probability $P(X \in \mathcal{G}_m)$ for the m-th cluster may be estimated as follows:

$$P(\overline{X} \in \mathcal{G}_m | \Theta) = \frac{\alpha_m \cdot g^{m,\Theta}(\overline{X})}{\sum_{r=1}^{k} \alpha_r \cdot g^{r,\Theta}(\overline{X})}$$

This defines the E-step for the categorical scenario. Note that this step provides a soft assignment probability of the data points to the different clusters.

Once the soft-assignment probability is determined, it is easy to analyze the individual components of the mixture in order to estimate the probability p_{ijm}. In this case, the maximization of the log-likelihood fit takes on a particularly simple form. While estimating the parameters for cluster m, the *weight* of a record is assumed to be equal to its assignment probability $P(\overline{X} \in \mathcal{G}_m | \Theta)$ to cluster m. The value α_m is estimated in exactly the same way as discussed in Chapter 2. Specifically, α_m is the weighted fraction of records assigned to cluster m on the basis of the soft probabilities computed in the E-step. For smaller data sets, the smoothing methods discussed in Chapter 2 may also be used. For each cluster m, the *weighted* number w_{ijm} of records for which attribute i

---

[1]Strictly speaking, Bernoulli models are designed for the case of binary data containing only two possible values for each attribute. This is a slightly generalized version of the model.

takes on the value j in cluster m is estimated. Then, the value of the probability p_{ijm} may be estimated as follows:

$$p_{ijm} = \frac{w_{ijm}}{\sum_{\overline{X} \in \mathcal{D}} P(\overline{X} \in \mathcal{G}_m | \Theta)}$$

In practice, sufficient data may often not be able to estimate the parameters robustly, especially when some of the attribute values may not appear in a cluster (or $w_{ijm} \approx 0$). This can lead to poor parameter estimations. A technique known as *Laplacian smoothing* is commonly used in order to address such ill-conditioned probabilities. This is achieved by adding a small positive value β to the numerator and denominator of the aforementioned estimation. The smoothed estimation is as follows:

$$p_{ijm} = \frac{\beta + w_{ijm}}{\beta + \sum_{\overline{X} \in \mathcal{D}} P(\overline{X} \in \mathcal{G}_m | \Theta)}$$

This completes the description of the M-step. As in the case of numerical data, the E-step and M-steps are iterated to convergence. Outliers may be declared as data points which either have low assignment probability to their best matching cluster, or have low absolute fit to the generative model. If desired, extreme value analysis can be used on the fit probabilities in order to determine the relevant outliers.

2.1 Modeling Mixed Data

Mixed data is particularly easy to address in the case of probabilistic models. One challenge with mixed data with most models are the *normalization* issues which arise in ensuring that all attributes are provided equal importance by an unsupervised algorithm in the absence of prior knowledge about the relative importance of attributes. In probabilistic models, all normalization issues which might arise in the context of mixed data sets are already addressed because each attribute is quantified in a homogeneous way corresponding to the probability.

The EM-algorithm is modified by defining the generative probabilities of each component as a composite function of the different kinds of attributes. For example, the continuous attributes are modeled with a *density function* $f^{r,\Theta}(\overline{X})$, whereas the categorical attributes are modeled with a discrete probability function $g^{r,\Theta}(\overline{X})$. In a scenario containing both categorical and mixed attributes, it is possible to define the following joint density function $h^{r,\Theta}(\overline{X})$, which is a product of these values:

$$h^{r,\Theta}(\overline{X}) = f^{r,\Theta}(\overline{X}) \cdot g^{r,\Theta}(\overline{X}) \tag{7.2}$$

The E-step can then be performed with the use of this joint density function $h^{r,\Theta}(\overline{X})$. The M-step is performed separately on the categorical

and numerical attributes in exactly the same way as discussed in Chapter 2 (for numerical attributes) and the discussion above (for categorical attributes).

A specific method for outlier analysis with mixed attribute data sets was proposed by [478]. This is an online approach in which temporal discounting is used in order to provide more importance to recent records. This approach has some differences with the more generic description provided above. For example, the technique in [478] uses the Hellinger scores in order to compute the outlier scores rather than the absolute fit probabilities. Here a simplified and more general description of probabilistic models is provided, though the interested reader may refer to [478] for a description optimized to the online scenario.

3.   Extending Linear Models to Categorical and Mixed Data

Linear models can be extended to categorical data sets by using the conversion from the categorical data set to the binary scenario. Since the ith attribute contains n_i possible values, this leads to a data set with dimensionality $\sum_{i=1}^{d} n_i$. Thus, when the value of n_i is large for many attributes i, this may lead to a significant expansion of the dimensionality of the data.

Another challenge that arises is that the different attributes may implicitly be provided different importance by such a transformation, when the values of n_i vary widely across different attributes. For example, a column in which 50% of the binary attributes take on the value of 1, is very different from a column in which only 1% of the attributes take on the value of 1. The data set can be normalized by dividing each column by its standard deviation. Consider the jth value of the transformed binary data, for which a fraction f_{ij} of the entries take in the value of 1. In such a case, the standard deviation is given by $\sqrt{f_{ij} \cdot (1 - f_{ij})}$. Therefore, one possibility would be to divide that column of the binary data by $\sqrt{f_{ij} \cdot (1 - f_{ij})}$. As in the case of numerical data, this corresponds to a normalization in which the variance of each *derived* binary dimension is set to the same value of 1. Unfortunately, such a normalization will favor attributes with large values of n_i, since their cumulative variance in the principal component matrix will be n_i. In order to account for this, all columns for the ith attribute are divided by $\sqrt{n_i \cdot f_{ij} \cdot (1 - f_{ij})}$. This ensures that the sum of the variances for all the columns corresponding to a particular attribute is equal to 1. The standard PCA models can be directly applied to this representation in order to determine the outlying points in the data.

Such an approach can also be used for mixed attribute data sets, by adding normalized numerical attributes to this representation. The normalization for the numerical columns is relatively straightforward. The columns are normalized, so that the variance of each column is one. This ensures that such methods can be used in a reasonable way over a diverse data set with mixed attributes. Principal component analysis methods are particularly effective for sparse binary data sets, since they can represent the data in a very small number of components.

Once the principal components have been discovered, a similar methodology to that discussed in Chapter 3 can be used for outlier modeling. The deviations along each of the principal components are determined, and the sums of the squares of these values are modeled as a χ^2 distribution with d degrees of freedom. The extreme values among the different point-specific deviations are reported as the outliers.

4.   Extending Proximity Models to Categorical Data

The design of effective similarity measures for categorical data is critical for the effective implementation of many proximity-based algorithms such as clustering and outlier detection. While the design of similarity measures for categorical data is a vast research area in its own right, this chapter will discuss some of the more common measures for categorical data. For the case of categorical data, it is generally more natural to study similarities rather than distances because of many measures which are inherently based on matching [75].

Consider two records $\overline{X} = (x_1 \ldots x_d)$ and $\overline{Y} = (y_1 \ldots y_d)$. Then, the similarity between the records $\overline{X}$ and $\overline{Y}$ is the sum of the similarities on the individual attribute values. In other words, if $S(x_i, y_i)$ be the similarity between the attributes values x_i and y_i, then the overall similarity is defined as follows:

$$Sim(\overline{X}, \overline{Y}) = \sum_{i=1}^{d} S(x_i, y_i)$$

This similarity measure can be defined differently, depending upon how $S(x_i, y_i)$ is instantiated.

The simplest possible measure is to fix $S(x_i, y_i)$ to 1 when $x_i = y_i$ and 0 otherwise. This is also referred to as the *overlap* measure. This measure is used most commonly for the case of categorical data, because of simplicity in evaluation. The major disadvantage of this measure is that it does not account for the relative frequencies among the different

attributes. Furthermore, inter-attribute correlations are often present in the data, which should be used in the modeling process.

There are two primary classes of methods for measuring similarity in categorical data:

- The aggregate statistical properties of the data can be used in order to enable better similarity measures. This typically corresponds to the statistical frequency of attributes. Measures of these types are discussed in detail in [75].

- The statistical neighborhoods of data points are used to compute similarity. This approach implicitly models the inter-attribute correlations in similarity computations. Of course, since the definition of neighborhood is itself based on similarity, this definition is essentially circular. Therefore, simpler methods such as the *overlap* measure may be required in order to perform the initial computation. Of course, a better definition of the first kind of measure can enable better overall initialization of the similarity computation process. An example of such a measure is provided in [126].

The second approach for similarity computation is clearly much richer. However, it is also computationally more intensive. This section will discuss both kinds of methods.

4.1   Aggregate Statistical Similarity

In the context of categorical data, the *aggregate statistical properties* of the data set should be used in computing similarity. For example, consider a case where an attribute value x_1 takes on the value of "*Normal*" 99% of the time, and the value of "*Abnormal*" 1% of the time. In such a case, the similarity between two values which are common should be given less weight than two values which are less common. This forms the basis of many common normalization techniques such as the *Inverse Document Frequency (IDF)* in the information retrieval domain. Another weakness of the overlap measure is that it does not properly compute similarity for the case where x_i and y_i are not the same. Even when x_i and y_i are not the same, the value of $S(x_i, y_i)$ could be different depending upon the relative frequencies of x_i and y_i.

The most common measure which weights the frequencies appropriately is the *Inverse Occurrence Frequency*, where the similarity between a pair of attributes is inverse weighted by a function of the frequencies of that pair of attribute values. In practice, a function such as the square-root or the logarithm of the frequency is used in order to ensure more stable behavior. Thus, when $x_i = y_i$, the similarity is equal to the

inverse weighted frequency function, and zero otherwise. This measure is used very commonly in the information retrieval domain, in order to compute similarities between text attributes [391].

Another common measure which is used for categorical data is the *Goodall* measure which uses the relative presence of the different attribute values. Correspondingly, a higher similarity value is assigned to a match when the value is infrequent. Furthermore, multivariate dependencies between different attribute values are utilized in the final computation. The original method proposed in [187] is slow. Therefore, the work in [75] uses a number of heuristic approximations which attempt to capture the spirit of this measure in different ways, but can also be computed efficiently. Let $p_k(x)$ be the fraction of records in which the kth attribute takes on the value of x in the data set. Then, one possible variant of the Goodall measure would be to use the $1 - p_k(x_i)^2$ as the similarity on the kth attribute, when $x_i = y_i$, and 0 otherwise. Other measures which lie in this general category are discussed in [187, 175, 87].

An important observation is that many of the above measures are somewhat similar in broad principle, in terms of weighting different attribute values differently. The major differences lie in how this broad principle is applied. For example, the work in [87] uses information theoretic measures for computation of similarity. However, the major difference from the aforementioned definitions really lies in the choice of the exact function which is used for weighting the attribute frequencies. Other measures also distinguish between mismatches on a given attribute value. In other words, $S(x_i, y_i)$ is not always set to 0 (or the same value), when x_i and y_i are different. This is because infrequent attribute values are statistically expected to be more different than frequent attribute values. A number of such measures are proposed in [42, 303, 411], and summarized very well in [75].

The work in [75] performed an extensive evaluation of a variety of categorical similarity measures, which are based on aggregation measures. This work is especially relevant because the evaluations were performed in the context of the outlier detection problem. The broader conclusion of this work was that the use of statistical measures definitely improves the quality of the similarity. However, no single measure was dominant over all the other measures.

4.2   Contextual Similarity

Contextual similarity measures provide a fundamentally different point of view, in which the neighborhood of a data point is used to compute

similarity. For example, let us consider a binary categorical database in which each attribute corresponds to whether or not an attribute bought *Coke*, *Pepsi*, *Mustard* etc. In such cases, it is evident that two customers who buy *Coke* and *Pepsi* respectively are more likely to show similar buying patterns on the other attributes, than a customer who buys *Mustard*. Therefore, it is possible to use this relationship in order to define contextual similarities between values of *different* attributes. Note that in the case of aggregate similarity, the value of $S(x_i, y_j)$ was always assumed to be 0, when $i \neq j$. However, in the case of the contextual similarity measure [126], this is no longer the case.

The set of rows which correspond to *Coke* represent a *sub-relation* of the data. When two attribute values (across different attributes) have similar sub-relations in terms of the underlying patterns, the corresponding attribute values are also similar. Therefore, the work in [126] uses an *Iterative Contextual Distance* algorithm, in which the distances are determined by repeating the following steps in circular fashion, after an initialization, which is based on simpler aggregate measures of similarity:

- The distance between attributes is used in order to construct a real valued representation of the rows in the data. This real valued representation encodes a lot of information about the inter-attribute correlations. Therefore distances between the rows in this real-valued representation automatically encode inter-attribute similarity. While a number of simpler ways could be used to achieve this, the work in [126] uses a kernel mapping approach in order to define the real-valued representations of the rows. The details of this approach may be found in [126]. A simpler intuitive description will be provided here. In order to encode the distances between different attribute values, this approach defines real values for each entry which corresponds to the similarity of a particular attribute to the different attributes which are present in the record.

- The distances between attribute values is defined on the basis of distances between their sub-relations. First, the sub-relations are isolated for each attribute value (eg. *Coke* customers, and *Pepsi* customers). The real-valued centroid of each of the kernel mapping of the rows in the sub-relations is determined. The L_1 distance between the centroids is used in order to measure the distance between the corresponding attributes. Other more complex measures such as the probabilistic differences between the distributions of the row values could also be used in order to achieve this goal in a more sophisticated way.

These steps are repeated by the algorithm till convergence. While this algorithm was originally presented in the context of binary (market-basket) attributes, it can easily be generalized to any kind of categorical data.

4.3 Issues with Mixed Data

It is fairly straightforward to generalize the approach to the case of mixed data, as long as appropriate weights can be assigned to the categorical and numerical components. For example, let us consider two records $\overline{X} = (\overline{X_n}, \overline{X_c})$ and $\overline{Y} = (\overline{Y_n}, \overline{Y_c})$, where $\overline{X_n}$, $\overline{Y_n}$ are the subsets of numerical attributes and $\overline{X_c}$, $\overline{Y_c}$ ate the subsets of categorical attributes. Then, the overall similarity between $\overline{X}$ and $\overline{Y}$ is defined as follows:

$$Sim(\overline{X}, \overline{Y}) = \lambda \cdot NumSim(\overline{X_n}, \overline{Y_n}) + (1 - \lambda) \cdot CatSim(\overline{X_c}, \overline{Y_c}) \quad (7.3)$$

The parameter λ regulates the relative importance of the categorical and numerical attributes. The choice of λ can sometimes be a challenging task, especially since the similarities on the two domains may not be of the same scale. In the absence of prior or domain knowledge about the relative importance of attributes, a natural choice would be to use a value of λ, which is equal to the fraction of numerical attributes in the data. Furthermore, the proximity in numerical data is often computed with the use of distance functions rather than similarity functions. However, distance values can be converted to similarity values using a method suggested in [75]. For a distance value of $dist$, the corresponding similarity value is denoted by $1/(1 + dist)$.

Further normalization is required in order to meaningfully compare the similarity value components on the numerical and categorical attributes, which may be in completely different scales. One way of achieving this goal would be to determine the standard deviations in the similarity values over the two domains with the use of sample pairs of records. Each component of the similarity value (numerical or categorical) is divided by its standard deviation. Therefore, if σ_c and σ_n be the standard deviations of the similarity values in the categorical and numerical components, then Equation 7.3 needs to be modified as follows:

$$Sim(\overline{X}, \overline{Y}) = \lambda \cdot NumSim(\overline{X_n}, \overline{Y_n})/\sigma_n + (1 - \lambda) \cdot CatSim(\overline{X_c}, \overline{Y_c})/\sigma_c$$

By performing this normalization, the value of λ becomes more meaningful, as a true *relative weight* between the two components. By default, this weight can be set to be proportional to the number of attributes in each component, unless specific domain knowledge is available about the relative importance of attributes.

4.4   Density-based Methods

Density-based methods are more natural to perform in the context of discrete data, because frequency profiles can be constructed on different combinations of attribute values. Such methods have been discussed in section 4.3 of Chapter 4 in the context of numerical attributes.

In this case of categorical data, such methods are very natural to use, since the additional step of discretization does not need to be performed in order to convert the numerical values to discrete values. In the case of mixed attribute data, the numerical attributes may be discretized, whereas the categorical attributes may be retained in their original form. All other steps are identical to the methodology discussed in section 4.3 of Chapter 4.

4.5   Clustering Methods

As in the case of numerical data, outliers are defined as data points which are not members of clusters, or are not in sufficient proximity of clusters. While numerous clustering methods exist for categorical data, such methods are often not applied to the categorical domain, because of the greater difficulty in defining similarity between individual records and cluster representatives (centroids). The only clustering method which is commonly used in the categorical domain for outlier analysis is the EM-method discussed earlier in this section. In that case, fit probabilities can be used to effectively represent outlier scores.

While it is possible to define outliers on the basis of non-membership to clusters, such methods are not optimized to finding true anomalies in the data, and may often discover noise. Therefore, methods need to be defined in order to compute the distances between cluster representatives and the individual data points. One such method was described in [26], and the method is also used in order to detect outliers. A discussion of common clustering methods for categorical data is also provided in bibliographic section of this chapter, though most of these methods are optimized towards finding clusters rather than outliers.

5.   Outlier Detection in Binary and Transaction Data

A significant amount of categorical data is binary in nature. For example, transaction data, which contains customer buying patterns is almost always binary. Furthermore, all forms of categorical data and numerical data can always be transformed to binary data by various forms of discretization. Since binary data is a special case of all forms of cat-

egorical and numerical data, most of the methods discussed in this and other chapters can be applied to such data. Nevertheless, transaction data, which exists in a binary form in its natural state is a bit different from other forms of binary data which are obtained by artificial conversion. The former kind of data is typically very high dimensional, but is sparse, and may often contain highly correlated patterns. Therefore, a number of methods have also been designed which are specifically optimized to such forms of data. Some of these methods can also be applied to categorical data, when the underlying patterns are very sparse.

5.1   Subspace Methods

Since transaction data is inherently high-dimensional, it is natural to utilize subspace methods [4] in order to identify the relevant outliers. The challenge in subspace methods is that it is no longer computationally practical or statistically feasible to define subspaces (or sets of items), which are sparse for outlier detection. For example, in a sparse transaction database containing hundreds of thousands of items, sparse itemsets are the norm rather than the rule. Therefore, a subspace exploration for sparse itemsets is likely to report the vast majority of patterns. The work in [208] addresses this challenge by working in terms of the relationship of transactions to dense subspaces, rather than sparse subspaces. In other words, this is a reverse approach of determining transactions, which are *not included* in most of the relevant dense subspace clusters of the data. In the context of transaction data, subspace clusters are essentially frequent patterns.

The idea in such methods is that frequent patterns are less likely to occur in outlier transactions, as compared to normal transactions. Therefore, a measure has been proposed in [209], which sums up the support of all frequent patterns occurring in a given transaction in order to provide the outlier score of that transaction. The total sum is normalized by dividing with the number of frequent patterns. However, this term can be omitted from the final score, since it is the same across all transactions.

Let $\mathcal{D}$ be a transaction database containing the patterns denoted by $T_1 \dots T_N$. Let $s(T_i, \mathcal{D})$ represent the support of transaction T_i in $\mathcal{D}$. Therefore, if $FPS(\mathcal{D}, s_m)$ represents the set of frequent patterns in the database $\mathcal{D}$ at minimum the support level s_m, then, the frequent pattern outlier factor $FPOF(T_i)$ of a transaction $T_i \in \mathcal{D}$ at minimum support s_m is defined as follows:

$$FPOF(T_i) = \frac{\sum_{X \in FPS(\mathcal{D}, s_m), X \subseteq T_i} s(T_i, \mathcal{D})}{|FPS(\mathcal{D}, s)|}$$

Intuitively, a transaction containing a large number of frequent patterns with high support will have high value of $FPOF(T_i)$. Such a transaction is unlikely to be an outlier, because it reflects the major patterns in the data.

As in other subspace methods, such an approach can also be used in order to describe, why a data point may not be considered an outlier. Intuitively, the frequent patterns with the largest support, which are also not included in the transaction T_i are considered *contradictory patterns* to T_i. Let S be a frequent pattern not contained in T_i. Therefore, $S - T_i$ is non-empty, and the *contradictiveness* of frequent pattern S to the transaction T_i is defined by $s(S, \mathcal{D}) * |S - T_i|$. Therefore, a transaction which does not have many items in common with a very frequent itemset is likely to be one of the explanatory patterns for the T_i being an outlier. The patterns with the top-k values of contradictiveness are reported as the corresponding explanatory patterns.

At an intuitive level, such an approach is analogous to non-membership of data points in clusters in order to define outliers, rather than directly trying to determine the deviation or sparsity level of the transactions. As was discussed in the chapter on clustering-based methods, such an approach may sometimes not be able to distinguish between noise and anomalies in the data. However, the approach in [209] indirectly uses the weight and number of clusters in the outlier score. Furthermore, it uses *multiple* patterns in order to provide an ensemble score. This is at least partially able to alleviate the noise effects. In the context of very sparse transactional data, in which direct exploration of rare subspaces is infeasible, such an approach would seem to be a reasonable adaptation of subspace methods.

Frequent pattern mining methods are closely related to information theoretic measures for anomaly detection. This is because frequent patterns can be viewed as a code-book in terms of which to represent the data in a compressed form. It has been shown in [407], how frequent patterns can be used in order to create a compressed representation of the data set. Therefore, a natural extension is to use the description length [410] of the compressed data in order to compute the outlier scores. This approach was further improved in [34].

5.2   Novelties in Temporal Transactions

Transaction data is often temporal in nature, where the individual transactions are associated with a time-stamp. A new transaction which does not reflect the aggregate behavior of the transactions which have been maintained so far can be flagged as a novelty. A method for novelty

detection in context of fast binary data streams has been proposed in [26]. In fact, this approach falls in a general class of proximity-based methods, which will be discussed for text data later in this chapter. Thus, this method is fairly general and can be applied to both text and categorical data.

The broad idea of the approach is to continuously maintain the clusters in the underlying temporal data stream. Data points which do not naturally fit into any of these clusters are regarded as novelties. Thus, for each incoming data point, its best similarity to the centroids of the current set of clusters is determined. If this similarity is statistically too low, then the data point is flagged as a novelty, and it is placed in a cluster of its own. Subsequently, it is possible that this data point may correspond to the beginning of a new trend or cluster in the data. Such situations are quite common in many novelty-detection applications, where a detected outlier eventually becomes a normal part of the data. However, in some cases, such data points may continue to remain an outlier over time. This issue will be discussed in more detail in Chapter 8 on outlier detection in temporal data.

6. Outlier Detection in Text Data

Many of the probabilistic, linear and proximity-based models have also been generalized to the case of text data. In the context of text data, outlier analysis may be used for either *noise removal* or for the detection of *interesting anomalies*, such as a *first story* in a text stream. The most common method used for noise removal and correction in text is Latent Semantic Indexing, and the common method utilized for determining unique segments of text includes a variety of probabilistic and proximity-based methods.

6.1 Latent Semantic Indexing

The goal of LSI is largely to *improve the underlying data representation*, so as to reduce the impact of noise and outliers. One of the noisy aspects of text data is that it often contains a significant level of *synonymy* and *polysemy*. In synonymy, multiple descriptions exist for the same concept. For example, a "*car*" may also be referred to as an "*automobile*" in a different document. In polysemy, the same word, may correspond to different concepts. For example, the word "*Jaguar*" may either refer to a car, or a kind of cat. Clearly, the presence of such words in text collections is a kind of noise, which can greatly reduce the quality of many applications such as similarity search. LSI is a method to improve the underlying data representation, so as to reduce the impact of

such noise. This broader principle has also been discussed in the section on noise correction with PCA in Chapter 3. While the method of LSI was originally proposed for indexing applications [133], its impact on noise correction was first observed in [355].

LSI is quite similar to PCA, except that the covariance matrix is not used for analysis. Specifically, let A be the $N \times d$ term-document matrix in which the (i, j)th entry is the normalized frequency for term j in document i. Then, $A^T \cdot A$ is a $d \times d$ matrix which is close (scaled) approximation of the covariance matrix, in which the means have not been subtracted out. In other words, the value of $A^T \cdot A$ would be the same as a scaled version (by factor n) of the covariance matrix, if the data is mean-centered. While text-representations are not mean-centered, the sparsity of text ensures that the use of $A^T \cdot A$ is quite a good approximation of the (scaled) covariances. As in the case of numerical data, the eigenvectors of $A^T \cdot A$ with the largest variance are determined in order to represent the text. In typical text collections, only about 300 to 400 eigenvectors are required for the representation. One excellent characteristic of LSI is that the truncation of the dimensions removes the noise effects of synonymy and polysemy, and the similarity computations are more closely affected by the semantic concepts in the data. This is particularly useful for a semantic application such as text clustering.

It should be further noted that documents which are very incoherent, spam, or otherwise incongruent with the rest of the data are more easily revealed in the new representation. Such documents can be more easily distinguished from the data, once the noise dimensions have been removed. If desired, LSI can also be used in combination with proximity-based techniques for more effective outlier detection. Such methods will be discussed in the next subsection.

6.2   First Story Detection

Much of the work on outlier detection in the text domain has been studied in the context of the problem of *first-story detection* in a stream of text documents, such as what is encountered in a news wire service. While some related methods for temporal data will be studied in Chapter 8, the case of text is studied in this chapter, because it is more closely related to the other material in this chapter. Furthermore, there are few methods for text outlier detection in non-temporal scenarios, and most of the methods for *first-story detection* can trivially be generalized to the non-temporal scenarios. In the context of temporal data, such data points are also referred to as *novelties.* Two broad classes of methods

exist for this problem, corresponding to proximity-based models and probabilistic models. While both of these methods have been proposed in the context of temporal data, they can trivially be generalized to the non-temporal scenario, since the temporal component is utilized only from the perspective of data subset selection in the modeling process. In the non-temporal scenario, these methods simply need to be applied to the entire data, rather than the subset of the data from the previous history.

6.2.1   Proximity-based Models.   Since proximity-based models require the use of a similarity function, a natural question arises as to which similarity function is most suitable for the case of text data. As in the case of categorical data, the word frequencies need to be normalized in terms of their relative frequency of presence in the document and over the entire collection. In general, a common representation used for text processing is the *vector-space based* TF-IDF representation [391]. In the TF-IDF representation, the term frequency for each word is normalized by the *inverse document frequency*, or IDF. The inverse document frequency normalization reduces the weight of terms which occur more frequently in the collection. This reduces the importance of common terms in the collection, ensuring that the matching of documents be more influenced by that of more discriminative words which have relatively low frequencies in the collection. In addition, a sub-linear transformation function is often applied to the term-frequencies in order to avoid the undesirable dominating effect of any single term that might be very frequent in a document. Once this normalization has been achieved, the standard cosine function is applied to the vector-space representations in order to measure similarity. The cosine is essentially equal to the dot product of two vectors, once they have been normalized to unit length. Thus, for two normalized document frequency vectors $\overline{X}$ and $\overline{Y}$, the cosine function $Cosine(\overline{X}, \overline{Y})$ can be defined as follows:

$$Cosine(\overline{X}, \overline{Y}) = \frac{\overline{X} \cdot \overline{Y}}{||\overline{X}|| \cdot ||\overline{Y}||} \qquad (7.4)$$

Here, $|| \cdot ||$ represents the L_2-norm. The earliest work on proximity-based models was done in the context of the topic detection and tracking (TDT) project [39, 515]. Most of the methods [37–39, 515] determine key novelties in the stream by using the following broad two step framework for each incoming document, with respect to a current summary which is maintained for the corpus:

- Determine the similarity of the incoming document to the current summary of the corpus. The summary of the corpus could

correspond to either a fine grained clustering, or a sample of the documents which have been received so far. Report any document which have low similarity with the current summary as anomalies. For example, the similarity to the closest cluster centroid, or to the nearest document in the sample could be used for this purpose.

- Update the summary of the incoming documents in the data stream, by incorporating the incoming document into an appropriate component of the summary.

Recently, a method has also been proposed for novelty detection in fast data streams [26]. This approach uses an exponential decaying model in order to determine clusters and novelties simultaneously from the data stream. Eventually such novelties are converted into clusters, if more data points arrive which match the content of the document. Such methods are also able to identify novelties which are eventually converted into broad trends in the data stream.

The method of [26] makes the assumption of fixed memory availability, because of which the number of clusters in the data remain fixed. As a result, a current cluster needs to be ejected from the buffer when a new cluster is inserted. Typically such ejected clusters are chosen as those (faded) trends, which have not been updated for a while. However, it is possible that at a future time, such a trend will re-appear. As a result, a similar cluster may need to be created at a future point in time. Such outliers are referred to as *infrequently recurring outliers*, which are distinct from novelties that were never seen before. In the presence of *fixed space* it is often not possible for any summarization method to distinguish between novelties and infrequently recurring outliers, because the latter may not be available in the stream summary. However, if the number of clusters are allowed to grow over time, or if the total space availability is very large, then it may be possible to distinguish between such events. Nevertheless, in the growing cluster scenario, such an approach is likely to slow down over time, and may no longer be relevant to the stream scenario.

6.2.2   Probabilistic Models.   Probabilistic models are a common technique for clustering text data. Since novelty detection is closely related to maintaining online summaries of text collections, it is natural that such a method may be used for novelty-detection in text data. A popular method for probabilistic document clustering is that of *topic modeling*. The idea of topic modeling is to create a *probabilistic generative model* for the text documents in the corpus. This approach is

essentially an application of the EM algorithm (see Chapter 2) to the text domain.

The main approach is to represent a corpus as a function of hidden random variables, the parameters of which are estimated using a particular document collection. The primary assumptions in any topic modeling approach (together with the corresponding random variables) are as follows:

- The N documents in the corpus are assumed to have a probability of belonging to one of k topics. Thus, a given document may have a probability of belonging to multiple topics, and this reflects the fact that the same document may contain a multitude of subjects. For a given document D_i, and a set of topics $T_1 \ldots T_k$, the probability that the document D_i belongs to the topic T_j is given by $P(T_j|D_i)$. The topics are essentially analogous to clusters, and the value of $P(T_j|D_i)$ provides a probability of cluster membership of the ith document to the jth cluster. In non-probabilistic clustering methods, the membership of documents to clusters is deterministic in nature, and therefore the clustering is typically a clean partitioning of the document collection. However, this often creates challenges, when there are overlaps in document subject matter across multiple clusters. The use of a *soft cluster membership in terms of probabilities* is an elegant solution to this dilemma. In this scenario, the determination of the membership of the documents to clusters is a secondary goal to that of finding the *latent topical clusters* in the underlying text collection. Therefore, this area of research is referred to as *topic modeling*, and while it is related to the clustering problem, it is often studied as a distinct area of research from clustering.

  The value of $P(T_j|D_i)$ is estimated using the topic modeling approach, and is one of the primary outputs of the algorithm. The value of k is one of the inputs to the algorithm and is analogous to the number of clusters.

- Each topic is associated with a probability vector, which quantifies the probability of the different terms in the lexicon for that topic. Let $t_1 \ldots t_d$ be the d terms in the lexicon. Then, for a document that belongs completely to topic T_j, the probability that the term t_l occurs in it is given by $P(t_l|T_j)$. The value of $P(t_l|T_j)$ is another important parameter which needs to be estimated by the topic modeling approach.

Note that the number of documents is denoted by N, topics by k and lexicon size (terms) by d. Most topic modeling methods attempt to learn

the above parameters using maximum likelihood methods, so that the probabilistic fit to the given corpus of documents is as large as possible.

The *probabilistic latent semantic indexing (PLSI)* method will be described in this section. The above set of random variables $P(T_j|D_i)$ and $P(t_l|T_j)$ allow the modeling of the probability of a term t_l occurring in any document D_i. Specifically, the probability $P(t_l|D_i)$ of the term t_l occurring in document D_i can be expressed in terms of aforementioned parameters as follows:

$$P(t_l|D_i) = \sum_{j=1}^{k} p(t_l|T_j) \cdot P(T_j|D_i) \tag{7.5}$$

Thus, for each term t_l and document D_i, one can generate a $N \times d$ matrix of probabilities in terms of these parameters, where N is the number of documents and d is the number of terms. For a given corpus, the $N \times d$ term-document occurrence matrix X provides the statistics for which term *actually* occurs in each document, and its corresponding frequency. In other words, $X(i,l)$ is the number of times that term t_l occurs in document D_i. Therefore, it is possible to use a maximum likelihood estimation algorithm which maximizes the product of the probabilities of terms that are observed in each document in the entire collection. The logarithm of this can be expressed as a weighted sum of the logarithm of the terms in Equation 7.5, where the weight of the (i,l)th term is its frequency count $X(i,l)$. This is a constrained optimization problem which optimizes the value of the log likelihood probability $\sum_{i,l} X(i,l) \cdot \log(P(t_l|D_i))$ subject to the constraints that the probability values over each of the topic-document and term-topic spaces must sum to 1:

$$\sum_{l} P(t_l|T_j) = 1 \quad \forall T_j \tag{7.6}$$

$$\sum_{j} P(T_j|D_i) = 1 \quad \forall D_i \tag{7.7}$$

The value of $P(t_l|D_i)$ in the objective function is expanded and expressed in terms of the model parameters with the use of Equation 7.5. A Lagrangian method can be used to solve this constrained problem. The Lagrangian solution essentially leads to a set of iterative update equations for the corresponding parameters which need to be estimated. It can be shown that these parameters can be estimated [221] with the iterative update of two matrices $[P_1]_{k \times N}$ and $[P_2]_{d \times k}$ containing the topic-document probabilities and term-topic probabilities respectively.

The matrices are randomly initialized, and normalized so that the probability values in their columns sum to one. Then, the following E- and M-steps are iteratively performed on each of P_1 and P_2 respectively:

- (E-Step) The generation probability of a document from each cluster is computed in a straightforward way by using the Bayes rule in conjunction with the current term-topic probabilities. Thus, the current entries in P_2 are used to update P_1. The matrix P_1 is normalized, so that each column sums to 1.

- (M-Step) The expected frequency of each term in the different clusters is computed by using the frequencies of the terms in the current soft assignment. This is the maximum likelihood estimate of the term-topic probabilities. Thus, the current entries in P_1 are used to update P_2. The matrix P_2 is normalized, so that each column sums to 1.

These steps are exactly analogous to the ones discussed in Chapter 2, except that the probabilities are designed for the text representation. The process is iterated to convergence. The output of this approach are the two matrices P_1 and P_2, the entries of which provide the topic-document and term-topic probabilities respectively. Many other related probabilistic methods such as LDA are commonly used in the text clustering literature [72, 73].

Documents which have low probability of belonging to their closest cluster can be declared outliers. Such an approach can also be adapted to online scenarios. In such scenarios, the parameters of the model are iteratively updated in the context of an online data stream. In such cases, only a window of the current set of documents are used for the update equations, so that the model is updated over time in order to reflect the current trends in the data. Whenever a document is encountered which does not fit well into the current list of topics, a new topic can be initiated containing this document in a dominant way. Thus, this scenario corresponds to the case of growing clusters. Such an approach, which is based on the LDA model was proposed in [503]. This approach maintains a special cluster (potential novelty cluster), with a low a-priori probability. Whenever a document is received which fits the special cluster better than the current set of clusters, the document is added to the special cluster, and it becomes a one of the normal clusters. Thus, this approach uses a probabilistic alternative to the method of creating a new cluster, and discovering novelties in the process.

7.   Conclusions and Summary

This chapter studies methods for outlier detection in categorical, text, transaction, and mixed data sets. Such data sets share a number of broad similarities with one another. Therefore, it is natural to study the outlier detection methods for these different problems in a single chapter. Most of the existing proximity-based outlier detection methods can be easily adapted to these different domains of data, as long as appropriate similarity functions are defined for such data. Mixed-attribute data sets can also be addressed by proximity-methods, as long as appropriate methods are defined for weighting the similarity of each component. Similarity computation of categorical data is an important area in its own right, and numerous methods exist for adapting similarity functions to that domain. Text data shares a number of similarities with binary data, in that it is high-dimensional and sparse. Therefore, many of the techniques for one domain can be used for the other, and vice-versa. This chapter provides an overview of the key techniques which are used for these related domain.

8.   Bibliographic Survey

The earliest work on categorical outlier detection was performed in the context of clustering algorithms [176, 186, 189, 232, 255]. In many of the clustering algorithms, the outliers are constructed as a side-product of clustering algorithms. Some of the earliest work on using probabilistic clustering for outlier detection in categorical data is proposed in [478]. This approach uses a joint probability model, which can address both categorical and mixed attributes. The approach can be effectively used for mixed data, since probabilistic models are able to normalize in an intuitively uniform way over numerical and categorical attributes. Furthermore, the approach uses a time-decaying model for the clustering component. This ensures that it can be used effectively in an evolving data set.

While linear models have not been studied extensively in the context of outlier detection of categorical data, they are a natural choice for measuring inter-attribute correlations. A number of other correspondence analysis methods [216] are available, which are specifically optimized for categorical data. However, the potential of such methods has not been fully explored in the context of outlier detection.

Proximity-based algorithms can naturally be generalized to categorical data, as long as a similarity or neighborhood measure is available in order to perform the distance (similarity) computations. However, in most of these cases, the similarity is performed with the use of relatively

straightforward overlap measures. Categorical similarity measures are crucial to all proximity-based applications such as clustering. Similarity measures in categorical data are closely related to corresponding measures in text data, because they all use aggregate statistics in order to provide lower values to attributes which are rare in occurrence. The most common example in the case of text is the use of the inverse document frequency in order to normalize the similarity measures [391]. The corresponding measure in the case of categorical data is referred to as the inverse occurrence frequency.

Many of the similarity measures defined in the literature [31, 69, 126, 291, 353], are *contextual* in the sense that they use the neighborhood of a point in order to compute similarity. Of course, the relationship between similarity derivation and neighborhood computation is circular, because neighborhood computation itself requires the computation of similarity. Therefore, many of these techniques use simpler measures such as the *Overlap* in order to define the initial neighborhood for similarity computation. Subsequently, iterative methods [126] are used in order to simultaneously refine the neighborhood and the similarity computation. Therefore, it is sometimes also helpful to develop similarity measures, which are based on the aggregate statistics, rather than on specific neighborhoods of data points.

Generally such aggregate methods either measure similarity only between same values of an attribute (using different kinds of normalizations), or they compute similarity between different values of the same attribute [75, 303]. Some of the common techniques which do not use similarity between values of different attributes are discussed in [175, 187, 391, 87]. Other measures which uses functions of both matches and mismatches on different attribute values are proposed in [42, 303, 411]. Many of these similarity measures have been tested in the context of the outlier detection problem [75]. Contextual measures [126] have also been proposed, but have not been widely explored in the context of the outlier detection problem. The work in [494] also provides some similarity measures, which can be used both in the context of categorical and mixed data. The effectiveness of these measures in the context of the outlier detection problem has been studied in the same work.

Link-based methods [461] are a natural method for modeling outliers, since links provide a way to model common attribute values between data instances. The work in [461] models the categorical values as a graph, and distances between data instances are modeled by using the connectivity of this graph. Distributed link-based methods for finding outliers in categorical and mixed data are proposed in [352]. Two pattern-based

methods for anomaly detection are discussed in [34, 127], the first of which is information-theoretic in nature.

The problem of binary and transaction data is especially challenging because of its high dimensionality. As shown in [209] direct adaptations of various methods such as replicator neural networks [464], and clustering methods [210] do not work effectively. In such data, a given transaction may contain only a very tiny fraction of the available items. The work in [209] shows how to adapt subspace methods to transaction data by finding transactions which do *not* lie in dense subspaces (or do not contain frequent patterns), rather than determining transactions which contain sparse subspaces. Another approach which is based on frequent patterns is proposed in [345]. Methods which use frequent patterns for information-theoretic anomaly detection methods are discussed in [34, 267, 410].

In the context of text data, outliers are explored in terms of either data transformation methods to reduce noise, or in terms of determining the actual data points which are outliers. For the former problem, Latent Semantic Indexing is commonly used. The problem of Latent Semantic Indexing [133] (LSI) is closely related to Principal Component Analysis [244], which is studied in some detail in Chapter 3. Latent Semantic Indexing was originally proposed as a method for retrieval of text [133], and its effects on improving the quality of similarity in text were observed in [355].

The problem of topic detection and tracking has been studied both in the unsupervised scenario [26, 37–39, 77, 254, 485, 503], and in the supervised scenario [484, 486]. Some of these methods have also been generalized to social streams, which contain a combination of text and linkage information [27, 362]. Most of the methods for novelty detection are based on proximity models, though a few methods [503] are also based on probabilistic models. A variety of probabilistic models such as PLSI and LDA [72, 73, 221] can be used for outlier analysis in text data. In the supervised scenario, it is assumed that training data is available in order to enhance the novelty detection process. This can be achieved either by providing examples of the rare class, the normal class or both. Such methods are often quite useful for detecting novel events in conventional news streams or social media such as *Twitter* [362]. A probabilistic model for determining novelties in text streams is proposed in [503]. Another aggregate method for detection of correlated bursty topic patterns from coordinated text streams is proposed in [458].

9. Exercises

1. Download the *Adult data* from the UCI Machine Learning Repository [169]. Determine outliers with the use of k-nearest neighbor algorithm on the categorical attributes only by using:

   ■ Match-based similarity measures.

   ■ Inverse occurrence frequency similarity measure.

   How do the outliers compare to one another?

2. Repeat Exercise 1 by including the numerical attributes in the analysis. Use a euclidian distance measure on the numerical attributes. Do you obtain the same outliers?

3. Compute the principal components of the mixed representation of the *Adult data* with and without attribute-specific normalization. Determine the outliers by using the χ^2 statistic on the sum of the squares of the deviations along the different principal components. Do you obtain the same set of outliers in the two cases?

4. Repeat the Exercises 1, 2, and 3 with the use of EM-based modeling. How do the outliers compare with the ones obtained in the previous exercises?

5. Repeat the Exercises 1, 2 and 3 with the use of the *Breast Cancer* data set. Use the rare classes in the data set as ground truth in order to construct an ROC curve in these cases. In which case do you obtain the best results?

6. Download the 20-newsgroups data set [517]. Use the following methods to determine the outliers:

   ■ Use a k-nearest neighbor approach with cosine similarity.

   ■ Use the probabilistic modeling approach, and report the data points with the least fit as the outliers.

   Do you obtain the same outliers in the two cases?

7. Repeat Exercise 6 with the Reuters-215788 data set [516].

Chapter 8

TIME SERIES AND MULTIDIMENSIONAL STREAMING OUTLIER DETECTION

"To improve is to change; to be perfect is to change often."– Winston Churchill

1. Introduction

The data generated by many applications is a continuous temporal process. Therefore, the temporal context is particularly important in outlier analysis. In many cases, the detection of unusual events needs to be performed in a time-critical manner. This is also referred to as *streaming outlier detection*. In some cases, the stream may not be available in real time, but may be available at a later stage for offline processing. In such cases, the advantage of hind-sight can allow the discovery of better outliers with more sophisticated models.

The temporal and streaming outlier detection scenario arises in the context of many applications such as sensor data, mechanical system diagnosis, medical data, network intrusion data, newswire text posts, or financial posts. In such problem settings, the assumption of *temporal continuity* plays a critical role in defining and determining outliers. It is worth noting that outlier analysis has diverse formulations in the context of temporal data, in some of which temporal continuity is more important than others. In *time-series data*, temporal continuity is immediate, and expected to be very strong. In multidimensional data with a temporal component (eg. text streams), temporal continuity is much weaker, and is present only from the perspective of *aggregate trends*. Therefore, outliers in these cases are defined in somewhat different ways:

- *Abrupt Change Detection in Time Series:* These correspond to sudden changes in the trends in the underlying data stream. In

such cases, the issues of temporal continuity are critical, and the outlier is defined as unusual because it exhibits a *lack of continuity* with its immediate [479] or long-term history. For example, sudden changes in time series values (with respect to immediate history), or distinctive shapes of subsequences of the time series (with respect to long-term history) are identified as outliers. In cases, where the entire time-series is available off line, the advantage of hind-sight may be leveraged to compare a shape both with past and future instances. In time series analysis, *vertical* analysis is more important where each individual series (or dimension) is treated as a unit, and the analysis is primarily performed on this unit. In the event that multiple series are available, cross-correlations may be leveraged, though they typically play a secondary role to the analysis of each individual series. This is because time-series data is *contextual*, which imposes strong temporal locality on series values.

- *Novelty and Change Detection in Multidimensional data:* In this case, the data contains individual multidimensional points, which are independent of one another, and the issue of temporal continuity is much weaker as compared to time series data. For example, in a time series derived from sensor data, two successive data values are often almost identical. On the other hand, an individual text document (multidimensional data point) in a stream of newswire articles may normally be quite different from its immediately preceding article. The data point needs to be compared to a much larger history of the documents in order to construct a robust outlier model. The anomalies in multidimensional streaming data could either correspond to *time-instants* at which *aggregate* trends have changed, or *individual* data point novelties, which vary from these aggregate trends. The latter corresponds to incoming data points in the stream, the likes of which have not been seen earlier in the stream. Such cases are almost identical to offline outlier analysis. The temporal aspects of the stream are important only to the extent that the novelties are defined with respect to the *past* behavior of the data stream, rather than the entire data set. In such cases, all the dimensions of a record are treated as a unit, and the anomalies are identified by aggregating the temporal trends in these units. This kind of analysis is *horizontal*.

For example, a first story on a particular topic in a text stream is considered a novelty outlier, while a change in the aggregate trend of the topics in the text stream is a change point outlier. It
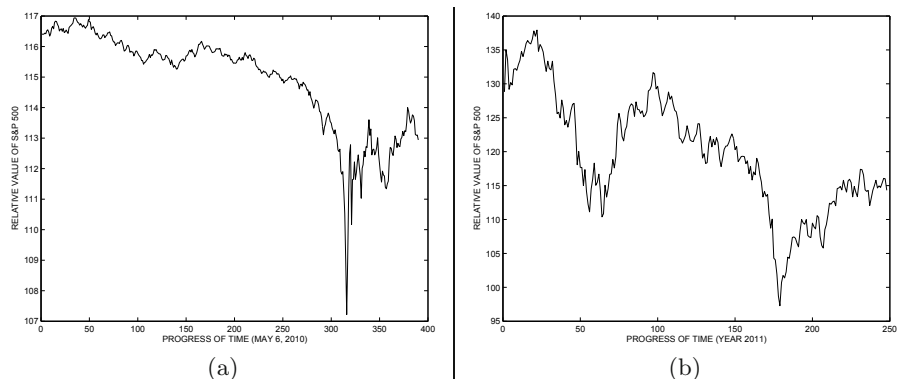
Figure 8.1. Behavior of the $S\&P500$ on (a) the day of the flash crash (May 6, 2010), and (b) year 2001

is worth noting that novelties are often *trend-setters*. Therefore, at a later stage, similar data points may no longer be considered novelties, but may become a normal part of the data. Thus, while the temporal aspects of the stream are used, they are slightly less important from an *immediate* continuity perspective. However, significant changes from aggregate trends continue to be important. For example, it has been shown in [160] that interesting outliers in widely separated domains such as intrusion detection and text news story analysis are related to the problem of change detection.

Both the above definitions are consistent with Hawkins' notion of outliers, which was introduced at the very beginning of the book.

Outlier detection in a time-series can also be divided into two categories. This depends on whether the values at specific time stamps are classified as outliers because of sudden changes (contextual anomalies), or whether entire time-series or large subsequences within a time series are classified as outliers because of their unusual shapes (collective anomalies). For example, consider the two cases illustrated in Figures 8.1(a) and (b). The first time series illustrates the relative behavior[1] of the $S\&P500$, on May 16, 2010 which was the date of the stock market flash crash. This is a very unusual event *both* from the perspective of the deviation during the stock value drop, *and* from the perspective of the time-series shape. On the other hand, Figure 8.1(b) illustrates the behavior of the $S\&P500$ during the year 2001. There are two significant

---

[1]The tracking ETF SPY was used.

drops over the course of the year both because of stock market weakness and also because of the 9/11 terrorist attacks. While the specific time-stamps of drop may be considered somewhat abnormal based on deviation analysis over specific windows, the actual shape of these time-series is not unusual, since it is frequently encountered during different kinds of stock market drops. It is evident from the aforementioned discussion that the determination of time-series of unusual shapes is much more challenging than pure deviation-based analysis. The latter case arises in many applications such as vehicle diagnostics or medical conditions, in which an unusual condition corresponds to an unusual shape of the series.

In some cases, labels may be available in order to supervise the anomaly detection process. This is true of both time-series outlier detection and multidimensional outlier detection. In the case of time-series, the labels may be associated with time-instants, with time intervals, or they may be associated with the entire series. In the multidimensional case, labels are associated with the individual data points.

Thus, temporal data allows multiple ways of defining anomalies. This is consistent with the observation in the first chapter: *"The more complex the data, the more the analyst has to make prior inferences of what is considered normal for modeling purposes."* The appropriate way of defining anomalies is highly application-dependent, and in some cases even the same scenario may require the determination of different kinds of anomalies. For example, in the case of sensor data, it may sometimes be helpful to use deviation detection methods for noise outlier filtering, whereas in other cases, unusual shapes in a medical stream can diagnose heart conditions such as arrythmia. This chapter will investigate some of the more common scenarios for outlier analysis in temporal data.

Even though temporal data may comprise either continuous data or discrete sequences, this chapter focusses on continuous data in order to preserve homogeneity in presentation. This is because the concept of temporal continuity is defined differently in discrete data and continuous data. In the case of discrete data, a lack of ordering in the data values significantly affects the nature of the methods used for outlier analysis. It should be noted that a continuous series can always be discretized into symbolic data, though at the loss of some information. Thus, some of the methods for outlier detection are common to both kinds of data. The material in the two chapters on temporal and sequence-based outlier detection will be carefully organized, so as to point out the relationships among different scenarios. The case of discrete data will be discussed separately in the next chapter.

This chapter is organized as follows. In the next section, algorithms for detection of outlier *instants* in streaming time series will be presented. Typically, these methods are based on deviation-detection from predicted values at time instants. The detected anomalies are contextual outliers. In section 3, methods for detecting unusual shapes in time-series data will be presented. These are collective outliers. Methods for multidimensional streaming outlier detection are discussed in section 4. Section 5 presents the conclusions and summary.

2. Prediction-based Outlier Detection of Streaming Time Series

The most common application of temporal outlier detection is that of detecting *deviation-based* outliers of specific time-instants with the use of regression-based forecasting models. These anomalies are contextual anomalies, because they define abnormalities at specific instants of the data, on the basis of relationships between data values at adjacent time instants. Such an approach can either be used to detect sudden changes in the underlying process, or to filter noise from the underlying streams. Deviation-based outliers in time-series are very closely related to the problem of time-series forecasting, since outliers are declared on the basis of deviations from expected (or forecasted) values.

In these methods, *temporal continuity* plays a significant role, since it is assumed that time-series data values are highly correlated over successive instants, and do not change abruptly. Deviation-based outliers use the predicted value at the next time-stamp through a variety of regression models. The correlations in a single time-series, or across multiple series, may be used in order to perform the prediction. Thus, two kinds of correlations are used:

- **Correlations across time:** This is essentially the same principle as temporal continuity, and a variety of auto-regressive models are used for forecasting and other applications. This general area is also referred to as stream *filtering. Significant* deviations from the *expected* predictions are utilized in order to define outliers. Such *significant* deviations are experienced as *abrupt* changes, which are different from the smooth long-term concept drift which is also often experienced in data streams.

- **Correlations across series:** Many sensor applications result in time series which are often closely correlated with one another. For example, a bird call at one sensor, will typically also be recorded by a nearby sensor. In such cases, one series can frequently be

used in order to predict another. Deviations from such expected predictions can be reported as outliers.

While regression modeling techniques were discussed in Chapter 3, they were studied in the context of non-temporal data. This chapter will study their application in the context of temporal data, which has a number of unique characteristics in terms of data specification and modeling. While the core theory behind both domains is essentially the same, the way in which it is applied is different. Therefore, regression modeling techniques will be re-visited in this chapter. The subsequent discussion assumes familiarity with the material presented in Chapter 3.

2.1   Autoregressive Models

Autoregressive models (AR) are particularly useful in the context of univariate time-series. Let $X_1 \ldots X_t \ldots$ be the values in the univariate time series. In the auto-regressive model, the value of X_t is defined in terms of the values in the last window of length p.

$$X_t = \sum_{i=1}^{p} a_i \cdot X_{t-i} + c + \epsilon_t$$

A model which uses the last window of length p is referred to as an $AR(p)$ model. The values of the regression coefficients $a_1 \ldots a_p, c$ need to be learned from the training data, which is the previous history of the time series. A set of linear equations between the coefficients can be created, by using each time stamp in the training data, along with its immediate history of length p. When the number of time-stamps available is much larger than p, this is an over-determined system of equations. The coefficients $a_1, \ldots a_p, c$ can be approximated with *least-squares regression*, in order to minimize the square-error of the over-determined system. The details of the solution are provided in section 2.1 of Chapter 3. As discussed in Chapter 3, a matrix of size $(p+1) \times (p+1)$ needs to be inverted in order to determine the co.

In the above system of equations, the value of c is a constant, and the value of ϵ_t represents the noise, or the *deviation* from the expected values. Large absolute values of this deviation represent the anomalies in the underlying data. Therefore, the extreme value analysis techniques of Chapter 2 can be used on ϵ_t in order to determine those deviations which vary significantly from the expected values. These values are assumed to be independent and identically distributed random variables, which are drawn from a normal distribution. When a longer time series is available, the number of available values of ϵ_t is large. Therefore, the normal distribution can be utilized in order to compute the significance

level of the deviations. In the case of a shorter time series, the t-value test can be used in order to provide a more accurate estimation of the level of significance.

The auto-regressive model can be made more robust by combining it with a moving average model (MA Model). This model predicts subsequent deviations on the basis of the past history of deviations. The moving average model is defined as follows:

$$X_t = \sum_{i=1}^{q} b_i \cdot \epsilon_{t-i} + \mu + \epsilon_t$$

The aforementioned model is also referred to as $MA(q)$. The value of μ is the mean of the time-series. The values of $b_1 \ldots b_t$ are the coefficients, which need to be learned from the data. The moving average model is quite different from the auto-regressive model, in that it relates the current value to the mean of the series and the previous history of deviations, rather than the previous history of values. Here the values of ϵ_i are assumed to be *white noise* error terms, which are uncorrelated with one another. The error terms ϵ_i are not part of observed data, but are also dependent on the values of the coefficients. Therefore, the set of relationships implied by the model is inherently non-linear. Typically, iterative non-linear fitting procedures are used instead of the linear least squares approach [387].

In practice, both the previous history of deviations and values may be important for calculating expected values. The two models can then be combined with p autoregressive terms and q moving average terms. This corresponds to the *Auto-Regressive Moving Average (ARMA)* model. This corresponds to the following model:

$$X_t = \sum_{i=1}^{p} a_i \cdot X_{t-i} + \sum_{i=1}^{q} b_i \cdot \epsilon_{t-i} + c + \epsilon_t$$

The aforementioned model is the $ARMA(p, q)$ model. A key question here is about the choice of the parameters p and q in these models. If the values of p and q are picked to be too small, then the model will not fit the data well, and the absolute values of all noise terms will be too high to provide information about true anomalies. On the other hand, if the values of p and q are picked to be too large, then the model is likely to overfit the data. In general, it is good to pick the values of p and q as small as possible, so that the model fits the data well. While the model is always assumed to be linear by default, many non-linear variations of the model are also available. The use of such non-linear models is a

double-edged sword; on the one hand, it increases the expressive power of the model, and on the other hand, it could lead to overfitting.

In some cases, the time-series may have some random characteristics, as a result of which it may drift away from the mean. A random walk time series would be an example of such a situation. This is referred to as the *non-stationarity* of the time series. In such cases, the series can be de-trended by first differencing the time-series before ARMA modeling. Such a modeling approach is referred to as *Autoregressive Integrated Moving Average Model (ARIMA)*. Specific details of the solution techniques for these models are beyond the scope of this book, are discussed in detail in [387].

2.2   Multiple Time Series Regression Models

In many applications, multiple time series are available which can be used in order to perform the prediction of time-series values in a more robust way. The idea is that different time series may often contain the same information, and may sometimes contain *lag correlations*. For example, a bird-call at one sensor will also be heard at a nearby sensor, though with a small lag. Such lag correlations can be used in order to make more robust predictions. The standard regression-based model can be generalized to this case, by defining the variable X_t in terms of its past history, as well as the history of other time series. A number of common methods can be used in order to perform the prediction across multiple time series. These methods are discussed in the following different subsections.

2.2.1   Direct Generalization of Auto-Regressive Models.

Most of the auto-regressive models discussed in the previous sections can be generalized to the case of multiple time series, by incorporating coefficients for the other series in addition to the current series. Therefore, if $X_t^1 \ldots X_t^d$, be the tth value of the d different series, then the simple auto-regressive model expresses X_t^j as follows:

$$X_t^j = \sum_{k=1}^{d} \sum_{i=1}^{p} a_i^k \cdot X_{t-i}^k + c^j + \epsilon_t^j$$

Thus, at each time instant, a total of d different residuals denoted by ϵ_t^j are generated. Large absolute values of ϵ_t^j are reported as anomalies. As before, the normal distribution assumption (or the t-distribution) can be used in order to determine the level of significance of the different anomalies. In general, the values of ϵ_t^j for a *fixed* value of of j are assumed to be drawn from a normal or t-distribution. This is because

the values of ϵ_t^j across different values of j are likely to have different means and standard deviations, and are also likely to be correlated with one another. This broad formulation can also be extended to the auto-regressive moving average (ARMA) and auto-regressive integrated moving average models (ARIMA).

One observation is that the number of parameters $p \cdot d$ becomes large, when the number of time-series d is large. This can increase the complexity of the least-squares regression technique. The solution to this problem requires the inversion of a matrix of size $(p \cdot d + 1) \times (p \cdot d + 1)$, as discussed in Chapter 3. This can become computationally challenging with increasing number of time series d. Furthermore, this process needs to be repeated d different times for each of the streams. An exponential forgetting mechanism can also be incorporated in order to give more importance to the recent history of the stream in learning the cross-stream and auto-regressive coefficients. This is because the stream auto-correlations and cross-correlations may also change significantly over time.

A number of methods have been proposed in the literature in order to speed up the regression process. For example, the *Muscles* and *Selective Muscles* techniques proposed in [491] can perform the regression analysis much more efficiently. The *Muscles* technique is a relatively straightforward application of the least squares technique for multivariate regression as discussed above. A variation known as *Recursive Least Squares* is employed to solve the regression more efficiently. Another difference from the standard multivariate autoregressive model is that the current values of the time stamps in the other series are also used for the purposes of prediction. This provides a more accurate estimation of the time stamps, and is also therefore likely to improve the accuracy of anomaly detection.

On the other hand, the *Muscles* technique is quite slow, since it tries to use the values in *all* the different series in order to perform the prediction. In practice, different subsets of the time-series may be well correlated with one another. The main observation in *Selective Muscles* is that the vast majority of advantage from multivariate regression may be obtained by selecting a small subset of the series for predictive purposes. Therefore, for each time series $\overline{X^j}$ which needs to be predicted, a subset S_j of streams needs to be determined. We note that when $|S_j| << d$, this greatly reduces the number of streams for predictive purposes. A greedy algorithm is proposed in [491]. The first series picked is the one with the highest correlation coefficient to $\overline{X^j}$. Subsequently, the next series which is picked is one which minimizes the expected estimation error over the current set of choices. This process is continued till the k series are

picked with the highest predictive power. These series are then used in order to make the predictions for the jth time series. These predictions are used for anomaly detection by finding values at time stamps which deviate significantly from expected values. The subset selection process can be performed periodically in the context of an evolving stream, in order to minimize the overhead resulting from the selection algorithm itself. A number of other methods [24] can also be used in order to achieve the goal of stream selection with the use of regression analysis.

2.2.2 Principal Component Analysis and Hidden Variable-based Models.

The aforementioned models are based on treating a single stream as the dependent variable, then trying the predict this stream by assuming that the other variables as independent variables. While such an approach optimizes the regression process to a particular stream, it can be quite slow, since the analysis needs to be performed d different times. For example, in the above description, the stream subset selection needs to be performed for each of the d different streams, and then a separate regression analysis needs to be performed for each of the d streams. The apparent advantage is that the regression coefficients are optimized to each of the different streams, and therefore the outlier can be localized to a particular time-series from the group. Such an approach is sometimes not effective, because the examples in Chapter 3 showed that dependent-variable analysis can sometimes be very sensitive to the presence of noise and outliers. A specific example was presented in Figure 3.3 of Chapter 3, where a complete break down of (dependent-variable) regression analysis was caused by an outlier.

On the other hand, global directions of correlation such as PCA, which are not optimized to any particular variable, are usually much more robust to the presence of outliers. Such analysis leads to a set of hidden variables, which can be used in order to generate *all variables* in the stream simultaneously. Of course, these hidden variables are not optimized to predict any specific time series, and therefore cannot localize the outlier prediction to any particular time series (in general). However, this approach generally turns out to be much more robust to noise and outliers for predictive purposes.

In the temporal context, the principal component analysis method of Chapter 3 needs to be modified in order to account for correlations between streams, and across time. The most straightforward generalization of principal component analysis uses the correlations across different streams, but it ignores the correlations across time. In this instantiation, the pairwise correlations among the different dimensions are used in an exactly identical way to the methods discussed in Chapter 3. Specif-

ically, a $d \times d$ covariance matrix is maintained, which represents the covariance between the streams i and j. The assumption is that the values of the stream at each time-stamp are available for predictive purposes. The principal components of this covariance matrix provide the correlations between the different streams. Significant deviations from these correlations are reported as outliers.

The covariance matrix can easily be maintained incrementally for a data stream. Note that the covariance between the series $\overline{X^j}$ and $\overline{X^k}$ can be expressed as follows:

$$Cov(\overline{X^j}, \overline{X^k}) = \frac{\sum_{i=1}^{t} X_i^j \cdot X_i^k}{t} - \frac{\sum_{i=1}^{t} X_i^j}{t} \cdot \frac{\sum_{i=1}^{t} X_i^k}{t}$$

Note that the computation of the above requires the maintenance of (i) the d additive sums of the values of each variable in the stream, and (ii) the d^2 additive sums of the pairwise products of the time series values in the stream. This can easily be achieved in an incremental setting.

The major disadvantage of the above model is that it only accounts for the correlation across streams, and does not account for the correlations *across* time, as is achieved in auto-regressive models. For example, the covariance matrix does not change, if the values on the time stamps are re-ordered randomly, as long as all the series are re-ordered in the same way. However, correlations across time can also be accounted for by using window-based analysis.

The PCA method can be generalized to window-based analysis, by using a window of length p in order to create the covariance matrix. For the d different time-series there are $d \cdot p$ different values in the last window of length p. This can be used in order to create a $d \cdot p \times d \cdot p$ co-variance matrix. Each entry of this covariance matrix provides the relationship both across streams and time over the past window of length p. As in the previous case, this covariance matrix can also be maintained incrementally in a data stream setting. Therefore, the eigenvectors of this covariance matrix provide a generative model for the values in the data stream. Significant deviations from this model represent deviations in the underlying data stream.

This approach requires the computation of the eigenvectors of a co-variance matrix of size $d \cdot p \times d \cdot p$. This can sometimes be computationally expensive. In the case of multivariate auto-regressive models, a similar complexity is required, where a $(d \cdot p + 1) \times (d \cdot p + 1)$ matrix needs to be inverted for determining the regression coefficients. In the case of PCA however, only the top k eigenvectors are required, where $k << p \cdot d$. This can be achieved more efficiently with a progressive eigenvector extraction algorithm. Furthermore, in the case of multivariate auto-regressive mod-

els, the process needs to be repeated d times, corresponding to setting each of the variables as the dependent variable. On the other hand, since PCA is a global approach, which does not treat any particular stream as special, it needs to be performed only once in order to determine the hidden variables in the stream. These hidden variables can be used in order to make predictions in the stream. We further note that such an approach can account for lag-correlations between different streams, as long as the lag is less than the window size p. We note that it is possible to generate a new time series containing only k components $Y_t^1 \ldots Y_t^k$ by projecting the block of size $p \times d$ onto the k different eigenvectors at each time stamp. These correspond to the *hidden variables* in the time series. Since these k components are uncorrelated with one another, univariate auto-regressive methods can be applied to each of the k streams $\overline{Y^j}$ in order to predict the tth values. The deviation from the expected value in $\overline{Y^j}^t$ can be fitted to the normal distribution or t-distribution. Another possibility is to create a composite deviation by computing the sum of the squares of the k components. Since this is the sum of the squares of k normal distributions, a χ^2-distribution can also be used to measure the significance of the composite level of deviation.

Numerous variations of this broader principle have been proposed in the literature, the most well known of which is SPIRIT [357]. In this approach, a $d \times d$ matrix $[W]_{d \times d}$ of participation weights is used, where the jth hidden time series $\overline{Y^j}$ (vector) can be expressed in terms of the d different time series vectors as follows. Therefore, we have:

$$\overline{Y^j} = \sum_{i=1}^{d} W_{ij} \cdot \overline{X^i}$$

In a sense, each column of the matrix W represents a principal direction, and therefore we have $W^T \cdot W = I$. The hidden variable Y_t^j for the jth column of the matrix W is obtained by projecting the d-dimensional vector $(X_t^1 \ldots X_t^d)$ onto this principal direction. This corresponds to the dot product between the eigenvector representing the jth column, and the d different values at time stamp t in order to yield the hidden variable at time t. As in the case of PCA, most of the variance is captured in the top k principal directions. Without loss of generality, assume that the columns of matrix W are ordered, so that these principal directions correspond to the first k columns of W. Then, the vector $\overline{X^j}$ can be (approximately) expressed in terms of the participation weights, by using the hidden variables to create a linear combination of only these top k
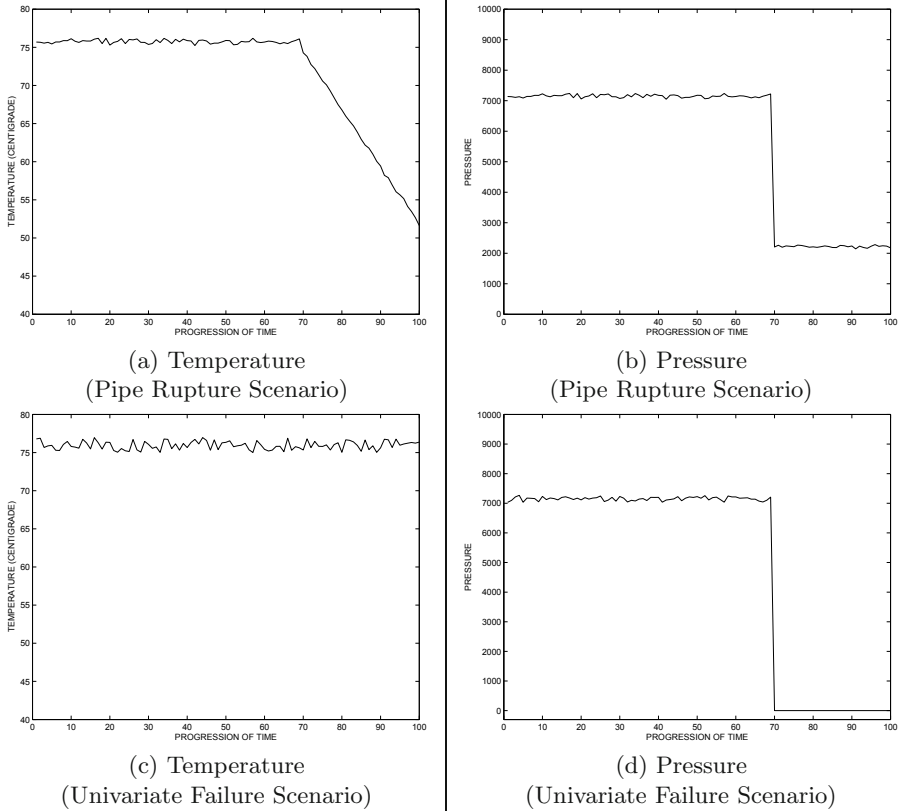
Figure 8.2. Behavior of temperature and pressure sensors due to (a, b) Pipe Rupture, and (c,d) Failure of pressure sensor

top principal components:

$$\overline{X^j} \approx \sum_{i=1}^{k} W_{ji} \cdot \overline{Y^i}$$

The SPIRIT approach uses an efficient update process, so as to update the eigenvectors of the co-variance matrix dynamically, so that the reconstruction error is small. Furthermore, an exponential forgetting mechanism can be used in order to ensure that the co-variance matrix is based on the recent history of the stream, rather than the entire stream of data points.

2.3 Supervised Outlier Detection in Time Series

The aforementioned methods determine the significant deviations from the expected values, and report them as outliers. In many cases, such

deviations could have many causes which are not necessarily indicative of events of interest. For example, in the context of an environmental monitoring applications, many deviations may be result of the failure of the sensor equipment, or other spurious event which causes deviations in sensor values. This may not necessarily reflect an anomaly of interest. While anomalous events often correspond to extreme deviations in sensor stream values, the precise causality of different kinds of deviations may be quite different. Therefore, in the context of noisy time-series data, the anomalies of interest may be embedded among a number of spurious abnormalities, which may not be of any interest to an analyst. For example, consider the case illustrated in Figure 8.2, in which we have illustrated the temperature and pressure values inside pressurized pipes containing heating fluids. The figures 8.2(a) and (b) illustrate values on two sensors in a pipe rupture scenario. The figures 8.2(c) and (d) illustrate the values of the two sensors in a scenario where the pressure sensor malfunctions, and this results in a value of 0 at each tick. In the first case, the readings of both pressure and temperature sensors are affected by the malfunction, though the final pressure values are not zero, but the reflect the pressure in the external surroundings. The readings on the temperature sensor are not affected at all in the second scenario, since the malfunction is specific to the pressure sensor.

So how does one distinguish between noise and true anomalies, which are of interest to the analyst? The time-tested method for making the approach more sensitive to analyst interest, is to use *supervision* from previous examples. In the multivariate scenario, the truly anomalous events of interest may be detected only from the *differential* behavior of the deviations across different time series data streams. In such scenarios, supervision can be very helpful in distinguishing the true anomalies from the spurious abnormalities in the time series data stream. The approach discussed in [9] proposes a method for abnormality detection in spuriously populated data streams. It is assumed that the true events of interest are available as the *ground truth time stamps*, $T(1) \ldots T(r)$, which are used for supervision. These are referred to as *primary abnormal events*. In addition, a number of examples of time stamps which correspond to the common spurious deviations are also available. If desired, the training data for such spurious events can also be continuously generated during the progress of the stream by using the previous time instants of large average deviations of sensor values, which did not correspond to true anomalies. These time stamps are referred to as *secondary abnormal events*.

The overall process of event prediction is to create a composite alarm level from the error terms in the time series prediction. The first step

is to use a univariate time-series prediction model in order to determine the error terms at a given time-stamp. This step is identical to the methods discussed earlier in this section, and any of these methods can be used, with or without exponential forgetting factors. These error terms are then normalized to Z-values for the d streams, and are denoted by $z_1 \ldots z_d$. The first step is to prune those streams which do not have large absolute deviations within a certain lag of the ground truth events of interest. Under the normal distribution assumption, a threshold of three standard deviations was used in [9]. At this point, a set of $s \leq d$ streams may remain after pruning. Then, the goal is to learn a vector of coefficients, $\alpha_1 \ldots \alpha_s$, which can create the composite alarm level Z:

$$Z = \sum_{i=1}^{s} \alpha_i \cdot z_i$$

This vector of *discrimination coefficients* $\alpha_1 \ldots \alpha_s$ should be picked in order to maximize the differences in the alarm level between the primary and secondary events.

In order to achieve this goal, the composite alarm level is averaged at the time-stamps for all primary events of interest in order to create an alarm level $Z^p(\alpha)$ which is function of α. A similar secondary alarm level $Z^s(\alpha)$ is computed. Then, the vector α is learned by optimizing $Z^p(\alpha) - Z^s(\alpha)$, subject to the normalization constraint that $\sum_{i=1}^{s} \alpha_i^2 = 1$. This objective function essentially provides the maximum discrimination between the primary and secondary events.

In practice, the overall anomaly detection and learning processes are executed simultaneously, as new primary and secondary events are encountered, and the vector of discrimination coefficients can be learned more accurately over time. The composite alarm level can either be reported as an outlier score, or thresholding can be used in order to provide discrete time stamps at which anomalous events are reported.

3. Time-Series of Unusual Shapes

Much of the work on time-series outlier detection is to determine *unusual changes or very large deviations* from the underlying series. However, certain kinds of deviations are based not only on the individual deviations of the data points, but also on the *shapes* of specific portions of the time-series with respect to the other extracted portions. For example, consider the case of the flash-crash illustrated in Figure 8.1(a). In this case, the stock marker showed very unusual behavior over *multiple time stamps* by first dropping precipitously over a very short period, and then recovering very quickly to almost original levels. This unusual

behavior had a different causality from most other drops in the stock market. Therefore, the *shape* of the series is different from other large deviations. It is clear that determining large deviations from previous time-stamps cannot *differentially* discover such anomalies, because the entire series (or subsequence) needs to be viewed from the perspective of other normal series in the database.

The goal in such methods is to determine windows of the data (or subsequences) in which a given series behaves differently from a database of multiple sequences. Unlike the cases discussed earlier in this chapter where outliers are defined by *a single position*, the outliers correspond to multiple consecutive time stamps in this case. Thus, the previous cases correspond to contextual outliers, whereas this case corresponds to collective outliers. Two possibilities may exist within this case of anomaly detection:

- *Full Series Anomaly:* In this case, the shape of the entire series is treated as an anomaly. However, in most cases, unless the database of sequences corresponds to a relatively short segment of time-stamps, the noise variations within the series will mask the anomalous shape. This is analogous to the problems of noise encountered in detecting outliers in high-dimensional data. Full series anomaly detection also requires somewhat more sophisticated pruning techniques in order to account for the challenges in distance computation for adaptations of k-nearest neighbor techniques. However, in some spatial applications, it is required to use the entire series for anomaly detection. An example of such a scenario along with a corresponding method [469] will be studied in the context of unusual shape detection in spatial data in section 5 of Chapter 10.

- *Subsequence-based Anomaly:* In this case, anomalous shape is detected over small windows of the time-series. This is analogous to the problem of detecting subspaces of high dimensional data, which show anomalous behavior.

This section will focus on the problem of subsequence anomaly detection. Interested readers are referred to section 5 of Chapter 10 for methods[2] on full series anomaly detection. In all cases, it is assumed that the series has been normalized to zero mean and unit standard deviation, since it is not meaningful to compare series of different means and amplitudes.

---

[2]The approach in Chapter 10 uses a different (rotation-invariant) distance function in make it relevant to spatial data, though the general principles remain the same for applying it to time-series data.

3.1   Transformation to Other Representations

There are two kinds of transformations which may be used in order to compare segments of a time-series with other portions.

- *Numeric Multidimensional Transformation:* In one case, the series (or each subsequence of the series) is transformed into a multidimensional vector. The representation of each dimension corresponds to numeric coefficients in standard vector space. Proximity can be computed on this representation with the Euclidean distance. Therefore, many of the methods discussed in Chapter 4 of this book can be used in order to determine proximity-based anomalies.

- *Discrete Sequence Transformation:* In the second case, the series can be transformed to symbolic representations by using discretization methods. In such cases, the methods discussed in Chapter 9 can be used in order to determine unusual shapes in the time series. Such methods can also be used for fast approximations to the exact nearest neighbor distances, and can therefore be used to improve the *efficiency* of the discord-discovery process.

This section will discuss the transformation of different kinds.

3.1.1   Numeric Multidimensional Transformations.   The simplest possible transformation would be to consider each window of length n in the time-series as a multidimensional vector of length n. Other methods such as *Discrete Wavelet Transform (DWT)* and *Fast Fourier Transform* are available for compressing the series with the multi-scale approach into numeric coefficients [366]. The smaller coefficients can be pruned from the representation in order to enable more efficient similarity search in the context of proximity-based outlier detection. Since such representations are well known to work effectively in the context of similarity search with time-series, they can also be used to implement proximity-based anomaly detectors. Euclidean distances can be utilized on these representations in order to measure proximity. In particular, given two subsequences (or transformed representations) of length n denoted by $A = (a_1 \ldots a_n)$ and $B = (b_1 \ldots b_n)$, the Euclidean distance between them can be computed as follows:

$$Dist(A, B) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2} \tag{8.1}$$

A standard multidimensional proximity method can be used for determining discords in the data set. The nearest neighbor distance, or the

k-th nearest neighbor distance to a series can be be used as the anomaly score. A major challenge of this method is that such an approach may result in the creation of a large number of subsequences. Unless effective pruning methods are developed, this will require $O(N^2)$ time, where N is the total number of subsequences. This problem was also observed in Chapter 4 for the case of proximity-based multidimensional outlier detection. The section below will also discuss some methods for improving the pruning efficiency with the use of discrete approximations.

3.1.2   Discrete Sequence Transformations.   In theory, it is possible to convert continuous time series to discrete data, and then use the methods discussed in the next chapter for anomalous shape discovery. This transformation is typically performed on windows of the data, and it leads to a compressed and approximate representation of the underlying time series. Such methods can either be used for *stand-alone* anomaly detection of the discrete sequences with the methods discussed in Chapter 9, or for improving the *efficiency* of the nearest neighbor detectors discussed above by quick approximate representations and pruning. The latter case will be discussed in the next subsection. A variety of discrete transformations are possible such as (symbolically discretized representations of) the means over specific windows, slopes over specific windows, discrete wavelet coefficients, and Fourier transform coefficients. The specific representation which is used should depend upon the application domain at hand.

A commonly used discretization technique is the *Symbolic Aggregate Approximation (SAX)* [305]. In this method, *Piecewise Aggregate Approximations (PAA)* are used in order to represent the time series. This method comprises two steps:

- *Window-based Averaging:* The series is divided into windows of length w, and the average time-series value over each window is computed.

- *Value-based Discretization:* The (already averaged) time-series values are discretized into a smaller number of approximately *equi-depth* intervals. The idea is to ensure that each symbol has an approximately equal frequency in the time-series. The actual interval boundaries are constructed by assuming that the time series values are distributed with a Gaussian assumption. It is to be noted that the mean and standard deviation of the (windowed) time series values need to be computed in order to construct the Gaussian distribution. The quantiles of the Gaussian distribution are used to determine the boundaries of the intervals. Typically,

the values are discretized into 3 or 4 intervals for the best results [258]. Each such equi-depth interval is mapped to a symbolic value. This creates a symbolic representation of the time-series.

It is important to note that symbolic discretization does lose a significant amount of information about the underlying time-series. For example, the symbols provide no information about how close or far the different intervals are from one another. Nevertheless, such approximations are useful in streaming scenarios because of their simplicity and ease in construction.

3.2   Distance-based Methods

The *Hotsax* approach [258] uses the standard numeric Euclidean distances introduced earlier in this section in order to define distance-based outliers. Furthermore, it uses the discrete approximations in order to perform pruning. The euclidian distances to the k-nearest neighbor is used in order to determine the outlier scores of each subsequence. Furthermore, where it is desirable to determine only the top-n outliers.

The outlier analysis is performed over windows of length K. These subsequences are extracted from the time-series and the nearest euclidian distances are determined. The euclidian distance is computed using Equation 8.1. Care must be taken to compare a window of values with non-overlapping windows, in order to minimize the self-similarity bias of overlapping windows [258]. Therefore, an overlapping window is not included in the computation of the k-nearest neighbor. In principle, a variety of other similarity functions exist for time-series data, the most prominent of which is *Dynamic Time Warping*. Different kinds of similarity functions may be used to determine qualitatively more effective outliers, though the euclidian function has the advantage that it can be efficiently computed and lends itself to an effective pruning methodology, as discussed subsequently. Therefore, more general distance functions may require the use of sophisticated indexes for efficient retrieval. The reader is referred to [190] for a review of time-series similarity measures and time series indexing.

The discretized versions of time series discussed in the last section are often used for scenarios where quick approximations are needed for *pruning purposes*, and the anomaly scores are computed on the basis of the original numeric representations [258]. Therefore, any imperfections in the representation affect the *efficiency* of the results, rather than the *effectiveness*. A nested loop approach is used to implement the method. The algorithm examines the candidate subsequences iteratively in an outer loop. For each such candidate subsequence, the k-nearest

neighbors are computed progressively in an inner loop with distance computations to other subsequences. Each candidate subsequence is either included in the current set of best n outlier estimates at the end of an outer loop iteration, or discarded via *early abandonment* of the inner loop without computing the *exact* value of the k-nearest neighbor. This inner loop can be terminated early, when the currently approximated k-nearest neighbor distance for that candidate subsequence is less than the score for the nth best outlier found so far. Clearly, such a subsequence cannot be an outlier. In order to obtain the best pruning results, the subsequences need to be heuristically ordered, so that the earliest candidate subsequences examined in the outer loop have the greatest tendency to be outliers. Furthermore, the pruning performance is also most effective when the subsequences are ordered in the inner loop such that the k-nearest neighbors of the candidate subsequence are found early. It remains to explain how the heuristic orderings required for good pruning are achieved.

Pruning is facilitated by an approach which can measure the clustering behavior of the underlying subsequences. Clustering has a well known relationship of complementarity with outlier analysis, and therefore it is useful to examine those subsequences first in the outer loop, which are members of clusters containing very few (or one) members. The SAX representation is used in order to create a simple mapping of the subsequences into clusters. The piecewise aggregate approximations of SAX are performed over windows of length $w < K$. Then, each window for outlier analysis examines a sequence of symbols (or words) of length K/w. Subsequences which map to unique words are more likely to be discordants than those which map to the same word. This is because the latter case corresponds to subsequences which are members of the same cluster. This observation is leveraged in [258] in order to design a pruning mechanism for outlier analysis. In the discrete case, since the number of distinct words is often limited, a trie data structure can be used in order to maintain the counts of the distinct words in the subsequences, and identify the more unusual ones. Furthermore, the identities of the set of actual subsequences to which these series map to are also maintained in this data structure. Subsequences which are anomalous can be identified by words with low frequency count. This provides an *ordering* in which to examine the different candidate subsequences. For each candidate subsequence, those subsequences which map to the same word may be considered first for computing the nearest neighbor distances in order to provide quick and tight upper bounds on the nearest neighbor distances. As these distances are computed one by one, a tighter and tighter upper bound on the nearest neighbor distance

is computed. *A candidate can be pruned, when an upper bound on its nearest neighbor distance is guaranteed to be smaller (i.e. more similar) than the nth best outlier distance found so far.* Therefore, for any given series, it is not necessary to determine its exact nearest neighbor by comparing to all subsequences. Rather, early termination of the inner loop is often possible during the computation of the nearest neighbor distance. This forms the core of the pruning methodology used in [258], and is similar in principle to one of the pruning methodologies used in multidimensional distance-based methods [381].

The SAX representation is used in order to provide a good *ordering* of candidates for outliers, as well the ordering in which to iteratively compute the nearest neighbor distance for a candidate. A good ordering of the candidates ensures that strong outliers are found early, and therefore a tight *lower bound* on the outlier score is obtained early, as the nth best outlier found *so far*. A good ordering for nearest neighbor computation of a candidate ensures that the iterative nearest neighbor computation process reaches a tight *upper bound* on the outlier score of a particular candidate early. The processing of the candidate can be terminated early, when its upper bound distances are less than the lower bounds on the nth best outlier score. The effectiveness of the pruning is dependent on the tightness of the approximations, which in turn is dependent on the quality of the ordering created by SAX-based clustering.

3.3   Single Series versus Multiple Series

The beginning of this section introduced the problem scenario where the subsequence-based time series-based anomalies needed to be detected over a database of multiple time series of the same type. For example, one may have a database of a few thousand ECG time series, from which anomalies need to be found. However, in many cases, a single very long series may be available, and it may be desirable to determine unusual segments of the series from this large series. The case of a single large series is not very different from the case of multiple series, since the former can be transformed into the latter by using window-based techniques.

In practice, this transformation does not need to be performed at all. It is possible to directly extract subsequences from the series (whether single or multiple) using the same approach. The origin of these subsequences is immaterial from a modeling perspective, since the identification of subsequence anomalies compares with all other subsequence segments, whether from the same series or other series. The anomalous series are discovered from this set of extracted series subsequences.

While some methods such as *Hotsax* assume a single long time-series in the original problem definition, others assume multiple series. The (approximate) equivalence of the two definitions ensures that methods for one can be used for the other, and vice-versa.

The presence of a larger number of series makes the modeling process more robust. Furthermore, if some of these series are identified to be definitely normal as part of the problem input, then this corresponds to the semi-supervised scenario. In such cases, the data modeling process should use only the normal series. For example, in a nearest neighbor anomaly detector, a candidate sequence should be compared only to subsequences from the normal series. Note that the case of multiple series of the same type, is different from the *multivariate* scenario, in which the different *types* of time series are available, which are synchronized by their time stamps.

3.4   Finding Unusual Shapes from Multivariate Series

The case of finding unusual shapes from multivariate series is much more challenging. Here different behavioral attributes such as temperature, pressure, or the same behaviorial attribute such as the temperature may be measured *at the same instant* by different sensors. The problem of finding unusual shapes therefore needs to be very carefully defined in this case. As will be evident from the subsequent discussion, this problem maps directly to that of trajectory outlier detection, which will be discussed in detail in Chapter 10.

In multivariate temporal data, the different behavioral attributes are typically measured with the use of multiple sensors simultaneously. An example is the *Intel Research Berkeley Sensor data* described in [357], which measures different behavioral attributes over time. For example, the behavior of one of the temperature and pressure sensors at the same segment of time is illustrated in Figures 8.3(a) and (b) respectively. So how does one determine multivariate time-series of different shapes?

It turns out that existing work on trajectory outlier detection can be leveraged for this scenario. Even though the existing work [292] has been designed for spatial trajectories, it can also be extended to the non-spatial case with arbitrary values on the X-coordinates and Y-coordinates. In this case, it is possible to visualize the variation of the two behaviorial attributes by eliminating the common time attribute, or by creating a 3-dimensional trajectory containing the time and the other two behaviorial attributes. Examples of such trajectories are illustrated in Figures 8.3(c) and (d) respectively. The most generic of these
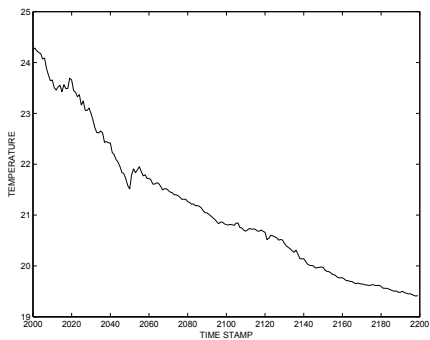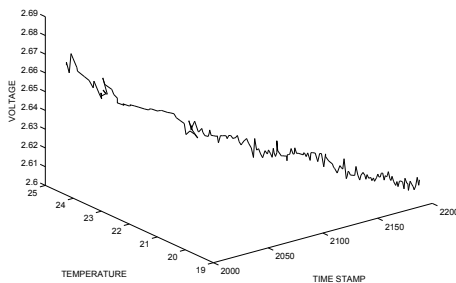
(a) Temperature
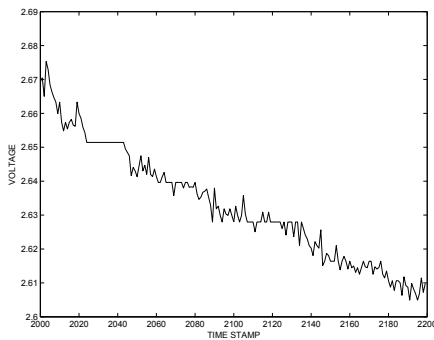
(b) Voltage
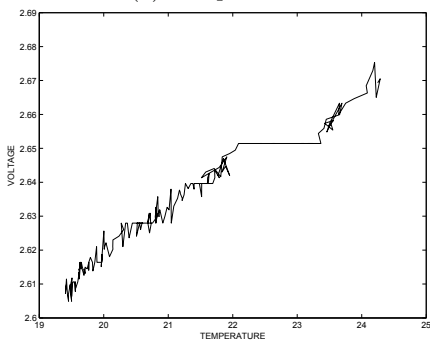
(c) Temperature-Voltage
Trajectory

(d) Time-Temperature-Voltage
Trajectory

Figure 8.3. Multivariate time series can be mapped to trajectory data

trajectories is illustrated in Figure 8.3(d), which shows the simultaneous variation between all three attributes. In general, a multivariate time-series with n behavioral attributes can be mapped to a $(n + 1)$-dimensional trajectory. One issue is that when the number of behaviorial attributes increases, the dimensionality of the corresponding trajectory also increases. This leads to the challenges arising from the curse of dimensionality. In such cases, it may be better to explore subsets of behavioral attributes in order to determine outliers. This corresponds to the subspace methods discussed in Chapter 5. This is an extremely difficult problem because it combines trajectory analysis with multivariate time series analysis, and is still an open problem in the literature.

The TROAD method [292] can be used in order to determine unusual shaped trajectories in such cases. This approach is described in detail in section 6.1 of Chapter 10. While this method was designed for 2-dimensional spatial data, a generalized version of the method can be used for higher dimensional trajectory data containing any kind of attributes. Care needs to be takes to normalize the data along each attribute to unit variance. This is particularly important in the non-spatial scenario, since the different axes of the trajectory may be drawn on very different scales.

3.5 Supervised Methods for Finding Unusual Time-Series Shapes

In many medical applications, characteristic pathological properties may be captured by unusual shapes of the underlying time series. In such cases, training data is often available about either the normal or the pathological behavior or both. Thus, such an approach requires the detection of unusual shapes in time series in a supervised manner. The labels may either be associated with the entire time-series, with portions of the time-series (subsequences). The simplest possible approach in such a scenario is develop subsequence profiles for both the normal class and the anomalous class. For a given test subsequence, a k-nearest neighbor approach may be used for classification purposes. Methods for querying and classifying data streams may be found in [141].

Feature extraction and transformation forms the core of all supervised methods for time series classification. If the time-series is represented in the form of discriminative features, it becomes much easier to classify it. One possibility is to transform the series to a discrete representation, and then use Hidden Markov Models (HMM) of Chapter 9 for the purposes of classification. A much larger number of models are available for sequence classification because of the natural ease in developing pattern-based classification models in the context of discrete data. A

second possibility proposed in [343, 490], is to mine *shapelets* from the data, which are highly discriminative features for classification purposes. While many of these methods have been proposed for generic time series classification, they can usually be generalized to the case where the classes are imbalanced.

4. Outlier Detection in Multidimensional Data Streams

The previous sections discussed the case, where outliers were determined from time-series, based on either deviations from expected values, or unusual shapes. Even when multiple correlated time series are available, each time series was largely processed as a unit for analysis. On the other hand, in the case of multidimensional data, each record contains d-dimensions which form an indivisible unit. Furthermore, in the case of time-series a very high level of temporal continuity is observed in the individual series. This is not necessarily the case for multidimensional data streams, where the temporal continuity is much weaker. For example, in a stream of multidimensional text records, the individual frequency of an attribute in a text record cannot be reliably predicted from its immediately preceding records. On the other hand, the words present in the document can be compared at an *aggregate* level with the history of the stream in order to predict outliers. Thus, outlier analysis in multidimensional data steams is very different from outlier analysis in (possibly multivariate) time series data because of the differences in the expected level of temporal continuity in these scenarios. The multidimensional stream scenario is much more similar to much of the methods discussed in the book for multidimensional outlier analysis. The only difference is the addition of a temporal component to the analysis, though this temporal component is much weaker than in the case of time series data. In the context of multidimensional data streams, efficiency is a core concern, because the outliers need to be discovered quickly. There are two kinds of outliers, which may arise in the context of multidimensional data streams.

- One is based on outlier detection of individual records. For example, a first news story on a specific topic represents an outlier of this type. Such an outlier is also referred to as a *novelty*.

- The second is based on changes in the *aggregate trends* of the multidimensional data. For example, an unusual event such as a terrorist attack may lead to a burst of news stories on a specific topic. This essentially represents a higher level and aggregated outlier based on a specific time window. The second kind of change point

almost always begins with an individual outlier of the first type. However, an individual outlier of the first type may not always develop into an aggregate change point.

Both kinds of outliers (or change points) will be discussed in this section. Furthermore, supervised methods for detecting rare and novel classes from multidimensional data streams will also be discussed in this section.

4.1   Individual Data Points as Outliers

The problem of detecting individual data points as outliers is closely related to the problem of *unsupervised novelty detection*, especially when the entire history of the data stream is used. This problem is studied extensively in the text domain in the context of the problem of *first story detection* [515]. Such novelties are often trend-setters, and may eventually become a part of the normal data. However, when an individual record is declared an outlier in the context of a *window* of data points, this may not necessarily correspond to a novelty. In this context, proximity-based algorithms are particularly easy to generalize to the incremental scenario, by an almost direct applications of the algorithm to the window of data points. Numerous variations of proximity-based algorithms have been generalized to the temporal scenario in the literature.

4.1.1   Proximity-based Algorithms.   A distance-based method to detect such outliers was proposed in [48]. The original distance-based definition of outliers is modified in the following way:
The outlier score of a data point is defined in terms of its k-nearest neighbor distance to data points in a time window of length W.
Note that this is a relatively straightforward modification of the original distance-based definition. The work in [48] proposes the STORM algorithm for distance-based outlier detection. When the entire window of data points can be maintained in main memory, it is fairly easy to determine the outliers. On the other hand, in many interesting cases, it may not be possible to hold the entire window of data points in main memory. In such cases, the streaming scenario is much more challenging, because it is difficult to create efficient indexes for distance-based pruning. For this case, approximate outliers are returned, because the exact determination of outliers is too expensive. Accuracy guarantees are provided for the outlier detection process.

The LOF algorithm has also been extended to the incremental scenario [369]. Furthermore, the approach can handle both insertion and deletion of data points. Two steps are performed in the insertion process:

- The statistics of the newly inserted data point are computed with respect to the existing LOF model.

- The existing LOF model is updated in terms of densities, reachability distances, and the outlierness of the underlying data points. In other words, the parameters of many of the existing data points need to be updated, because they are affected by the addition of a new data point. However, not all points need to be updated, because only the locality of the new data point is affected. The work in [369] performs these updates in a judicious way, so as to efficiently determine the outliers.

Since distance-based methods are well known to be computationally expensive, many of the aforementioned methods are still quite expensive in the context of the data stream. Therefore, the complexity of the outlier detection process can be greatly improved by using a clustering-based approach. While clustering-based methods are generally not advisable, when the number of data points is limited, this is not the case in streaming analysis. In the context of a data stream, a sufficient number of data points are typically available in order to maintain the clusters at a very high level of granularity. *In the context of a streaming clustering algorithm, the formation of new clusters is often associated with unsupervised novelties.* Of course, this may not always be the case, when the entire history of the stream is not reflected in the limited-space summary provided by the current set of clusters. For example, the work in [25] explicitly regulates the creation of new clusters in the data stream, when an incoming data point does not lie within a specified statistical radius of the existing clusters in the data. Such data points may be considered outliers. In many cases, this is the beginning of a new trend, as more data points are added to the cluster at later stages of the algorithm. In some cases, such data points may correspond to novelties, and in other cases, they may correspond to trends which were seen a long time back, but are no longer reflected in the clusters. In either case, such data points are interesting outliers. However, it is not possible to distinguish between these different kinds of outliers, unless one is willing to allow the number of clusters in the stream to increase over time. The work in [266] leverages the clustering process discussed in [25] in order to improve the efficiency of the outlier analysis process.

This approach has also been generalized to the case of text data [26]. In these cases, it can be used in order to detect the first story [515] from a possibly new trend of more stories on the topic. Other distance-based algorithms for first story detection from text streams are discussed in section 6.2 of Chapter 7. Therefore, the reader is referred to Chapter

7 for a discussion on how such proximity-based methods for applied to first-story detection in text streams. Such methods can also be combined with window-based strategies in order to detect outliers by performing the clustering on specific chunks of the data stream [150].

4.1.2   Probabilistic Algorithms.

The clustering approach discussed above can also be used in the context of probabilistic learning algorithms. As discussed in Chapter 2, probabilistic learning algorithms are generally not advisable in the context of limited data. However, in the context of data streams, a much larger amount of data is available for learning, and this is less of an issue, as long as the learning algorithm can be implemented *efficiently.* A method in [478] proposes methods for creating mixture models from mixed attribute data sets. The idea is to compute a fit of the incoming data point to the model both before and after the data point is added to the model. In addition, such methods have also been extended to first story detection in text streams [503]. These approaches is discussed in detail in Chapter 7 of this book, and the reader is referred to that chapter.

4.1.3   High-dimensional Scenario.

The combination of the stream scenario and high dimensional data is particularly challenging because of the complexity of high dimensional projected clustering algorithms. Clearly, computational efficiency is a primary concern in the context of high-dimensional algorithms.

Many of the high dimensional projected stream clustering algorithms can also be used in order to determine anomalies in the data stream. This is because many of these algorithms report outliers as side products of the clustering algorithm [6, 111]. Data points which start new clusters in the stream can typically be reported as outliers.

A method called SPOT [498] is designed for detecting outliers in high-dimensional data streams. Unlike clustering methods, this method is designed to directly find sparse subspaces of the data. This approach uses a decaying cell based summary of the underlying data stream. This is then used in order to determine projected cell based summaries. The sparse subspaces in the underlying data stream. The data points which lie in the cells corresponding to the sparse subspaces are reported as outliers.

4.2   Aggregate Change Points as Outliers

Numerous methods have been proposed in the literature in order to determine significant change points in a multidimensional data stream. Many change point detection techniques have been studied indepen-

dently in the database literature and the outlier detection literature. The reality is that these two areas are too closely related to be treated separately. The sudden changes in aggregate local and global trends in the underlying data are often indicative of anomalous events in the data. Many methods also provide statistical ways of quantifying the level of the changes in the underlying data stream. Therefore, we will discuss some of the significant methods for change detection, which can also be used for outlier detection. Some of these methods have also been used in the anomaly detection literature for finding significant changes in stock order data streams [313], or for finding anomalies in network data streams [162, 280, 281].

4.2.1 Velocity Density Estimation Method.

The idea in velocity density [16] 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 [409], we provide a continuous estimate of the density of the data at a given point. The value of the density at a given point is estimated as the sum of the smoothed values of kernel functions $K_h'(\cdot)$ associated with each point in the data set. Each kernel function is associated with a kernel width h which determines the level of smoothing created by the function. The kernel estimation $\overline{f}(\overline{X})$ 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. It has been shown [409] that for most smooth functions $K_h'(\cdot)$, when the number of data points goes to infinity, the estimator $\overline{f}(x)$ asymptotically converges to the true density function $f(x)$, provided that the width h is chosen appropriately. For the d-dimensional case, the kernel function is chosen to be the product of d identical kernels $K_i(\cdot)$, each with its own smoothing parameter h_i.

In order to compute the velocity density, a temporal window h_t was used in order to perform the calculations. Intuitively, the temporal window h_t is associated with the time horizon over which the rate of change

is measured. Thus, if h_t is chosen to be large, then the velocity density estimation technique provides long term trends, whereas if h_t is chosen to be small then the trends are relatively short term. This provides the user flexibility in analyzing the changes in the data over different kinds of time horizons. In addition, a spatial smoothing vector h_s is used, whose function is quite similar to the standard spatial smoothing vector which is used in kernel density estimation.

Let t be the current instant and S be the set of data points 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 [16] 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 [409]. 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 multidimensional data distributions into a quantification, which can be used in conjunction with extreme-value analysis.

It is important to note that the data point $\overline{X}$ 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 [16]. Time instants with large values of the aggregate velocity density may be declared outliers.

4.2.2 Statistically Significant Changes in Aggregate Distributions.

A different way to characterizing aggregate changes in multidimensional data streams would be to estimate the aggregate distributions in these time windows. Significant changes in these time windows can be reported as the unusual changes in the data stream. We note that the velocity-density method also estimates the aggregate distributions with the use of kernel-density estimation. However, in the context of change detection, it is also sometimes useful to be able to perform statistical tests directly on the underlying distributions in order to determine significant change points.

The work in [260] proposes a non-parametric framework for change detection in data streams. The key contribution in the work is a definition of the distance between two probability distributions. Generalizations of the Wilcoxon and Kolmogorov-Smirnoff tests are used in order to determine the significant change points. The work is however proposed for the case of 1-dimensional data streams, and the generalization to higher dimensions is not discussed.

The work in [131] is a bit more general, in that it can address multidimensional data streams effectively. Furthermore, it can be applied to different data types, because the computational aspects of the problem are different from the type of the underlying data. Since change detection is essentially relevant to the concept of finding distances between distributions, a very general way of representing this distance is the relative entropy from information theory. This is also known as the Kullback-Leibler (or KL) distance.

The broad principle is as follows. Given a set of data that should be fit to a family of distributions, the maximum likelihood estimator is the one that minimizes the KL-distance to the true distribution. This is a very general form of many other kinds of change analysis. For example, the *t*-test is equivalent to the KL-distance between two normal distributions. The KL-distance also allows for better intuitive interpretability, and is therefore particularly appealing for outlier detection. The definition of the KL-distance is also independent of the specific dimensionality or the type of the data representation itself. Therefore, this abstracts out the

modeling of the change from the computational process. Details of this approach are presented in [131].

4.3 Rare and Novel Class Detection in Multidimensional Data Streams

An interesting case is when supervision is available in order to guide the temporal outlier detection process. In such cases, class labels are attached to the individual data points. Many different kinds of supervised outliers may be of interest in such scenarios. The supervised scenario is a a very rich one in the temporal domain, because *different kinds of temporal and frequency-based aspects of the classes could correspond to outliers.* These could correspond to novel class outliers, rare class outliers, or infrequently recurring class outliers. Thus, a combination of methods for concept drift analysis, outlier detection, and rare class detection may need to be used in order to determine interesting outliers in the streaming scenario. Such scenarios could arise quite often in applications such as intrusion detection, in which some known intrusions may be labeled, but new intrusions may also arise over time. Therefore, it is critical for the anomaly detection algorithm to use a combination of supervised and unsupervised methods in order to detect outliers.

The different kinds of outliers which can be defined in such scenarios are as follows:

- *Rare Class Outliers:* The determination of such classes is similar to the static supervised scenario, except that it needs to be done efficiently in the streaming scenario. In such cases, a small fraction of the records may belong to a rare class, but they may not necessarily be distributed in a non-homogenous way from a temporal perspective. While some concept drift may also need to be accounted for, the modification to standard stream classification algorithms remains quite analogous to the modifications of the static classification problem to the rare-class scenario.

- *Novel Class Outliers:* These are classes, which were not encountered before in the data stream. Therefore, they may not be reflected in the training model at all. Eventually, such classes may become a normal part of the data over time. This scenario is somewhat similar to semi-supervised outlier detection in the static scenario, though the addition of the temporal component brings a number of challenges associated with it.

- *Infrequently Recurring Class Outliers:* These are classes, which have not been encountered for a while, but may re-appear in the

stream. Such classes are different from the first type of outliers, because they arrive in *temporally rare bursts.* Since most data stream classification algorithms use some form of discounting in order to address concept drift, they may sometimes completely age out information about old classes. Such classes cannot be distinguished from novel classes, if the infrequently recurring classes are not reflected in the training model. Therefore, issues of model update and discounting are important in the detection of such classes. The third kind of outlier was first proposed in [328].

We discuss each of the above cases below. Since some of the issues of class imbalance also arise in static data, this aspect is also covered in detail in Chapter 6.

4.3.1   Detecting Rare Classes.   Numerous classifiers are available for the streaming scenario [10], especially in the presence of concept drift. For detecting *rare classes*, the only change to be made to these classifiers is to add methods which are intended to handle the *class imbalance.* Such changes are not very different from those of addressing class imbalance in the static context. Therefore, the reader is referred to Chapter 6 for a detailed description of methods which are used for this case. The reader is also referred to the bibliography section of the current chapter for references to research which have generalized the methods of Chapter 6 to the streaming scenario. Since the broad principles of detecting rare classes do not change very much between the static and dynamic scenario, the discussion in this section will focus on the other two kinds of outliers.

4.3.2   Detecting Novel Classes.   The problem of detecting novel classes is also discussed in Chapter 6. Chapter 6 discusses the problem of novel class detection in a general setting which could either be online or offline. In this chapter, the problem of novel class detection will be studied in the online setting, which is the most common scenario in which it is encountered. Detecting novel classes is also a form of *semi-supervision*, because models are available about many of the other classes, but not the class that is being detected. This is different from many unsupervised settings such as the first story detection setting [26, 503, 515] discussed in Chapter 7, and the streaming outlier detection setting [25] in which no labels are available at all. In the latter case, the models are created in an unsupervised way with the use of clustering or other unsupervised methods. The work in [328, 329] addresses the problem of novel class detection in the streaming scenario.

Much of the traditional work on novel class detection [325] is focussed only on finding novel classes which are different from the current models. However, this approach does not distinguish between the different novel classes which may be encountered over time. A more general way of understanding the novel class detection problem is to view it as a combination of supervised (classification) and unsupervised (clustering) models. Thus, as in unsupervised novel class detection models such as first story detection [26, 515], the cohesion between the test instances of a novel class is important in determining whether they belong to the same novel class or not. The work in [328, 329] combines both supervised and semi-supervised models by:

- Maintaining a supervised model of the classes available in the training data as an ensemble of classification models.

- Maintaining an unsupervised model of the (unlabeled) novel classes received so far as cohesive groups of tightly knit clusters.

When a new test instance is received, the classification model is first applied to it to test whether it belongs to a currently existing (labeled) class. If this is not the case, it is tested whether it naturally belongs to one of the novel classes. The relationship of the test instance to a statistical boundary of the clusters representing the novel classes is used for this purpose. If neither of these conditions hold, it is assumed that the new data point should be in a novel class of its own. Thus, this approach creates a flexible scenario which combines supervised and unsupervised methods for novel class detection.

4.3.3 Detecting Infrequently Recurring Classes.
Many outliers in real applications often arrive in *infrequent temporal bursts*. Many classifiers cannot distinguish between novel classes and rare classes, especially if the old models have aged out in the data stream. Therefore, one solution is to simply report a recurring outlier as a novel outlier.

This is however not a very desirable solution because novel-class detection is a semi-supervised problem, whereas the detection of recurring classes is a fully supervised problem. Therefore, by remembering the distribution of the recurring class over time, it is possible to improve the classification accuracy. The second issue is related to computational and resource efficiency. Since novel class detection is much more computationally and memory intensive than the problem of identifying an existing class, it is inefficient to treat recurring classes as novel classes. The work in [328] is able to identify the recurring classes in the data, and is also able to distinguish between the different kinds of recurring classes, when they are distinct from one another by examining their

relationships in the feature space. For example, two kinds of recurring intrusions in a network intrusion detection application may form distinct clusters in the stream. The work in [328] is able to distinguish between the different kinds of recurring classes as well.

5.   Conclusions and Summary

This chapter studies the problem of outlier detection in streaming time series data and multidimensional data streams. The nature of the outliers are very different in the two cases, since time-series data requires the analysis of each series as a unit, whereas the multidimensional data requires the analysis of each multidimensional point as a unit. Different kinds of outliers can also be defined in time series data, depending upon whether it is desirable to determine unusual deviation points in the series, or whether it is desirable to determine unusual shape subsequences in the time series. For the case of multidimensional data, individual data points can be classified as novelties, or aggregate change points in the data may be defined as outliers. Thus, the area of temporal outlier detection provides a wide variety of different problem definitions.

6.   Bibliographic Survey

Outlier detection has been studied extensively in the context of traditional time-series data analysis, especially from the perspective of removing noise from the underlying data [82, 109, 110, 168, 387]. Much of this work such as Kalman Filtering [82] has focussed on the removal of noise from and smoothing of temporal time series in order to facilitate more accurate regression and prediction. Outliers are defined as deviations from predicted values in such cases. Nevertheless, in such cases, the deviations may not necessarily represent noise, but may represent more significant anomalies in the underlying data generation process.

Significant changes in the time series trends can be detected as changes and outliers in the data [479]. This includes many traditional methods for regression modeling such as Autoregressive Modeling (AR), Autoregressive Moving Average (ARMA) and Autoregressive Integrated Moving Average (ARIMA). Methods such as principal component analysis [244] have also been commonly used in order to track correlations among multiple streams. This allows for more robust detection of outliers, since the prediction process is more robust as well. Many methods have also been designed to speed up the regression modeling in the context of large number of data streams and real time data [24, 227, 240, 357, 491]. An information-theoretic method for determining anomalous points in a time-series was proposed in [233], where an outlier was defined as a point

in a time-series, whose removal results in a better histogram representation in the same storage. The storage in the two cases is compared after explicitly accounting for the space required by the outlier point. This is essentially an information theoretic approach. This method is however not designed for online outlier detection on the basis of *past* window of history. Supervised methods can also be used in order to interpret the abnormal deviations of the underlying streams in a supervised way, so that specific kinds of abnormal events can be detected by discriminative analysis between different kinds of deviations [9].

While time series outlier detection is often understood in terms of real time change detection and deviations, the problem of finding unusual shapes in time series is an entirely different scenario. While the former is related to extreme value analysis, the determination of the latter is more subtle, since it requires a careful analysis of the regularities in the series over different windows of the data. One of the earliest methods for anomaly detection in time series using ideas from immunology was proposed in [129]. In such cases, the overall magnitude of the deviations matters less than the shape of the overall time-series.

A variety of methods can be used in order to compute unusual shapes. The use of the Haar transform for anomaly detection in time series has been explored in [171]. The most common method [258] is the use of the Euclidian distance on the fixed windows of the time series. It has been shown in [258, 304] that this problem can be further sped up with the use of symbolic aggregate approximation. These methods have been scaled up to be disk-aware, and work with terabyte scale data sets in [487]. Methods for anomaly detection in multivariate time-series are discussed in [59, 115]. Another interesting approach proposed in [61] was to determine anomalous *regimes* in time-series data, where the correlations and dependencies among the different time series has changed over time. Discrete transformations can also be used on a stand-alone basis [257] in order to perform the anomaly detection directly on the discrete representation, though this is rarely done because of the information loss associated with discretization.

Methods for finding anomalies in discrete sequences are discussed in Chapter 9. A detailed survey for the discrete case may be found in [108]. Some semi-supervised and supervised methods have also been proposed for unusual shape detection in the literature. In the semi-supervised scenario, it is assumed that examples of normal time series are available. In the fully supervised scenario, examples of both the normal data and the anomalous shapes are available [141, 237, 343, 474, 490]. The transformation to the discrete case also enables the use of supervised string-based models such as Hidden Markov Models (HMM) [108]. Other

transformation methods include feature transformation methods, which construct relevant *shapelets* on the underlying time series [343, 490]. These are patterns which can discriminate between different classes of instances. Another well known method is the use of distance function such as dynamic time-warping [237]. The design of effective distance functions is enables the development of proximity-based classification techniques. While most of the aforementioned techniques have been developed in the literature for the *generic* version of the classification problem, most of these methods can be easily adapted to the rare class scenario by using methods discussed in Chapter 6.

Multivariate temporal data can also be represented as trajectories. Methods for trajectory outlier detection are discussed in the next chapter. Significant *changes* in trajectory directions is useful for many applications such as hurricane tracking [94]. In such cases, the trajectory can be treated as bivariate temporal data, and the prediction-based deviation detection techniques can be applied to this representation. The works in [83, 181] determine anomalies in moving object streams in real time, by examining patterns of evolution. On the other hand, the detection of anomalous trajectory *shapes* is a very different problem. The earliest methods for trajectory shape outlier detection were proposed in [263]. However, this method transforms the trajectories into point data by using a set of features describing meta-information about the trajectories. Unsupervised methods for trajectory outlier detection, which actually use the sequence information explicitly were first investigated in [344, 292]. The work in [344] uses the fourier transform in order to represent the trajectories in terms of the leading coefficients, and find anomalies. In the second method [292], trajectories are divided into different line segments and anomalous patterns are identified in order to determine outliers. Supervised methods for anomaly detection in trajectory data may be found in [300]. These methods transform the data into discrete sequences, and a classifier is learned in order to relate the trajectories to the class labels.

Outlier detection has also been studied extensively in the context of sensor data [3, 86, 76, 170, 427, 502]. Sensor streams are one of the most common applications of anomaly detection in temporal data [502]. Sensor data is also noisy because of errors and deviations in the data collection and transmission process. Therefore, the outlier detection problem has dual applicability to this scenario, both in terms of removing the underlying noise, and in terms of detecting unusual events from the sensor stream. The same techniques are typically used for both cases. In this context, supervision [9] can sometimes be useful in distinguishing between noise and anomalies. A number of unique technological issues

arise in the context of outlier detection in sensor data, because of the large amounts of data involved. Consequently, *real-time* and *in-network* processing is important in order to minimize delays and communication costs. A detailed survey on outlier detection in sensor networks may be found in [502].

The problem of outlier detection in multidimensional data streams has also been studied extensively in the literature [25, 26, 48, 50, 347, 369, 503] in the context of different kinds of multidimensional and text data. In all these methods the goal is to determine unsupervised novelties from the underlying data. Efficiency is particularly important in these cases because of the stream scenario. The problem of novelty detection is also closely related to clustering in the stream scenario, since the formation of new clusters corresponds to novelties in the data. Many of the aforementioned techniques use either clustering or distance-based methods to detect novelties. For example, the work in [266] uses stream clustering algorithms [25] in order to improve the efficiency of the outlier analysis process. A method in [347] uses data editing techniques, which progressively remove data points with the smallest nearest neighbor distance. This is referred to as *numerosity reduction*, which reduces the size of the data, while retaining certain desirable properties for outlier detection. Recently, streaming methods for outlier detection have also been extended to the case of probabilistic data [460].

Another form of outliers in multidimensional outlier detection is to determine aggregate changes in data streams [160]. This requires the determination of whether the *aggregate distribution* of the multidimensional data has changed enough to be considered significant. Methods for characterizing the change in the form of velocity density contours may be found in [16]. These methods can also be generalized to the high dimensional case. Other methods which uses different forms of statistical testing for change detection such as the KL-distance, Wilcoxon, and the Kolmogorov-Smirnoff test, may be found in [131, 260]. A number of other statistical methods [279, 417] use log-likelihood criteria in order to quantify the change. A martingale framework for change detection in data streams is proposed in [217]. Such change points can be very useful in determining anomalies in stock market order distributions [313] or temporal network traffic data streams [162, 280, 281]. The latter class of methods can sometimes also be used to diagnose network intrusions. Methods for aggregate change point detection in the form of correlated bursty topic patterns in coordinated text streams are proposed in [458].

The effectiveness of novelty detection can be enhanced by using class labels, which can identify some of the classes (either normal or abnormal) in the data. Thus, this scenario can be considered a semi-supervised one,

since information about a subset of the classes is available. A detailed survey of traditional methods for semi-supervised novelty detection may be found in [325, 326]. Methods for using support vector regression models for online novelty detection are discussed in [318]. Consequently, methods need to be designed which combine streaming classification models with streaming clustering models in order to distinguish normal classes, novel classes, rare classes and rarely recurring classes [328, 329, 36].

7.   Exercises

1. Develop a closed form solution to the auto-regressive model (AR) in this chapter using the relationship discussed in Chapter 3. What is the size of the matrix which needs to be inverted in order to solve this model?

2. Develop a solution to the ARMA model for deviation detection. How does the complexity compare to the simple AR model?

3. Apply each of the algorithms in Exercises 1 and 2 to detect point (time-stamp) outliers in the *Ozone Level Detection* data set of the UCI Machine Learning Repository [169], for each attribute (series) separately. How do the outliers compare for different attributes in the two cases?

4. Apply the unusual shape detection algorithm discussed in this chapter in order to detect unusual shape subsequences from the *Ozone level detection* data set of the UCI Machine Learning Repository [169]. Do the unusual shapes occur at time stamps which are in any way related to the time-stamps of unusual point deviations found in Exercises 2 and 3?

5. Apply a multivariate PCA technique across the different attributes of the *Ozone Level Data Set*, where each time-stamp is treated as a multi-attribute data point corresponding to the values over different series. Note that the time-stamps are treated independently of one another and temporal continuity is not used. Determine points of significant deviations from the regression model. Which time-stamps are the outliers?

6. Repeat Exercise 5 using windows of length p in order to generate each data point. Thus, a data point of dimensionality $p * d$ is generated at each time-stamp for multivariate series with d attributes.

7. Use the methodology of Exercise 5, in order to create a multidimensional data set from the time-series data set. Determine outliers with:

  ■ A k-nearest neighbor algorithm over the entire data set.
  ■ A k-nearest neighbor algorithm over the segment of the data set containing only earlier time stamps?

  How do the outliers found relate to each other, and to those found in Exercise 5?

8. Repeat Exercise 7 using windows of length p according to the approach in Exercise 6 rather than Exercise 5.

Chapter 9

OUTLIER DETECTION IN DISCRETE SEQUENCES

"Poets write the words you have heard before, but in a new sequence." – Brian Harris

1.   Introduction

In the previous chapter, the discussion of anomaly detection was focussed on continuous time series. A related problem is the case where the sequences are discrete. In other words, the values at the time stamps are categorical. Such scenarios arise quite commonly in a variety of system diagnosis, intrusion detection and biological data applications. It is to be noted that in some domains such as intrusion detection and system diagnosis, the discrete sequences are caused by *temporal ordering*, whereas in other domains such as biological data, the discrete sequences are caused by *physical ordering*. Nevertheless, at a logical level, the difference in the problem definition for the two cases is relatively small. The main difference is that temporal data often has a specific direction to the analysis in real scenarios (i.e. forward in time), whereas this may not be the case for data based on placement relationships. At the analytical level, the models for the two cases are different in minimal ways, and typically have cross-applicability. Some examples of applications which may generate discrete data sequences are as follows:

- *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.

- *Biological Data;* Biological data typically contains sequences of amino-acids, in which anomalous subsequences can provide interesting information about unusual properties of the genome sequences and their impact on different kinds of genetic conditions [308].

- *User-action Sequences:* A variety of sequences abound in daily life, which are created by user actions in different domains.

  - Web logs contain long sequences of visits to web sites by different individuals. It is desirable to determine interesting subsequences which are indicative of anomalous or intrusive activity.

  - Customer transactions may contain sequences of buying behavior. The symbols may correspond to the identifiers of the different items which are bought. Unusual temporal patterns may correspond to changes in buying patterns [96]. This information may often be useful for a merchant in order to determine key changes in the trends.

  - User actions on web sites such as online banking sites are frequently logged. This is quite similar to the case of web logs above, except that the logs of banking sites often contain much more detailed information for security purposes. Anomalous subsequences of actions provide insights into unusual user activity, such as an attempt to break into the system.

In order to ease further discussion, some notations will be defined. A sequence is defined as an ordered set of symbols $a_1 a_2 \ldots a_r$ drawn from the symbol set $\Sigma = \{\sigma_1 \ldots \sigma_{|\Sigma|}\}$. In the most general form, a sequence can also be defined as an ordered set of sets $S_1 S_2 \ldots S_r$, where each S_i is a subset of Σ. In this chapter, the more common (and simpler) case will be examined in greater detail, where each element a_i is drawn directly from Σ. The more complex case of set-based sequences will also be examined briefly.

Discrete sequences are different from continuous time-series data, because it is difficult to directly compare two different symbols from the alphabet Σ in terms of distances. Therefore, commonly used regression-modeling methods for deviation detection in continuous data are not easily generalizable to this case. Nevertheless, even in this case, outliers can be defined in terms of either deviations from predicted values at specific time stamps, or in terms of unusual *successive combinations of sequence values.* The key is to define an appropriate predictive or

regularity model in the discrete case, which is analogous to its continuous counterpart. As in the case of continuous data, outliers are of two types, depending upon whether *specific positions* are considered outliers, or whether *combinations of symbols* are considered outliers.

- *Position Outliers:* In position-based outliers, the values at specific positions are predicted by a model. This is used in order to determine the *deviation* from the model, and predict specific *positions* as outliers. Typically, Markovian methods are used for predictive outlier detection. This is analogous to deviation-based outliers which are discovered in time-series data with the use of regression models. Unlike regression models, Markovian models are better suited to discrete data. These correspond to *contextual* outliers.

- *Combination Outliers:* In combination outliers, an entire test sequence is deemed to be unusual because of the combination of symbols in it. This is because this combination may rarely occur in a sequence data base (frequency-based), or its distance (similarity) to most other subsequences of similar size may be very large (small). More complex models such as Hidden Markov Models can also be used to model the frequency of presence in terms of generative probabilities. For a longer test sequence, smaller subsequences are extracted from it for testing, and then the outlier score of the entire sequence is predicted as a combination of these values. This is analogous to the determination of unusual shapes in time-series data. These correspond to *collective* outliers.

The rarity in the combination of values can be defined in different ways depending upon the specific regularity model used for outlier analysis. The different models, which are commonly used are as follows:

- *Distance-based:* In this case, the nearest neighbor distance of a subsequence to other candidate subsequences in the base data is computed. This provides an outlier score for the subsequence. The score of a longer sequence can be computed by combining the outlier scores of its smaller subsequences.

  For the case of shorter sequences, methods such as clustering can be used for the outlier analysis process. Such methods are analogous to distance-based techniques for outlier analysis in multidimensional data.

- *Frequency-based:* In this case, the frequency of the occurrence of different subsequences of values is compared between a test instance and the base training data. Sequences, which show unusual

differences in frequency distributions of subsequences between the test and training sequence, are considered outliers. Such methods are useful in cases where the compared subsequence is relatively small, and the alphabet size is small. In such cases, a limited number of distinct values of the subsequence are possible, and it is meaningful to talk in terms of frequencies. As in the case of distance-based models, smaller subsequences are extracted for meaningful frequency comparisons.

- *Model-based:* In this case, a probabilistic generative model is constructed, which generates the subsequences. Typical examples of generative models which are commonly used are *Hidden Markov Models*. Subsequences which have low probability of being generated by the model are considered outliers. Such methods are analogous to EM-algorithms for outlier detection in multidimensional data. The advantage of such models in the discrete case is that a well designed generative Markov Model can encode both the user understanding of the sequences, and can also capture highly likely sequences which are not explicitly present in the data.

In addition, supervision can be used in order to perform the outlier analysis, when training data is available about previous anomalies. This requires the design of rare class adaptations of sequence classification problems.

This chapter is organized as follows. Section 2 discusses predictive models for outlier detection of individual positions in sequences. Section 3 discusses methods for determining combination outliers in discrete sequences. Complex sequences correspond to set-valued symbols, and multivariate sequences. The detection of outliers in such sequences is discussed in section 4. Online and early outlier detection is also discussed in this section. Methods for supervised outlier detection are discussed in section 5. The conclusions and summary are presented in section 6.

2.   Position Outliers

In the case of continuous time-series data discussed in Chapter 8, an important class of outliers was designed by determining significant deviations from *expected* values at time stamps. Thus, these methods intimately combine the problems of *forecasting* and *deviation-detection*. As was discussed in Chapter 8, a variety of regression models are utilized for the purpose of forecasting in continuous data. A similar principle applies to the case of discrete sequence data, in which the discrete positions at specific time-stamps can be predicted with the use of different models. When a position has very low probability of matching its forecasted

value, it is considered an outlier. For example, consider an RFID application, in which event sequences are associated with product items in a superstore with the use of semantic extraction from RFID tags. A typical example of a normal event sequence may appear as follows:

> `PlacedOnShelf, RemovedFromShelf, CheckOut, ExitStore.`

On the other hand, in a shop-lifting scenario, the event sequence may be *unusually* different. An example of event sequence in the case of the shop-lifting scenario is as follows:

> `PlacedOnShelf, RemovedFromShelf, ExitStore.`

Clearly, the sequence symbol *ExitStore* is anomalous in the second case, but not in the first case, because it does not depict the *expected* or *forecasted* value for that position. It is desirable to detect such anomalous *positions* on the basis of expected values. Such anomalous positions may appear *anywhere* in the sequence, and not necessarily in the last element, as in the case of the previous example. The basic problem definition for position outlier detection is as follows:

DEFINITION 9.1 (SEMI-SUPERVISED) *Given a set of training sequences* $\mathcal{D} = T_1 \ldots T_N$, *and a test sequence* $V = a_1 \ldots a_n$, *determine if the position* a_i *in the test sequence should be considered an anomaly based on its expected value.*

Some formulations do not explicitly distinguish between training and test sequences. This is because a sequence can itself be used for both model construction and outlier analysis, especially when it is very long. This is analogous to the case of unsupervised models in which training and test data are not differentiated. For example, such an unsupervised formulation may be as follows:

DEFINITION 9.2 (UNSUPERVISED) *Given a long sequence* $V = a_1 \ldots a_n$, *determine if the position* a_i *in the test sequence should be considered an anomaly based on its expected value.*

The differences between these two models are relatively small, since the latter model simply needs to use the sequence V in order to construct the model. For ease in discussion, the first definition will be used in this chapter.

Typically, the position a_i can be predicted in temporal domains only from the positions before a_i, whereas in other domains such as biological data, both directions may be relevant. The discussion below will assume the temporal scenario, though generalization to the placement scenario

(as in biological data) is straightforward, by examining windows on both sides of the position.

In its more general form, one would like to determine all positions which should be considered anomalies. Note that in many applications such as web logs (for anomalous web page access prediction), the training sequences may appear in the form of a single long sequence. Furthermore, the training sequence and the test sequence may not be cleanly separately from one another, as it may be desirable to use the past history of accesses in order to predict anomalous positions at any point in time. Nevertheless, the models discussed in this chapter are fairly general, and are applicable to these variations with small modifications. The main modification is in terms of properly recognizing the part of the training sequence which should be used with each model.

Just as regression modeling of continuous streams uses small windows of past history, discrete sequence prediction also uses small windows of the symbols. It is assumed that the prediction of the values at a position depends upon this short history. This is known as the *short memory property* of discrete sequences, which generally holds true across a wide variety of temporal application domains [386].

DEFINITION 9.3 (SHORT MEMORY PROPERTY) *For a sequence of symbols $V = a_1 \ldots a_i \ldots$, the value of the probability $P(a_i | a_1 \ldots a_{i-1})$ is well approximated by $P(a_i | a_{i-k} \ldots a_{i-1})$ for some small value of k.*

Another observation is that the value of a_{i-j} influences a_i more, when the value of j is smaller. This is intuitively obvious, since more recent sequence symbols are likely to be more relevant for predicting the current symbol. Once the value of $P(a_n | a_{n-k} \ldots a_{n-1})$ is estimated, a position can be flagged as an outlier, if the symbol which actually appears in a test sequence has very low probability on the basis of the models derived from the training sequences. Alternatively, if a different symbol (than is present in the test sequence) is predicted with very high probability, then that position can be flagged as an outlier.

This section will discuss two common methods, known as *Rule-based* models and *Markovian* models which are used for the position outlier detection problem. Both Markovian models and rule-based models can be considered technically equivalent, which exploit the *small memory* property of sequences in order to explicitly model the sequences as a set of states in a Markov Chain. When the size of the history used for prediction is large, the number states in the Markovian model increases to potentially $|\Sigma|^k$. Rule-based models can be used to provide a pruned and incomplete heuristic description of the transition behavior implied by the Markovian model. This provides a simpler framework for under-

standing the behavior of the sequences. Thus, rule-based models can be considered heuristic simplifications of more complete Markovian models.

2.1 Rule-based Models

The primary goal in rule-based models is to estimate the value of $P(a_i|a_{i-k} \ldots a_{i-1})$ from the training database of sequences $\mathcal{D}$. This can be expressed in the form of the following rule:

$$a_{i-k} \ldots a_{i-1} \Rightarrow a_i$$

The probability of the prediction is typically quantified by the *confidence* of the rule. The rules may be generated either in a lazy way, which is specific to a particular test sequences, or they may be generated using pre-processing on the training data. Then, the appropriate rules can be used for predicting position outliers from the sequence data.

The key challenge in this process is to perform *robust* estimation of the rules from the training data. The sparsity of the data creates a challenge, because even for small window sizes (value of k), the number of possibilities for the discrete sequence may be large. Therefore, the frequency of occurrence of *a particular* sequence $a_{i-k} \ldots a_{i-1}$ in a modestly sized training data set may be small. When there are only zero, one or two occurrences of this sequence $a_{i-k} \ldots a_{i-1}$ in the training data, the corresponding probability values cannot be estimated accurately. Therefore, for a particular antecedent subsequence in the test sequence, it may not be possible to estimate the probability of the next forecasted symbol in a robust way. In order to address this issue, a number of heuristic relaxations and variations of the basic probability estimation approach can be used. This increases the robustness of the rule generation process.

- *Support Criterion:* The rules should not be generated using only confidence criteria. Both support *and* confidence criteria should be used. The *support* of a rule corresponds to the number of times that the antecedent of the rule is present in the training data. Only rules with larger support may be relevant.

- *Variable Antecedent Size:* Not all relevant rules may correspond to a fixed window size. In many cases, the lengths of the antecedents of the rules may vary considerably.

- *"Don't Care" Positions:* Allowing "don't care" positions in the antecedents of the rules allows for the possibility of "noise" symbols, which are not relevant to the predictive process. In this case, a rule may be specified as:

$$a_1 a_2 * *a_5 \Rightarrow a_6$$

Here, the asterisk represents a "don't care" position, which may be noisy and may not have predictive power for the symbol a_6.

While the support-confidence criterion of frequent pattern mining provides a natural framework for rule-generation, many other off-the-shelf methods are available for rule generation. For example, classification rules can be generated by extracting windows from the training sequence, in which the last symbol is treated as a class label. Many off-the-shelf classification rule generators such as *RIPPER* are available for this purpose [120]. An example of such a rule-based model is provided in [294], where rules are generated from the training data. For a given test subsequence, the first rule which is fired is examined. If the right hand side of the fired rule corresponds to a *different* symbol as the value in the test sequence, then this position is flagged as an anomaly. The confidence of the rule (or its inverse) may be used to report the outlier score, depending upon whether it it desired to represent outliers by higher or lower scores.

It is important to understand that the precise design criterion for the rule-generation process depends upon the application at hand. Numerous off-the-shelf rule generators are available for sequence data in the literature. The performance of these different models is likely to vary significantly with the application at hand.

2.2 Markovian Models

These models represent the sequence generation process with the use of transitions in a Markov chain. This is essentially a special kind of *Finite State Automaton*, in which the states are defined by a short (immediately preceding) history of the sequences generated. Such models correspond to a set of states A, which represent the different kinds of memory about the system events. For example, in *first-order* Markov Models, each state represents the last symbol from the alphabet Σ, which was generated in the sequence. In kth order Markov Models, each state corresponds to the subsequence of the last k symbols $a_{n-k} \ldots a_{n-1}$ in the sequence. Each transition in this model represents an event a_n, the transition probability of which from the state $a_{n-k} \ldots a_{n-1}$ to the state $a_{n-k+1} \ldots a_n$ is given by the conditional probability $P(a_n | a_{n-k} \ldots a_{n-1})$. A Markov Model can be depicted as a set of nodes representing the states, and a set of edges representing the events, which cause movement from one state to another. The probability of an edge provides the conditional probability of the corresponding event. Clearly, the order of the model encodes the memory which the model retains for the generation process. First order models correspond to the least amount of retained
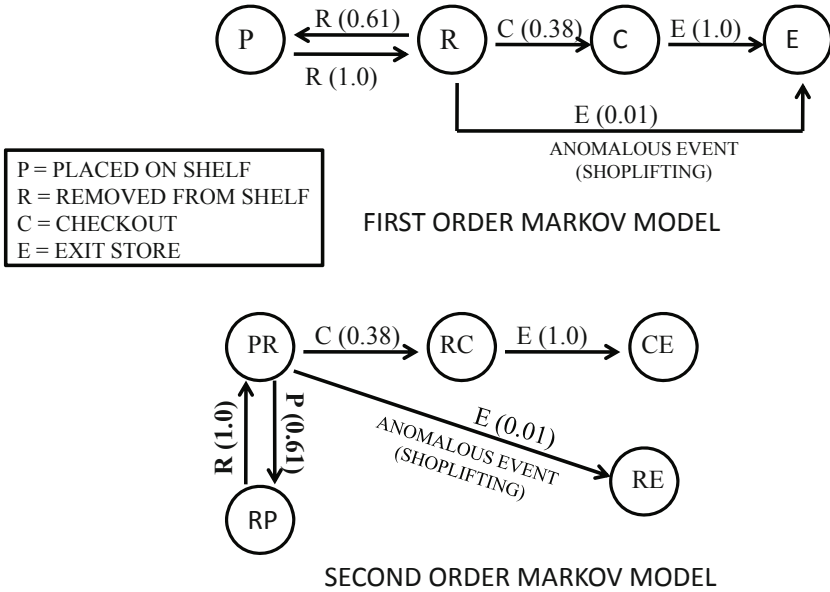
Figure 9.1. Markov Model for the RFID-based shoplifting anomaly

memory. From the perspective of the (equivalent) rule-generation model, first order models simply encode the case, where the antecedent of the rule $a_{n-1} \Rightarrow a_n$ is of unit length. The k-th order models correspond to rules, whose antecedents are of length k. For a given test sequence, windows of size $(k + 1)$ are extracted from the sequence. The first k symbols in it are used to determine the relevant state in the model, and determine the probability that the $(k + 1)$th event is the same as that which occurs in the test sequence. Note that this corresponds to the firing of a rule in the equivalent rule-based models. In the event that the transition impled by the test subsequence is not defined, its probability is estimated to be 0. A simple first order model is proposed in [488], and a kth order model is proposed in [332].

In order to understand how Markov Models work, the previous example of tracking items with RFID tags will be re-visited. An example of the different states of an item along with transitions is illustrated in Figure 9.1. Both a first-order and a second-order model have been illustrated in the figure. The edge transition probabilities are represented along with the edges, and are typically estimated from the training data. The transitions which correspond to the shoplifting anomaly are marked in both models, and correspond to a low transition probability. This is

a particularly simple example, in which a memory of one event is suffi-cient to completely represent the state of an item. In this case, identical results may be obtained from the first-order and second-order models in terms of the transition probabilities. This is not the case in general. For example, consider the case of a web log in which the Markov Models correspond to sequences of web pages visited by users. In such a case, the probability distribution of the next web page visited depends not just on the last page visited, but also on the other preceding visits by the user [140].

An observation from Figure 9.1 is that the number of states in the second-order model is larger than those in the first order model. This is not a coincidence. In general, as many as $|\Sigma|^k$ states may exist in an order-k model. Of course, many of these subsequences may either not occur in the training data, or may be invalid in a particular ap-plication. For example, a PP state would be invalid in the example of Figure 9.1, since the same item cannot be sequentially placed twice on the shelf without removing it at least once. Higher order models rep-resent complex systems more accurately at least at a *theoretical level.* However, choosing models of much higher order degrades *both* effective-ness and efficiency. The effectiveness is degraded, because the transition probabilities need to be estimated from the training data, and each tran-sition now represents a conditional on a sequence of length $k > 1$. Such sequences may be few in the training data, as a result of which the *estimation* process may be very inaccurate. This may be considered a kind of over-fitting, which will impact the overall accuracy of the model. Furthermore, the larger number of states will also make the estimation process much slower. Therefore, a number of relaxations and variations of order-k models can be constructed for greater robustness. These are analogous to the variations of rule-based methods.

- *Variable-order Models:* In these methods a state in the model might correspond to different orders, depending upon its frequency in the data. Higher order states with very low frequency can be pruned from the model, and replaced with lower order generaliza-tions. A method to achieve this goal with the use of *Probabilistic Suffix Trees (PST)* is presented in [434].

- *"Don't care" subsequences:* Each state of the Markov Model rep-resents a subsequence of length k. Allowing a "don't care" as a valid symbol in the subsequence significantly generalizes the state. This reduces the number of states, and also increases the number of training subsequences matching a state. This increases the ro-
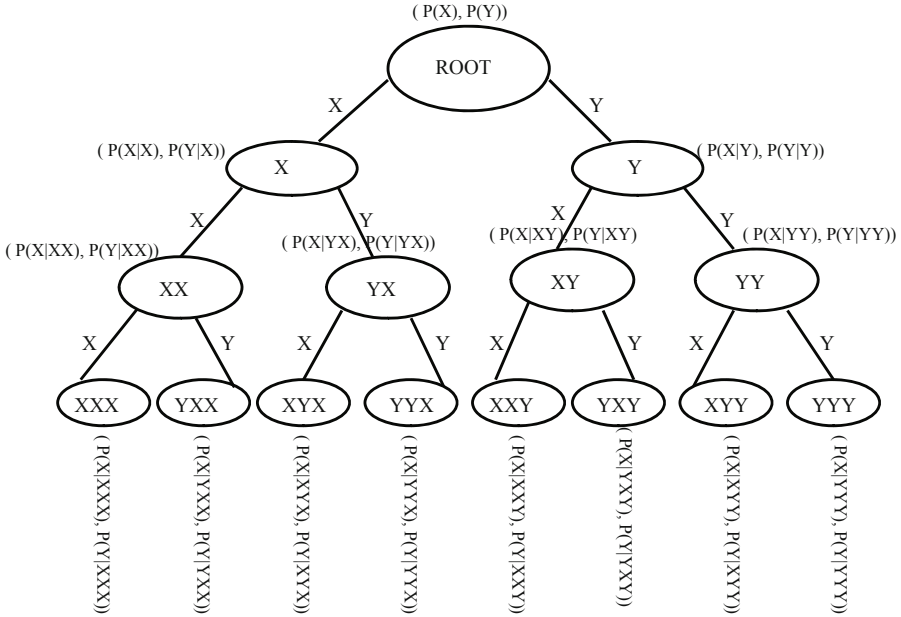
Figure 9.2. Probabilistic Suffix Tree

bustness and efficiency of the approach. Such models are referred
to as *Sparse Markov Transducers (SMT)* [156].

The aforementioned variations are computationally challenging, since
the number of states is likely to be larger in a variable order model.
Therefore, efficient data structures are required for representation and
processing.

2.3   Efficiency Issues: Probabilistic Suffix Trees

It is evident from the discussion in the previous sections that the
Markovian and rule-based models are equivalent, with the latter being a
simpler and easy-to-understand heuristic approximation of the former.
Nevertheless, in both cases, the challenge is that the number of possible
antecedents of length k can be as large as $|\Sigma|^k$. This can make the
methods rather slow, when a lookup for a test subsequence $a_{i-k} \dots a_{i-1}$
is required in order to determine the probability of $P(a_i|a_{i-k} \dots a_{i-1})$. If
these probability values are not organized properly, this can significantly
slow down the approach, when retrieving the probability values for a
particular subsequence. In many cases, the conditional probability needs

to be estimated for each position in the sequence. For longer sequences, this can be extremely slow.

Suffix trees [196] are a classical data structure, which store all subsequences in a given database. Probabilistic suffix trees (PST) represent a generalization of this structure, which also stores the conditional probabilities of generation of the next symbol for a given sequence database. For the case of order-k Markov Models, a suffix tree of depth at most k will store all the required conditional probability values for the kth order Markovian models, including the conditionals for all lower order Markov Models. Therefore, such a structure encodes all the information, which is required for variable order Markov Models as well. Of course, a key challenge is that the number of nodes in such a suffix tree can as large as $\sum_{i=0}^{k} |\Sigma|^k$, an issue which needs to be addressed with selective pruning.

A probabilistic suffix tree is a hierarchical data structure representing the different suffixes of a sequence. A node in the tree with depth k represents a suffix of length k, and is therefore labeled with a sequence of length k. The parent of a node $a_{i-k} \ldots a_i$ corresponds to the sequence $a_{i-k+1} \ldots a_i$, which is obtained by removing the *first* symbol from the sequence. Each edge is labeled with this symbol which needs to be removed, in order to derive the sequence at the parent node. Thus, a path in the tree corresponds to *suffixes* of the same sequence. Each node also maintains a vector $|\Sigma|$ probabilities, which correspond to the conditional probability of the generation of any symbol from $\Sigma = \{\sigma_1 \ldots \sigma_{|\Sigma|}\}$ after that sequence. Therefore, for a node corresponding to the sequence $a_{i-k} \ldots a_i$, and for each $j \in \{1 \ldots |\Sigma|\}$, the values of $P(\sigma_j | a_{i-k} \ldots a_i)$ are maintained. As discussed earlier, this corresponds to the conditional probability that σ_j appears immediately *after $a_{i-k} \ldots a_i$*, once the latter sequence has already been observed. This provides the generative probability which is crucial to the determination of position outliers. An example of a suffix tree with the two symbol alphabets $\Sigma = \{X, Y\}$ is illustrated in Figure 9.2. The two possible symbol generation probabilities at each node corresponding to either symbol X or Y are placed next to the corresponding nodes. It is also evident that a probabilistic suffix tree of depth k encodes *all* the transition probabilities for Markovian models up to order k. Therefore, such an approach can be used for variable order Markovian models.

The probabilistic suffix true is pruned significantly in order to improve its compactness. For example, suffixes which correspond to very low counts in the original data can be pruned from consideration. Furthermore, nodes with low generative probabilities of their underlying sequences can be pruned from consideration. The generative probability

of a sequence $a_1 \ldots a_n$ is approximated as follows:

$$P(a_1 \ldots a_n) = P(a_1) \cdot P(a_2|a_1) \ldots P(a_n|a_1 \ldots a_{n-1}) \qquad (9.1)$$

For Markovian models of order $k < n$, the value of $P(a_r|a_1 \ldots a_{r-1})$ in the equation above is approximated by $P(a_r|a_{r-k} \ldots a_{r-1})$ for any value of k less than r. In order to create Markovian models of order k or less, it is not necessary to keep portions of the tree with depth greater than k. These observations have been used in order to create an efficiently pruned suffix tree for outlier analysis in [434].

Now consider the sequence $a_1 \ldots a_i \ldots a_n$, in which it is desired to test whether position a_i is a position outlier. Then, it is desired to determine $P(a_i|a_1 \ldots a_{i-1})$. It is possible that the suffix $a_1 \ldots a_{i-1}$ may not be present in the suffix tree, because it may have been pruned from consideration. In such cases, the *short memory* property is used to determine the longest suffix which is $a_j \ldots a_{i-1}$, which is present in the suffix tree, and the corresponding probability is estimated by $P(a_i|a_j \ldots a_{i-1})$. Thus, the probabilistic suffix tree provides an efficient way to store and retrieve the relevant probabilities. The length of the longest path which exists in the suffix tree containing a non-zero probability estimate of $P(a_i|a_j \ldots a_{i-1})$ also provides an idea of the level of rarity of this particular sequence of events. Positions which contain only short paths preceding them in the suffix tree are more likely to be outliers. Thus, outlier scores may be defined from the suffix tree in multiple ways:

- If only short path lengths exist in the (pruned) suffix tree corresponding to a position a_i and its preceding history, it is more likely be an outlier.

- For the paths of lengths $1 \ldots r$, which do exist in the suffix tree for position a_i, a combination score may be used based on the models of different orders. In some cases, only lower order scores are combined. In general, the use of lower order scores is preferable, since they are usually more robustly represented in the training data.

One property of the suffix tree is that it can also be used to approximate the generative probability of local subsequences of the data with the use of Equation 9.1. This can be used to determine short *segments* of sequences, which are anomalies, by determining the segments with low generative probabilities. These are referred to as *combination outliers*, which will be discussed in the next section.

Position outliers can also be used for *correcting* noise in sequences, by replacing sequence values with predicted values. Predicted (corrected)

values can be obtained from the suffix tree as the symbols with the highest probability of generation after a suffix. Such a noise-correction approach has been discussed in detail in [386]. Such methods can also potentially be used to correct grammatical errors in text sentences, where each sentence is treated as a sequence of words, and the training data contains a large number of grammatically correct sentences.

3.   Combination Outliers

In combination outliers, the goal is to determine unusual combinations of symbols in a given sequence, with respect to other sequences. There are several versions of the problem corresponding to whether or not one-class semi-supervision is used, the size of the compared sequences, and whether the base data contains multiple sequences or whether it consists of a single long sequence. Interestingly, all these different versions of the problem can be reduced to the same primitive family of sequence anomaly detection problems, except that different *wrappers* need to be used for different scenarios. Many of these formulations are treated very differently in the literature, and are often not connected with one another, because of the diverse problem domains in which they arise. A number of selective reductions between some of these formulations are discussed in [108]. The treatment in this book is more unified, since the *specific model used* (eg. frequency-based method vs distance-based method vs Hidden Markov Modeling method) has been cleanly separated from the methodology itself. The presentation in this chapter goes further by treating all these formulations as different applications of the same primitive formulation. Different *methodological* variations to solve this same problem are then discussed in subsequent sections.

One challenge is that sequence-based outlier detection arises in rather diverse scenarios. Sequence outlier analysis can be formulated very differently, because the data may be present in many different formats, such as a single sequence, multiple sequences, long or short sequences, and the level of differentiation between training and test sequences. However these scenarios are often not too different from one another as long as the data is pre-processed appropriately. Therefore, these different formulation scenarios will be discussed first in order to either sharpen their distinction or approximate equivalence.

Unsupervised vs Semi-Supervised: In the unsupervised version of the problem, all anomalous sequences are determined within a database of sequences. In the semi-supervised version of the problem, an anomaly score is determined for a *test* sequence, with respect to a *training* database

of *normal* sequences. At the formulation level, there is no difference between these problems except for the fact that the training database is more robust for the semi-supervised case, since it is guaranteed not to contain any anomalies. For discussion purposes, the semi-supervised version of the problem will be used, since the model of comparing a test sequence with a set of training sequences is more basic, and can be generalized to the other version as a repetitively used primitive subroutine.

Long Test Sequence vs Short Test Sequence: When the test sequence is relatively short, the sequence can be directly compared to windows of the training sequences in order to determine its *rarity*. In cases, where both the test sequences and training sequences are short, and of comparable length, multidimensional methods for anomaly detection such as the *k*-nearest neighbor method can be adapted to this case. This can be achieved by defining appropriate measures of similarity between test and training sequences. These are referred to as *point-based* techniques. The most difficult case is one where both the test sequence and the training sequence are relatively long.

When the test and training sequences are long, it is not meaningful to compare whole sequences. In such cases, the curse of dimensionality prevents an informative computation of distances over long sequences. Similar to subspace methods, windows of the test sequence are extracted as small subsequences. Such small subsequences will henceforth be referred to as *comparison units*. Then, the *relative behavior* of the comparison units is compared in both the training data and test sequence. If this comparison unit (or small variations of it) are much less prevalent in the training sequence, as compared to the test sequence, it is considered an anomaly. As will be discussed later in this section, the notion of similarity will be quantified in different ways. It can be a distance value, a frequency of presence, or a generative probability, as in a *Hidden Markov Model (HMM)*. The final anomaly score for a given test sequence can be derived as a *combination score* from all the subsequences extracted from it. Different combination scores can be used, depending upon the model which is used. As will be discussed later, different models may behave differently. For example, for the case of distance based approaches, the test sequence need not be explicitly compared to its derived subsequence, since it is guaranteed to be contained in the sequence. Therefore, absolute distances to equivalent length windows in training sequences are used as the anomaly score.

Single Long Training Sequence vs Many Training Sequences: The sequence database may contain a single long training sequence, or

it may contain many training sequences of modest size. The test instance may be of either small or of modest size, but it will typically be much smaller than a single long training sequence. At a logical level, there is little difference between these cases, because all distance-based, frequency-based and markov models can be derived equivalently either from a single long training sequences or multiple training sequences. This is because the models describe the behavior of only a very small window of the data corresponding to the *comparison unit.* For example, when a comparison unit is compared to training sequences in a k-nearest neighbor approach, the anomaly score is the k-th nearest neighbor distance among all (similar size) subsequences in the training data, whether they are derived from the same series or multiple series. For greater generality, it will be henceforth assumed that the training data contains multiple sequences.

Single Long Sequence which is undifferentiated between Training and Test Data: It is also possible to create difficult combinations of the aforementioned cases. A particularly common case is that of web logs, where a single long sequence can be extracted, and it might be desirable to determine unusual *portions* of this sequence. The additional step required for this case is to extract portions of the undifferentiated sequence as test sequences. A multi-granularity approach can be used in order to extract test sequences of different lengths, possibly in geometrically increasing sizes. Then a *second level* of smaller subsequences are extracted from the test sequences, which correspond to the *comparison units.* These are used to model the *relative difference* between the derived test sequences and the training sequences in terms of the smaller window-based comparison units.

Presence or Absence of Domain Knowledge about Relevant Comparison Units for Anomaly Detection: In all the cases above, it is assumed that the comparison unit subsequences are derived as contiguous windows from the test sequences. In many applications, specific domain knowledge may be available about important comparison units. For example, in a security application designed to detect unusual login attempts, a sequence such as *Login Password Login Password Login Password* may be very relevant for detecting interesting anomalies. In such cases, the models can be easily evaluated and combined in terms of these *domain-dependent comparison units*, rather than the units extracted from the test sequences. This is a form of supervision with domain-knowledge. In general, when specific comparison units are

available in domain-dependent scenarios, it can significantly improve the quality of the overall results.

It should be noted that all the aforementioned variations are also relevant to the case of outlier analysis in time-series data. Furthermore, the techniques discussed above are also generally applicable to that case. However, these variations have been discussed here rather than in the time series chapter, since discrete data seems to exhibit a wider variation among these different possibilities in real application domains.

3.1 A Primitive Model for Combination Outlier Detection

From the aforementioned discussion, it is clear that in the most general case, a model needs to be constructed on the basis of relative comparisons between three different kinds of sequences: (i) the training sequences, (ii) the test sequence, and (iii) the comparison units. One of these three different kinds of sequences (the comparison unit) is usually extracted from the test sequence, and may not directly be a part of the input on a stand-alone basis, unless specific domain knowledge is available. However, it will still be included in the primitive formulation, since it provides a very general way to define a primitive formulation for combination outlier detection. This formulation and its minor variants are repeatedly used in various forms of sequence anomaly detection. Some notations and definitions will be used in order to distinguish between the training database, test sequence, and the comparison units.

- The training database is denoted by $\mathcal{D}$, and contains sequences denoted by $T_1 \ldots T_N$.

- The test sequence is denoted by V.

- The comparison units are denote by $U_1 \ldots U_r$. Typically, each U_i is derived from small contiguous windows of V. In domain-dependent cases, $U_1 \ldots U_r$ may be provided by the user.

  In the case of small test sequences, a single comparison unit $U_1 = V$ may be used. Furthermore, in cases where the test sequence is of similar (short) length as the training sequences, this reduces to the easier special case of point-anomaly detection.

This sets the stage for defining the primitive sequence anomaly detection problem.

DEFINITION 9.4 (PRIMITIVE SEQ. ANOMALY (RELATIVE)) *Given a training database of discrete sequences $\mathcal{D}$, a test sequence V, and a*

comparison unit U_i, determine the anomaly score of U_i in terms of relative rarity in $\mathcal{D}$ with respect to V with the use of model $\mathcal{M}$.

The model $\mathcal{M}$ may be a distance-based, frequency-based or Hidden Markov Model. Each of these will be discussed in subsequent sections. The comparison units may either be specified by the user, or they may be extracted from small sliding windows of the test sequence. In cases, where the comparison unit U_i is extracted directly from V, the *absolute* rarity (rather than *relative* rarity) of U_i is computed with respect to the training sequences, since the comparison unit is already known to belong to the test sequence. For example, for a distance-based model, the comparison unit will have zero distance from at least one window in the test sequence V, and absolute comparisons to the training sequence will provide the same results as a relative comparison between the training and test data. In such cases, the absolute distances to windows of the training sequence are used as the anomaly score. Even in the case of frequency-based and Markov Models, relative comparisons are possible, though not generally performed in domain-independent scenarios. Thus, the issue of *relative* surprise is more relevant in scenarios where the comparison units are provided as part of the input. For the case where the comparison units are extracted from the sequences, the primitive sequence anomaly detection problem may be restated as follows:

DEFINITION 9.5 (PRIMITIVE SEQ. ANOMALY (ABSOLUTE)) *Given a training database of discrete sequences $\mathcal{D}$, a test sequence V, and a comparison unit U_i extracted from V, determine the absolute anomaly score of U_i with respect to $\mathcal{D}$ with the use of model $\mathcal{M}$.*

Note that the above definition uses the absolute score rather than a relative comparison between the training sequence database and the test sequence. Multiple comparison units are extracted from V using a sliding window method, where each U_i corresponds to a contiguous window of V.

3.1.1   Model-Specific Combination Issues.   For each of the two formulations introduced in the previous section, the anomaly score of the model is computed with respect to a comparison unit. However, multiple comparison units are extracted from a test sequence. As a post-processing step, it is needed to compute the overall anomaly score of the test sequence by combining the results from different comparison units. The precise method for combining the different scores will be discussed along with the model in each of the following subsections. These methods are similar in spirit and principle, though differences may exist at the

detailed level, because of the differences in how the anomaly score for each comparison unit is quantified by the different models.

3.1.2   Easier Special Cases.   An important special case is one in which the test sequence and training sequences are short, and of comparable size. In such cases, extraction of comparison units becomes irrelevant. Furthermore, it is much easier to compute similarity measures between pairs of *short* sequences in a more robust way, because the curse of dimensionality becomes less relevant. Such cases can be reduced to point-anomaly detection. Some of the models discussed below can also be applied to these simplified cases. Since this is a common special case, a separate formulation is defined for this problem.

DEFINITION 9.6 (WHOLE SEQUENCE ANOMALY (SEMI-SUPERVISED))
Given a database $\mathcal{D}$ of short training sequences $\{T_1 \ldots T_N\}$, and a test sequence V of comparable size, determine the anomaly score of V with respect to $\mathcal{D}$ with the use of model $\mathcal{M}$.

The unsupervised variation of the above problem is very similar, and can be stated as follows:

DEFINITION 9.7 (WHOLE SEQ. ANOMALY (UNSUPERVISED)) *Given a database $\mathcal{D}$ of short training sequences $\{T_1 \ldots T_N\}$, determine all anomalous sequences with the use of model $\mathcal{M}$.*

As discussed earlier in this chapter, the unsupervised and semi-supervised versions are quite similar, and do not need separate treatment. Note that both formulations above are missing the comparison unit. Short sequences do not require the extraction of smaller comparison units for robust distance computations in anomaly detection. In the following sections, the problem of whole sequence anomaly detection will also be discussed. While certain models such as distance-based models and Hidden Markov Models can be used effectively for this case, frequency-based models cannot be used in such scenarios. This is because frequency-based models are dependent on the repetitive frequencies of short comparison units within long sequences. This is not possible in scenarios where all sequences are short.

3.1.3   Relationship between Position and Combination Outliers.   While the most popular and robust methods for detecting combination outliers use window-based comparison units, it is sometimes possible to determine the outlier score for smaller test sequences by repeated use of combination outliers. Specifically, for a sequence

$a_1 \ldots a_n$, its anomaly score can be estimated as the product of the following conditionals.

$$P(a_1 \ldots a_n) = P(a_1) \cdot P(a_2|a_1) \ldots P(a_n|a_1 \ldots a_{n-1}) \qquad (9.2)$$

Note that each of these probability values is available in a probabilistic suffix tree of order (depth) n. For cases, where only a depth of k is maintained, the short-memory property of sequences can be used in order to approximate $P(a_i|a_1 \ldots a_{i-1})$ with $P(a_i|a_{i-k} \ldots a_{i-1})$. Such an approach has been used in [434] for efficient determination of sequence outliers.

3.2   Distance-based Models

In distance-based models, the absolute distance of the comparison unit is computed to equivalent windows of the training sequence. The distance of the k-th nearest neighbor window in the training sequence is used in order to determine the anomaly score. In the context of sequence data, most of the proximity-functions are *similarity* functions rather than *distance* functions, in which higher values indicate greater proximity. Some common methods which are used in order to compute the similarity between a pair of sequences are as follows:

- *Simple Matching Coefficient:* This is the simplest possible function and determines the number of matching positions between two sequences of equal length. This is also equivalent to the Hamming distance between a pair of sequences.

- *Normalized Longest Common Subsequence:* The longest common subsequence can be considered the sequential analogue of the cosine distance between two ordered sets. Let T_1 and T_2 be two sequences, and the length of longest common subsequence between T_1 and T_2 be denoted by $L(T_1, T_2)$. Then, the value $NL(T_1, T_2)$ of the normalized longest common subsequence is computed by normalizing $L(T_1, T_2)$ with the underlying sequence lengths in a similar way as the cosine computation between unordered sets:

$$NL(T_1, T_2) = \frac{L(T_1, T_2)}{\sqrt{|T_1|} \cdot \sqrt{|T_2|}} \qquad (9.3)$$

  The advantage of this approach is that it can match two sequences of unequal lengths. The downside is that the computation process is relatively slow.

- *Edit Distance:* The edit distance is one of the most common similarity functions used for sequence matching [196]. This function

measures the distance between two sequences by the minimum number of edits required to transform one sequence to the other. The computation of the edit distance can be computationally very expensive.

■ *Compression-based Dissimilarity:* This measure is based on principles from information theory. Let W be a window of the training data, and $W \oplus U_i$ be the string representing the concatenation of W and U_i. Let $DL(S) < |S|$ be the description length of any string S after applying a standard compression algorithm to it. Then, the compression-based dissimilarity $CD(W, U_i)$ is defined [259] as follows:

$$CD(W, U_i) = \frac{DL(W \oplus U_i)}{DL(W) + DL(U_i)} \tag{9.4}$$

This measure always lies in the range $(0, 1)$ and lower values indicate greater similarity. The intuition behind this approach is that when the two sequences are very similar, the description length of the combined sequence will be much smaller than that of sum of the description lengths. On the other hand, when the sequences are very different, the description length of the combined string will be almost the same as the sum of the description lengths.

■ *Other Methods:* A variety of other methods may be used in order to compute similarity. These methods vary in terms of how two sequences may be aligned or the importance given to different lengths of the alignment. Some examples of such methods include counting mismatches among lookahead pairs [164], and a length-sensitive recursive computation of subsequence similarity [284, 285]. The latter method is of particular interest and is discussed below.

An important observation in the context of many sequence applications such as intrusion detection is that *contiguous mismatches are more important than non-contiguous mismatches.* This is because anomalous events usually cause *bursts* of anomalous symbols, which can be distinguished from the occasional noise. Let U_i be a comparison unit, and W be a window of the training sequence of the same length as U_i. For each position l in U_i, the length p of the longest contiguous set of positions to the left of position l is determined, so that the position indices $\{l - p + 1, \ldots l\}$ in both sequences match exactly. This length is aggregated over the different values of l from 1 to $|U_i|$. The intuition here is that long contiguous matches are rewarded significantly, as compared to matches which are distributed over different positions.

In order to compute the anomaly score for a comparison unit U_i with respect to the training sequences in $T_1 \ldots T_N$, the first step is to extract equivalent windows from $T_1 \ldots T_N$ as the size of the comparison unit. The k-th nearest neighbor distance is used as the anomaly score *for that comparison unit.* It now remains to explain how the results for the different comparison units $U_1 \ldots U_r$ extracted from V are combined together in order to provide a unified anomaly score for the test sequence V.

3.2.1   Combining Anomaly Scores from Comparison Units.

The anomaly scores from different comparison units extracted from a test sequence can be combined together in order to create a global anomaly score for the test sequence V. For ease in discussion, it will be assumed that higher scores correspond to greater outlierness. When outlier scores correspond to similarity values rather than distance values, the inverse can be used in order to create an outlier score. Some common combination methods are as follows:

- *Number of Anomalous Units:* For each comparison unit, its anomaly score is compared to a specific threshold. The number of anomalous units among $U_1 \ldots U_r$ is reported as the final anomaly score. Note that this method is not specific to the distance-based model for anomaly detection, and can be used in the context of any of the models. For example, such an approach has been used in the context of intrusion detection in [163, 222]. The disadvantage of this approach is that it does not account for varying levels of anomaly scores.

- *Aggregate Anomaly Score:* This aggregates the anomaly score over all comparison units. While such an approach accounts for varying levels of anomaly scores, it can be impacted from the noisy scores of windows which are not anomalous.

- *Selective Aggregate Anomaly Score:* This approach combines the virtues of the two methods discussed above. First windows with anomaly score greater than a threshold are identified. The anomaly scores over these windows are aggregated in order to yield a final anomaly score.

- *Clustered Anomaly Scores:* It is generally recognized that *contiguous anomalous windows are generally more significant in application-specific scenarios than anomalous windows which are arbitrarily distributed over the sequence.* This is because in many motivating applications such as intrusion detection, anomalies are rare,

but they often occur over locally clustered windows when they do occur. The use of a clustered score abstracts out the occasional noise.

- *Locality Frame Count:* A method known as *Locality Frame Count (LFC)* proposed in [163] examines each possible set of η contiguous comparison units, as a *super-unit.* If number of anomalous units within this super-unit is larger than a given threshold, then the entire super-unit is deemed anomalous. The total number of anomalous super-units is reported as the outlier score. Note that this method is not specific to distance-based methods. In fact, a similar method has been used for HMM-based models, which will be discussed later in this chapter [177].

- *Leaky Bucket:* A method based on a similar principle has been proposed in [183]. In this method, the comparison units are scanned in temporal order, and a running count is maintained of the difference between the number of anomalous comparison units and the number of normal units. However, the running count is never allowed to fall below 0. The highest value of the running count over the entire sequence is reported as the outlier score. Such an approach also favors highly clustered anomalous comparison units. For example, interleaved normal and abnormal units will have a (combined) anomaly score no larger than 1. Note that the use of the leaky bucket is not specific to the use of distance-based methods, but can be used with virtually any model.

Numerous variations are possible during the implementation process of above techniques. For example, the declaration of a particular comparison unit as anomalous requires statistical modeling. Different extreme value techniques can be used in order to identify thresholds at which scores should be considered anomalous. Such methods are discussed in Chapter 2.

3.2.2 Some Observations about Distance-based Methods.

Distance-based methods are more suitable for cases in which the comparison units are directly extracted from the test sequence. On the other hand, in many scenarios, comparison units may be provided by a domain expert, and may correspond to significant semantic events (eg. known intrusion patterns, shoplifting patterns, hacking attack patterns etc.). These correspond to the formulation in Definition 9.4. In such cases, other classes of models such as frequency-based or model-based meth-

ods (HMM) should be used. The latter classes of methods are suitable *both* for relative comparisons based on domain-dependent subsequences, and for absolute comparisons to the training data based on comparison units extracted directly from test sequences.

3.2.3   Easier Special Case: Short Sequences.   The use of extracted comparison units or domain-specific comparison units is necessitated by the curse of dimensionality in distance computations over longer sequences. However, when the data contains a set of relatively short sequences of comparable size, straightforward generalizations of multidimensional methods can be used. This scenario corresponds to Definitions 9.6 and 9.7. Any standard k-nearest neighbor technique or clustering method discussed in Chapter 4 may be used. The major distinction is that the distances need to be computed directly on the sequences. For that purpose, any of the aforementioned distance (or similarity) functions in this chapter can be used. Specific examples of algorithms of the two types are as follows:

- A k-nearest neighbor approach, which uses the inverse of the similarity value as the anomaly score, is proposed in [97]. Such an approach has been shown to be quite effective in spite of its simplicity.

- The work in [84, 85] uses a k-medoid based clustering approach in order of determine relevant outliers. A method called *CLUSEQ* for clustering with the use of probabilistic suffix trees is proposed in [481]. The core idea is that a probabilistic suffix tree provides an efficient representation of a cluster, and this can be used for efficient similarity computations in the clustering process. The probabilistic suffix tree is pruned of infrequent nodes in order to ensure a more compact tree for efficiency. The approach is designed for clustering as the primary goal, but is also able to determine outliers as a side product.

3.3   Frequency-based Models

Frequency-based models are typically used with domain-specific comparison units, which are specified by the user. In this case, the relative frequency of the comparison unit needs to be measured in the training sequences and the test sequences, and the level of surprise is correspondingly determined. While the model proposed in this case was originally designed for user-specified comparison units, it can easily be extended to the case of extracted comparison units.

3.3.1 Frequency-based Model with User-specified Comparison Unit.

When the comparison units are specified by the user [257], a natural way of testing the anomaly score is to test the frequency of the comparison unit U_j, in the training and test patterns. For example, when a sequence contains a hacking attempt such as a sequence of *Login* and *Password* events, this sequence will have much higher frequency in the test sequence, as compared to the training sequences. The specification of such relevant comparison units by a user provides very useful domain knowledge to an outlier analysis application.

Let $f(T, U_j)$ represent the number of times that the comparison unit U_j occurs in the sequence T. Since the frequency $f(T, U_j)$ depends on the length of T, the normalized frequency $\hat{f}(T, U_j)$ may be obtained by dividing the frequency with the length of the sequence:

$$\hat{f}(T, U_j) = \frac{f(T, U_j)}{|T|}$$

Then, the anomaly score of the training sequence T_i with respect to the test sequence V is defined by subtracting the relative frequency of the training sequence from the test sequence. Therefore, the anomaly score $A(T_i, V, U_j)$ is defined as follows:

$$A(T_i, V, U_j) = \hat{f}(V, U_j) - \hat{f}(T_i, U_j)$$

The absolute value of the average of these scores are computed over all the sequences in the database $\mathcal{D} = T_1 \dots T_N$. This represents the final anomaly score.

A useful output of this approach is the specific subset of comparison units specified by the user that are the most anomalous. This provides intensional knowledge and feedback to the analyst about *why* a particular test sequence should be considered anomalous. A method called *TARZAN* [257] uses suffix tree representations to efficiently determine all the anomalous subsequences in a comparative sense between a test sequence and a training sequence.

One issue with this model is that since the comparison units are specified by the user, they may not always be of the compact size required for effective frequency computations (unlike the case, when comparison units are extracted from test sequences with compactness in mind). When the comparison units themselves are long, the frequencies of these units in the training sequences may be small or zero. As a result, the anomaly scores may no longer remain significant. A number of methods have been proposed in the literature in order to address these issues, most of which have to do with examining the behavior of smaller portions of the comparison units. In particular, a method proposed in [257]

uses subsequences of the comparison units, and computes an aggregate quantity as a function of the frequencies of the underlying subsequences. A different method in [200] allows relaxation of the definition of "subsequence" in counting frequencies by allowing permutations of the comparison unit. The assumption is that the exact ordering of the symbols within a short window is not as important in certain application-specific scenarios. In fact, in such cases, relaxations can add to the robustness of the similarity computations.

In order to improve the *efficiency* of frequency computation, the work in [201] suggests that the training sequences should be decomposed into smaller windows for subsequence computations. The number of windows in which the comparison unit occurs is used as a proxy for the frequency.

3.3.2   Frequency-based Model with Extracted Comparison Units.
The frequency-based model is mostly designed for the case of user-specified comparison units. However, it is also possible to generalize the model to the case where extracted comparison units are used. In this model, the comparison unit is extracted directly from the test sequence. For each comparison unit, the same model may be used, as in the case of user-specified comparison units. While such methods are feasible, they have generally not been used very frequently in the literature. Distance-based methods are much more popular, when comparison units are extracted directly from the test sequences.

3.3.3   Combining Anomaly Scores from Comparison Units.
While frequency-based methods are generally used for surprise detection with user-specific comparison units, the results from multiple comparison units can also be combined into a single anomaly score. When many comparison units are specified by a user, this can be used to create a unified comparison score. The methodology for achieving this goal is the same as discussed for distance-based methods in section 3.2.1. This approach is also applicable, when the comparison units are extracted directly from the test sequences. The former results in a domain-dependent anomaly score, whereas the latter results in an unsupervised anomaly score.

3.4   Hidden Markov Models

Hidden Markov Models (HMM) are probabilistic models which generate sequences through a sequence of transitions between states in a Markov chain. So how are hidden markov models different from the Markovian techniques introduced earlier in this chapter? Each state in the Markovian techniques introduced earlier in this chapter is well de-
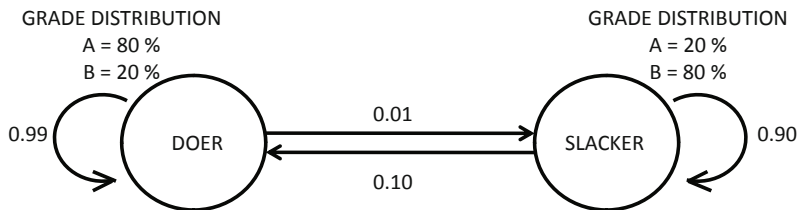
Figure 9.3. Generating grade sequences from a Hidden Markov Model
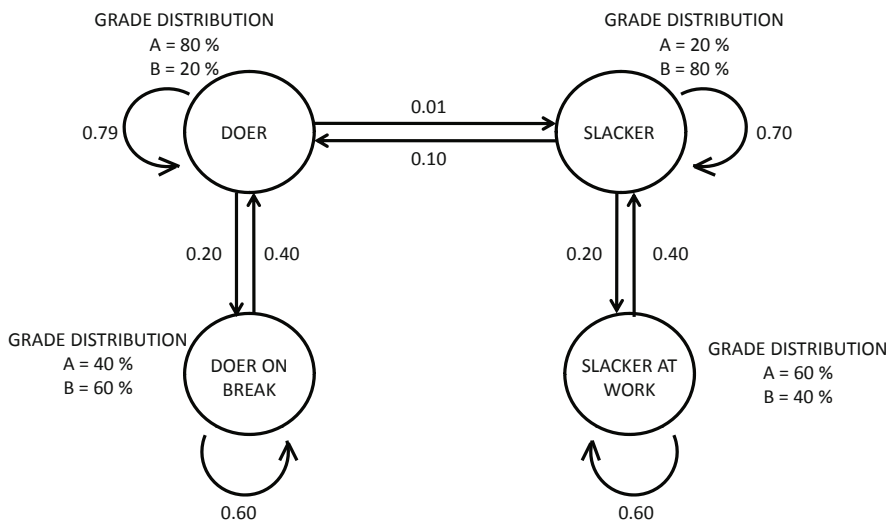
Figure 9.4. Extending the model in Figure 9.3 with two more states provides greater expressive power for modeling sequences

fined, and is based on last k positions of the sequence. This state is also directly visible to the user in terms of the precise order of transitions for a particular training or test sequence. Thus, the generative behavior of the Markovian model is always known completely.

In a Hidden Markov Model, the states of the system are *hidden*, and not directly visible to the user. Only a sequence of (typically) discrete

observations are visible to the user, which are generated by symbol emissions from the states after each transition. This corresponds to the application-specific sequence data. In many cases, the states may be defined (during the modeling process) on the basis of an *understanding* of how the underlying system behaves, though the precise sequence of transitions may not be known to the analyst. This is why such models are referred to as "*hidden*".

Each state is associated with an *set of emission probabilities* over the symbol Σ. In other words, a visit to the state j leads to an emission of one of the symbols $\sigma_i \in \Sigma$ with probability $\theta^j(\sigma_i)$. Correspondingly, a sequence of transitions in a Hidden Markov Model corresponds to an *observed data sequence*. Hidden Markov Models may be considered a kind of mixture model of the type discussed in Chapter 2, in which the different components of the mixture are not independent of one another, but are related through sequential transitions. Thus, each state is analogous to a component in the mixture of the multidimensional mixture model discussed in Chapter 2. Each symbol generated by this model is analogous to a data point generated by the multidimensional mixture model. Furthermore, the successive generation of individual data items (sequence symbols) is also not independent of one another. This is a natural consequence of the fact that the successive states emitting the data items are not independent of one another, but are continually transitioning to one another at different pairwise probabilities. Unlike standard mixture models, Hidden Markov Models are designed for sequential data, which exhibits temporal correlations.

In order to better explain Hidden Markov Models, an illustrative example will be used. Consider the scenario, where a set of students register for a course, and generate a sequence corresponding to the grades received in each of their weekly assignments. This grade is drawn from the symbol set $\Sigma = \{A, B\}$. *The model created by the analyst is that* the class contains students who, at any given time, are either *doers* or *slackers* with different grade-generation probabilities. A student in *doer* state may sometimes transition to *slacker* state and vice-versa. These represent the two states in the system. Weekly home assignments are handed out to each student, which are graded with one of the symbols from Σ. This results in a *sequence* of grades for each student, and it represents the only *observable* output for the analyst. The state of a student only represents a *model* created by the analyst in order to *explain* the grade sequences, and is therefore not observable in of itself. It is important to understand that if this model is a poor reflection of the true generative process, then it will impact the quality of the learning process.

Assume that a student in *doer* state is likely to get an *A* grade in a home-assignment with 80% probability, and a *B* with 20% probability. For *slacker*s, these probability values are reversed. While these probabilities are explicitly specified here for illustrative purposes, they need to be *learned* or *estimated* from the observed grade sequences for the different students, and are not known a-priori. The precise status (state) of any student in a given week is not known to the analyst at any given time. These grade sequences are in fact the only *observable* outputs for the analyst. Therefore, from the perspective of the analyst, this is a *Hidden* Markov Model, which generates the sequences of grades from a *unknown* sequence of states, representing the state transitions of the students. The precise sequence of transitions between the states can only be *estimated* for a particular observed sequence.

The two-state Hidden Markov Model for the aforementioned example is illustrated in Figure 9.3. This model contains two states corresponding to *doer* and *slacker*, which represent the state of a student in a particular week. It is possible for a student to transition from one state to another, though the likelihood of this is rather low. It is assumed that set of initial state probabilities govern the a-priori distribution of *doer*s and *slacker*s. This represents the a-priori understanding about the students when they join the course. Some examples of *typical sequences* generated from this model along with their rarity level, are illustrated in Figure 9.3. For example, the sequence `AAABAAAAABAAAA` is most likely generated by a student who is consistently in a *doer* state, and the sequence `BBBBABBBABBBB` is most likely generated by a student who is consistently in *slacker* state. The second sequence is typically rarer than the first, since the population mostly contains[1] *doers*. The sequence `AAABAAABBABBB` corresponds to a *doer* who eventually transitions into a *slacker*. This case is even rarer, because it requires a transition from the *doer* state to a *slacker* state, which has very low probability. The sequence `ABABABABABABA` is *extremely anomalous*, because it does not represent temporally consistent *doer* or *slacker* behavior that is implied by the model. Correspondingly, such a sequence has very low probability of fitting the model.

A larger number of states in the Markov Model can be used in order to encode more complex scenarios. It is possible to encode domain knowledge with the use of states which describe different generating scenarios. In the example discussed earlier, consider the case that a *doer* some-

---

[1]The assumption is that the initial set of state probabilities are approximately consistent with the steady state behavior of the model for the particular set of transition probabilities shown in Figure 9.3.
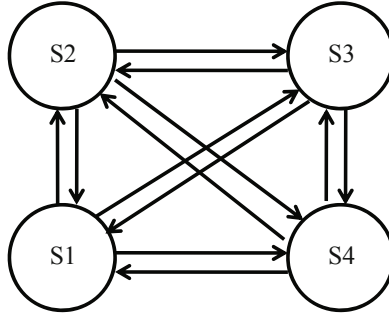
Figure 9.5. A black-box Hidden Markov Model with four states and no encoded domain knowledge: A larger number of transition probabilities need to be learned

times slacks off for short periods and then returns to their usual state. Alternatively, a *slacker* may sometimes become temporarily inspired to be a *doer*, but may eventually return to what they are best at. Such episodes will result in local portions of the sequence which are distinctive from the remaining sequence. These scenarios can be captured with the 4-state Markov Model illustrated in Figure 9.4. The larger the number of states, the more complex scenarios that can be captured. Of course, more training data is required to learn the (larger number of) parameters of such a model, or this may result in overfitting. For smaller data sets, the transition probabilities and symbol generation probabilities are not estimated accurately. As was discussed in Chapter 2, this is a common problem of generative models. Therefore, a number of important issues arise about the initial design choices in a Hidden Markov Model. These will be discussed in the next subsection.

3.4.1   Design Choices in a Hidden Markov Model.   In the previous example, an *understanding* of the generating process was used in order to design the basic architecture of the Hidden Markov Model. Such an understanding may or may not be available in a given application. A good initial design of the Markov Model is crucial, because a poor design could result in the model either overfitting the data or not fitting the training sequences at all. Two primary design choices exist in picking the parameters of a Hidden Markov Model.

- In its more general form, a Hidden Markov Model with n states will have n^2 possible transitions between different states (including self-transitions), and $n \cdot |\Sigma|$ symbol generation probabilities, where one distribution is associated with each state. This is a black-box model, in which no prior knowledge is encoded about the gener-

ating process. In such cases, the initial Markov Model is a clique with n states. In practice, the states may be designed with an intuitive understanding about the underlying generating process, and not all pairs of states may have transitions among them. This encodes significant *domain knowledge* about the sequence generation process. For example, in the case of Figure 9.4, the states are intuitively associated with characteristics of the generating process (students). Furthermore, transitions do not occur between all pairs of states. These represent an a-priori domain knowledge of the analyst about the generative process for the sequences.

The equivalent four-state black-box Markov Model is illustrated in Figure 9.5. Note that the states are not labeled in this case, since they do not reflect an analyst's understanding of the underlying generating process. Clearly, different tradeoffs exist between these choices. By picking a particular topology of the model, based on domain-understanding, the number of parameters to be learned are greatly reduced. For example, fewer transition probabilities need to learned in Figure 9.4 as compared to Figure 9.5. In the former case, many transition probabilities have already been set to zero before the training process, and this reflects the domain-specific understanding. This reduces the likelihood of overfitting in the case of Figure 9.4. On the other hand, if the specific topology of the domain-specific architecture of Figure 9.4 does not truly reflect the student behavior, then the resulting model will also poorly reflect the training data. In this sense, the model of Figure 9.5 has the advantage that it can be used as black-box for the sequences from any domain.

- The number of states represents the level of complexity in the Markov Model. This is analogous to the number of components in the mixture model of Chapter 2. A larger number of states can encode greater complexity about variations in sequence patterns, but may also overfit a training data set, when it is small. Therefore, when the black-box approach is used, it needs to be decided a-priori, how many states should be used in the model. When the data sets available are small, it is advisable to use a smaller number of states.

The first of the above design choices is important, because Hidden Markov Models are particularly effective when they are used to encode domain-specific understanding of the underlying system. The ability to encode a set of sequences with a Markov Model of a particular architecture is dependent upon the understanding of the analyst about the

generating process. As discussed throughout the book, the ability to encode domain knowledge in an indirect way during the model construction process is crucial for effective outlier analysis.

3.4.2   Formal Definition and Techniques for HMM.   In this section, Hidden Markov Models will be formally introduced, along with the associated training methods. It is assumed that a Hidden Markov Model contains n states, which are denoted by $\{s_1 \ldots s_n\}$. The symbol set from which the observations are generated is denoted by $\Sigma = \{\sigma_1 \ldots \sigma_{|\Sigma|}\}$. The symbols are generated from the model by a sequence of transitions from one state to the other. Each visit to a state (including self-transitions) generates a symbol which is drawn from a categorical probability distribution on Σ. The symbol emission distribution is specific to each state. The probability $P(\sigma_i|s_j)$ that the symbol σ_i is generated from state s_j is denoted by $\theta^j(\sigma_i)$. The probability of a transition from state s_i to s_j is denoted by p_{ij}. The initial state probabilities are denoted by $\pi_1 \ldots \pi_n$ for the n different states. The topology of the model can be expressed as a network $G = (M, A)$, in which M is the set of states $\{s_1 \ldots s_n\}$. The set A represents the possible transitions between the states. In the most common scenario, where the architecture of the model is constructed with a domain-specific understanding, the set A is not the complete network, and therefore many transition probabilities are implicitly set to 0. *The goal of training the HMM model is to learn the initial state probabilities, transition probabilities and the symbol emission probabilities from the training database $\{T_1 \ldots T_N\}$.*

Three methodologies are commonly leveraged in creating and using a Hidden Markov Model:

- *Training:* Given a set of training sequences $T_1 \ldots T_N$, estimate the model parameters, such as the initial probabilities, transition probabilities, and symbol emission probabilities with an Expectation-Maximization algorithm. The Baum-Welsch algorithm is used for this purpose.

- *Evaluation:* Given a test sequence V (or comparison unit U_i), determine the probability that it fits the HMM. This is used to determine the anomaly scores. A recursive forward algorithm is used to compute this.

- *Explanation:* Given a test sequence V, determine the most likely sequence of states which generated this test sequence. This is helpful for providing an understanding of why a sequence should be considered an anomaly, when the states correspond to an intuitive understanding of the underling system. In the example of Figure

, it would be useful to know that an observed sequence is an anomaly, because of the unusual oscillatory behavior of a student between *doer* and *slacker* states. This can provide the *intensional knowledge* for understanding the state of a system. This most likely sequence of states is computed with the Viterbi algorithm.

Since the description of the training procedure relies on technical ideas developed for the evaluation method, we will deviate from the natural order of presentation, and present the training algorithms last. The evaluation and explanation techniques will assume that the model parameters such as the transition probabilities are already available from the training phase.

3.4.3 Evaluation: Computing the Fit Probability for Observed Sequence.

One approach for determining the fit probability of test sequence $V = a_1 \ldots a_m$ would be compute all the n^m possible sequences of states (paths) in the HMM, and compute the probability of each of them, based on the observed sequence, symbol generation probabilities, and transition probabilities. The sum of these values can be reported as the fit probability. Obviously, such an approach is not practical, because it requires the enumeration of an exponential number of possibilities.

This computation can be greatly reduced by recognizing that the fit probability of the first r symbols (and a fixed value of the rth state) can be recursively computed in terms of the corresponding fit probability of first $(r-1)$ observable symbols (and a fixed $(r-1)$th state). Specifically, let $\alpha_r(V, s_j)$ be the probability that the first r symbols in V are generated by the model, and the last state in the sequence is s_j. Then, the recursive computation is as follows:

$$\alpha_r(V, s_j) = \sum_{i=1}^{n} \alpha_{r-1}(V, s_i) \cdot p_{ij} \cdot \theta^j(a_r)$$

This approach recursively sums up the probabilities for all the n different paths for different penultimate nodes. The aforementioned relationship is iteratively applied for $r = 1 \ldots m$. The probability of the first symbol is computed as $\alpha_1(V, s_j) = \pi_j \cdot \theta^j(a_1)$ for initializing the recursion. This approach requires $O(n^2 \cdot m)$ time. Then, the overall probability is computed by summing up the values of $\alpha_m(V, s_j)$ over all possible states s_j. Therefore, the final fit $F(V)$ is computed as follows:

$$F(V) = \sum_{j=1}^{n} \alpha_m(V, s_j)$$

This algorithm is also known as the *Forward Algorithm*.

3.4.4 Explanation: Determining the Most Likely State Sequence for Observed Sequence.

One of the goals of outlier analysis in different data domains is to provide an explanation for why a data point or sequence should be considered an outlier. Since the sequence of (hidden) generating states often provide an intuitive explanation for the observed sequence, it is sometimes desirable to determine the *most likely sequence of states* for the observed states. The Viterbi algorithm provides an efficient way to determine the most likely state sequence.

One approach for determining the most likely state path of the test sequence $V = a_1 \dots a_m$ would be compute all the n^m possible sequences of states (paths) in the HMM, and compute the probability of each of them, based on the observed sequence, symbol generation probabilities, and transition probabilities. The maximum of these values can be reported as the most likely path. Note that this is a similar problem to the fit probability except that it is needed to determine the *maximum* over all possible paths, rather than the *sum*. Correspondingly, it is also possible to use a similar recursive approach as the previous case in order to determine the most likely state sequence.

Any sub-path of an optimal state path must also be optimal for generating the corresponding subsequence of symbols. This property, in the context of an optimization problem of sequence selection, normally enables dynamic programming methods. The best possible state path for generating the first r symbols (with the rth state fixed to j) can be recursively computed in terms of the corresponding best paths for the first $(r-1)$ observable symbols and different penultimate states. Specifically, let $\delta_r(V, s_j)$ be the probability of the best state sequence for generating the first r symbols in V and also ending at state s_j. Then, the recursive computation is as follows:

$$\delta_r(V, s_j) = MAX_{i=1}^{n} \delta_{r-1}(V, s_i) \cdot p_{ij} \cdot \theta^j(a_r)$$

This approach recursively computes the maximum of the probabilities all the n different paths for different penultimate nodes. The approach is iteratively applied for $r = 1 \dots m$. The first probability is determined as $\delta_1(V, s_j) = \pi_j \cdot \theta^j(a_1)$ for initializing the recursion. This approach requires $O(n^2 \cdot m)$ time. Then, the final best path is computed by using the maximum value of $\delta_m(V, s_j)$ over all possible states s_j. This approach is essentially a dynamic programming algorithm.

3.4.5 **Training: Baum-Welsch Algorithm.** The problem of determining the optimal parameters in the case of HMM is a very difficult one, and no known algorithm is guaranteed to determine the global optimum. However, a number of options are available to determine a reasonably effective solution in most scenarios. The Baum-Welsch algorithm is one such method. It is also known as the *Forward-backward* algorithm, and it is an application of the EM approach to the generative Hidden Markov Model. First, a description of training with the use of a single sequence $T = a_1 \ldots a_m$ will be provided. Then, a straightforward generalization to N sequences $T_1 \ldots T_N$ will be discussed.

Let $\alpha_r(T, s_j)$ be the *forward* probability that the first r symbols in a sequence T of length m are generated by the model, and the last symbol in the sequence is s_j. Let $\beta_r(T, s_j)$ be the *backward* probability that the portion of the sequence after *and not including the rth position* is generated by the model, *conditional on the fact that* the state for the rth position is s_j. Thus, the forward and backward probability definitions are not exactly symmetric. The forward and backward probabilities can be computed from model probabilities in a similar way as the evaluation procedure discussed above in section 3.4.3. The major difference for the backward probabilities is that the computations start from the end of the sequence in the backwards direction. Furthermore, the probability value $\beta_{|T|}(T, s_j)$ is initialized to 1 at the bottom of the recursion, to account for the difference in the two definitions. Two additional probabilistic quantities need to be defined in order to describe the EM-algorithm:

- $\psi_r(T, s_i, s_j)$: Probability that the rth position in sequence T corresponds to state s_i, the $(r+1)$th position corresponds to s_j.

- $\gamma_r(T, s_i)$: Probability that the rth position in sequence T corresponds to state s_i.

The EM procedure starts off with a random initialization of the model parameters, and then iteratively estimates $(\alpha(\cdot), \beta(\cdot), \psi(\cdot), \gamma(\cdot))$ from the model parameters and vice-versa. Specifically, the iteratively executed steps of the EM-procedure are as follows:

- (E-Step) Estimate $(\alpha(\cdot), \beta(\cdot), \psi(\cdot), \gamma(\cdot))$ from currently estimated values of the model parameters $(\pi(\cdot), \theta(\cdot), p_{..})$.

- (M-Step) Estimate model parameters $(\pi(\cdot), \theta(\cdot), p_{..})$ from currently estimated values of $(\alpha(\cdot), \beta(\cdot), \psi(\cdot), \gamma(\cdot))$.

It now remains to explain how each of the above estimations is performed. The values of $\alpha(\cdot)$ and $\beta(\cdot)$ can be estimated using the forward

and backward procedures respectively. The forward procedure is already described in the evaluation section, and the backward procedure is analogous to the forward procedure, except that it works backwards from the end of the sequence. The value of $\psi_r(T, s_i, s_j)$ is equal to $\alpha_r(T, s_i) \cdot p_{ij} \cdot \theta^j(a_{r+1}) \cdot \beta_{r+1}(T, s_j)$, since the sequence generation procedure can be divided into three portions corresponding to that up to position r, the generation of the $(r+1)$th symbol, and the portion after the $(r+1)$th symbol. The estimated values of $\psi_r(T, s_i, s_j)$ are normalized to a probability vector by ensuring that the sum over different pairs $[i, j]$ is 1. The value of $\gamma_r(T, s_i)$ is estimated by summing up the values of $\psi_r(T, s_i, s_j)$ over fixed i and varying j. This completes the estimations of the E-step.

The re-estimation formulas for the model parameters in the M-Step are relatively straightforward. Let $I(a_r, \sigma_k)$ be a binary indicator function, which takes on the value of 1, when the two symbols are the same, and 0 otherwise. Then the estimations can be performed as follows:

$$\pi(j) = \gamma_1(T, s_j), \ p_{ij} = \frac{\sum_{r=1}^{m-1} \psi_r(T, s_i, s_j)}{\sum_{r=1}^{m-1} \gamma_r(T, s_i)}$$

$$\theta^i(\sigma_k) = \frac{\sum_{r=1}^{m} I(a_r, \sigma_k) \cdot \gamma_r(T, s_i)}{\sum_{r=1}^{m} \gamma_r(T, s_i)}$$

The precise derivations of these estimations on the basis of expectation-maximization techniques may be found in [379]. This completes the estimations for the M-step.

As in all EM methods, the procedure is applied iteratively to convergence. More details on robust termination criteria may be found in [379]. The approach can be generalized easily to N sequences, by applying the steps to each of the sequences, and averaging the corresponding model parameters in each step.

3.4.6   Computing Anomaly Scores.   In theory, it is possible to compute anomaly scores directly for the test sequence V, once the training model has been constructed from the sequence database $\mathcal{D} = T_1 \ldots T_N$. However, as the length of the test sequence increases, the robustness of such a model reduces because of the increasing noise resulting from the curse of dimensionality. Therefore, the comparison units (either extracted from the test sequence or specified by the domain expert), are used for computing the anomaly scores.

- When the comparison units are extracted from the test sequence (the absolute model of Definition 9.5), it is possible to compute the probability of fit of each comparison unit to the training data.

The negative logarithm of the resulting probability provides an anomaly score in which large values are indicative of anomalous behavior for a particular comparison unit. At this point, the method of section 3.2.1 can be used in order to combine the final anomaly scores over different windows.

- When the comparison units are provided by a domain expert (the relative model of Definition 9.5), two separate Markov Models are computed for the training sequences and test sequence respectively. This approach will not work, if the test sequence is not sufficiently long to create a robust model without overfitting. The negative logarithm of the fit probability of the comparison unit to the two models is computed with the Viterbi algorithm. The difference in the two values is reported as the anomaly score.

The second of the two methods is used rarely, because of the tendency of a single test sequence to create a model which overfits the data.

A number of methods also use the Viterbi algorithm on the test sequence in order to mine the most likely state sequence. In some domains, it is easier to determine anomalies in terms of the state sequence rather than the observable sequence. Furthermore, low transition probabilities on portions of the state sequence provide anomalous localities of the observable sequence. The downside is that the most likely state sequence may have a very low probability of matching the observed sequence. Therefore, the estimated anomalies may not reflect the true anomalies in the data, when an *estimated* state sequence is used for anomaly detection. The real utility of the Viterbi algorithm is in terms of providing an *explanation* of the anomalous behavior of sequences in terms of the intuitively understandable states, rather than quantifying its anomaly score.

3.4.7 Special Case: Short Sequence Anomaly Detection.

A special case arises, when database $\mathcal{D}$ of shorter sequences is available, and the anomalous sequences need to be determined, on the basis of their dissimilarity with other sequences. This corresponds to Definition 9.7. As discussed earlier, this problem is similar to point anomaly detection formulations, which are used in multidimensional data. Distance- and clustering-based methods for addressing this special case have been discussed in section 3.2.3. A question arises whether HMMs can also be used for probabilistic clustering and anomaly detection in sequences with the use of mixture models.

It turns out that it is possible to use hidden markov models for clustering and anomaly detection. The idea is to represent the data as a

mixture of HMMs [89, 413]. As discussed earlier, the HMM is itself a mixture model, in which each state can be considered a component of a mixture. This approach uses a *second* level of mixture modeling, where the observations correspond to individual sequences from $\mathcal{D}$ rather than the individual symbols of a sequence. A single HMM model for a training data set makes the implicit assumption that every sequence $T_i \in \mathcal{D}$ is generated from the same distribution (cluster). This is not true in practice, since the individual sequences might themselves belong to different clusters, each of which has its own generating HMM. This can be modeled using a second level of mixture modeling, in which the k different independent HMMs are used, and each sequence $T_i \in \mathcal{D}$ is associated with a generative probability from this mixture. Sequences which have low generative probabilities, or which have low probability of assignment to their best matching component may be considered anomalies.

4.   Complex Sequences and Scenarios

In many real applications, the available sequences may be much more complex than the ones which have been introduced in the chapter. For example, in a system diagnosis application, multiple sequences may continuously be produced by different kinds of sensors. In other cases, *each element* of the sequence may be a *set* drawn from the symbol Σ, rather than a symbol. In some scenarios such as online applications, it may be desirable to declare a sequence an anomaly as early as possible by observing only the early part of it. These cases may sometimes require more complex techniques for analysis than those discussed earlier in this chapter.

4.1   Multivariate Sequences

Multivariate sequences are quite common in system diagnosis applications, in which multiple sensors may record sequences drawn from possibly different alphabets. Consider the case, where r different sensors are available, which record sequences drawn from the alphabets $\Sigma_1 \ldots \Sigma_r$. Each generated symbol is associated with a time-stamp, which is particularly crucial in the multivariate case in order to determine the temporal correspondence between the sequences generated by the different sensors.

Much work has not been done in the literature on the multivariate case for *discrete* sequences, as compared to the continuous case [115]. However, a variety of solutions are possible for finding unusual *time-windows* by combining the results from univariate analysis of the different series. Let $q_i(t_c - h, t_c)$ represent the generative probability of the subsequence generated in the time-interval $(t_c - h, t_c)$. This generative probability can

be can be computed using Equation 9.1 on an individual sequence. The final generative probability for the anomaly score of the *time window* $(t_c - h, t_c)$ over all the r different series is given by $\prod_{i=1}^{r} q_i(t_c - h, t_c)$.

4.2   Set-based Sequences

Most of the algorithms in this chapter are designed for the case of sequences in which each symbol is drawn from the symbol set Σ. In practice, each unit element of the sequence may be a set $S_i \subseteq \Sigma$. Such cases arise commonly in domains such as market-basket analysis. The analysis of anomalies in such complex sequences are different from the case of simpler sequences, because two set-based elements may have varying levels of similarity, depending upon the number of common elements between the sets. Furthermore, temporal correlations between the set-elements in such sequences is typically much weaker than symbol-only sequences, because of the larger number of possibilities for a set element in the sequence. For example, in the market basket scenario, it is generally not meaningful to predict conditional probabilities of set-based symbols based on the previous history.

The problem of finding anomalies in set-based sequences is much closer to the problem of finding unusual novelties or change analysis in multi-dimensional data. Here each set is treated as discrete multidimensional binary vector of dimensionality $|\Sigma|$, corresponding to whether or not an element of Σ is present in the set. A number of methods available in the literature are applicable to these scenarios:

- The models discussed in Chapter 7 for first-story detection [515] in text can be applied to this scenario by treating each set-element analogous to a document, and the symbol set Σ analogous to a text lexicon. In fact, a few methods [26] treat the problem of novelty detection of text, categorical data, and market-basket sequences in a unified way, with the use of clustering. The creation of new clusters in a streaming cluster generation process, corresponds to the relevant anomalies. It has been shown in [26], that such a method can simultaneously determine evolving clusters, anomalies, and newly forming trends in the underlying data.

- Some methods have also been designed to find surprising patterns with the use of the concept of *temporal description length* [96]. The goal is to determine itemsets and data segments which reflect significant changes in correlations over time. This technique borrows ideas from information theory. The itemsets are temporally segmented. For each segment, the minimum number of bits requited to encode each segment is determined. Segments which require a

larger number of bits to encode are assumed to be less homogeneous, and may contain surprising patterns or change points. Such segments may be declared as outliers.

In general, set-based sequences are much closer in spirit to sparse multidimensional data, and provide a much richer domain for analytical purposes. Numerous methods for aggregate change analysis discussed in Chapter 8 can also be easily generalized to this case.

4.3   Online Applications: Early Anomaly Detection

In many sequential data mining applications, such as those arising in web click-streams, or system diagnosis applications, it may be desirable to perform the analysis in online fashion, as more data is received. In such cases, most of the techniques discussed earlier can be adapted to the online case.

- *Position Outliers:* For the case of position outliers, the prediction is dependent only on a short memory before the position being predicted. The current model can be used in order to make an efficient prediction. Furthermore, many of the commonly used training models such as probabilistic suffix trees can be efficiently updates, as more sequences are added to the training database.

- *Combination Outliers:* In these cases, the unit of prediction is typically the comparison unit. As long as the next comparison unit has been derived, it can be used in conjunction with the current model in order to make the prediction. The process of updating the training model may sometimes be more challenging. Some models such as k-nearest neighbors may slow down with increasing training data. In such cases, the training data may need to sampled, in order to retain efficient prediction.

Numerous examples of online sequential anomaly detection methods exist in domains as diverse as systems diagnosis and intrusion detection [10, 19].

5.   Supervised Outliers in Sequences

Because of the complexity of sequence data, it is often hard to obtain meaningful outliers with the use of purely unsupervised methods. Many patterns which are discovered as outliers may correspond to noise, and may not represent true anomalies. It has been shown in diverse application domains such as system call anomalies [284], financial fraud [315],

and web sequence robot detection [440], that the nature of the anomalous sequences are highly specific and governed by the underlying application. Therefore, the ability to distinguish noise from anomalies can only be obtained by incorporating additional knowledge about previously known (and interesting) examples into the outlier analysis process. Such learning methods can significantly improve the quality and relevance of the underlying outliers in real application scenarios.

The problem of supervised outlier detection in sequence data is equivalent to sequence classification. As in all supervised outlier analysis methods, the major difference is in terms of class imbalance and cost sensitivity, which needs to be accounted for in the classification process. This ensures that rare classes can be meaningfully discovered. In these models, labels may be available in different ways:

- Labels may be associated with each position in a sequence. Such situations occur commonly in natural language data, in which the different words may need to be classified in a sentence [41, 283]. In the rare-class versions of this problem, most of the labels may belong to the normal category, but an occasional label may correspond to an anomaly (eg. a word which is "out of place" or grammatically incorrect).

- Rare labels may be associated with small subsequences of a single large sequence [14]. Other subsequences are assumed to be normal by default.

- Labels are associated with the full sequences in a database of multiple sequences. Most labels are normal, but a small number of them may be anomalous. This is the most common scenario in real applications such as intrusion detection.

The second formulation above can be transformed to the last one by extracting windows of subsequences, and classifying them with the use of a full sequence classifier. Furthermore, in many cases of temporal sequences, it is desirable to learn the label of a sequence by examining as little of it as possible from the early portions of the sequence. This is known as *early classification* [475].

While most of the methods for classification of sequence data are not designed for the imbalanced case, the generalization to the imbalanced case is straightforward with the use of the meta-algorithms discussed in Chapter 6. Thus, the issue of class imbalance is orthogonal to the actual techniques which are used for sequence classification. A good survey on sequence classification methods may be found in [475]. While sequence

classification is too vast an area to be covered in this chapter, a brief review of the main classes of techniques are discussed below.

Feature Transformations   The first class of methods extracts symbolic sequences of size k from the base sequences. These are referred to as k-grams, and they form the "words" from the sequence vocabulary. These words are used as the features, and a variety of classifiers such as SVM [299] can be used on this new representation. Another class of common feature extraction methods include *pattern-based methods*, in which relevant patterns which have sufficient support and confidence in terms of the underlying sequence symbols are mined. Note that such an approach mines the *local* patterns from the underlying data, which are relevant to sequence classification. Other forms of feature extraction such as wavelet decomposition [12] can be used, which extract both the local and global properties of the underlying data. A rule-based classifier was constructed which relates the wavelet patterns to the underlying classes.

Distance-based Methods   As discussed in Chapter 8 on time-series outlier analysis, distance-based methods are particularly popular in the context of continuous time-series. In the case of continuous data, distance functions such as the Euclidean metric can be used. However, for discrete data sets, popular methods for sequence alignment such as the edit distance [196] can be used. Distance-based methods can also be used *in combination* with feature transformation methods. For example, the k-gram representation can be used in combination with distance-based methods in order to construct sequence classifiers.

Hidden Markov Models   These are extremely popular methods, which can be used in order to model either whole sequences or subsequences associated with class labels [14, 423]. In these methods, a Hidden Markov Model is used in order to create a generative model which is specific to each class. The major difference from the methodology discussed in this chapter is that the parameters of the HMM are learned separately for each class. This is done by using only the sequences specific to a particular class in order to train the relevant model. For a given test sequence, the evaluation algorithm is used to determine the identity of the best fit class. In the rare class scenario, the cost-sensitive methods of Chapter 6 can be easily generalized to this case. Specifically, let $q_1 \ldots q_m$ be the probabilities generated by the forward algorithm, when applied to each of the k different HMMs, corresponding to the m different classes. Let the costs for the m different classes be $c_1 \ldots c_m$. Then, the cost-weighted

probability f_i for the ith class is defined as follows:

$$f_i = \frac{c_i \cdot q_i}{\sum_{j=1}^{k} c_j \cdot q_j} \tag{9.5}$$

The class with the highest value of f_i is reported as the relevant one.

Comparative Studies A natural question arises as to how the aforementioned classifiers compare with one another. A detailed performance study, which compares different sequence classifiers is provided in [139]. The results suggest that SVM classifiers can work well, when they are used in combination with the right feature transformation methods. Furthermore, such classifiers can also provide high-levels of interpretability, when the feature space corresponds to k-grams extracted from the data.

6. Conclusions and Summary

Discrete sequences are extremely common in virtually all application domains such as biological data, user-generated data, system diagnosis data and web logs. This provides a massive source of temporal data, which is very rich, and is also very complex from the perspective of outlier analysis. The complexity of sequence data allows for diverse definitions of sequence outliers. The broad classes of models are similar to continuous time series data, though the details of the models are different because of the different format of the data. As in the case of time series data, a single position can be declared as a contextual outlier, or a set of positions may be declared a collective outlier. Contextual outliers are significantly easier to determine in terms of computational complexity and typically use Markovian techniques. This chapter also discusses some of the more common models and techniques for determining collective outliers in sequence data. These are distance-based methods, frequency-based methods, and Hidden Markov Models. Many of these methods can also be generalized to the supervised case, when labels are available.

7. Bibliographic Survey

The problem of sequence outlier detection is useful in the context of a wide variety of applications such as host-based intrusion detection, system fault detection, biological sequences, and web click-streams [284, 315, 399, 419, 440]. A general survey on sequence outlier detection may be found in [108]. Outliers are of two types corresponding to *position outliers* and *combination outliers*. Position outliers correspond to specific events (symbols) in the sequence (string), which may be considered an

outlier. Combination outliers correspond to unusual subsequences of symbols in the sequence, which differ from the overall trends in the data.

Position outliers are computed using either Markovian techniques [332, 488], or rule-based systems [120, 488], which are equivalent. The short memory property [386] can be used in order to predict the value of a sequence from the use of the last k positions. These models can be considered the discrete equivalent of continuous regression-based predictive models discussed in Chapter 8. Methods for using rule-based methods for finding position outliers are discussed in [294, 120]. The use of a suffix-tree to represent all k-order (or less) states for outlier prediction was proposed in [324]. Sparse markov transducers, which are constructed using probabilistic suffix trees are presented in [156]. The simplest Markov Model of the first order is proposed in [488], and a kth order Markovian model is presented in [332]. A discussion of pruning techniques for improving the efficiency of predictive Markovian Models is provided in [140]. Probabilistic suffix trees [434] can be used in order to greatly improve the speed of Markovian techniques. A method for determining outliers based in information theoretic measures was proposed in [259]. Given a sequence, the idea is to segment into two parts, and determine the segment which has the largest MDL (Minimum Description Length). This segment is assumed to be more noisy and contain the outliers. This process is recursively repeated, until the appropriate outlier position in the sequence is determined. The actual determination of the description length is based on the concept of Kolmogorov Complexity.

Combination outliers typically use windowing techniques in which comparison units are extracted from the sequence for the purposes of analysis [163, 222]. Proximity-based methods for outlier detection require the efficient computation of similarity functions in sequence data such as the longest common subsequences [228], and the edit distance [196]. A number of different similarity functions are discussed in [164, 284, 285]. These methods are typically used in conjunction with window-based methods in order to compare short subsequences of the test sequence with the training sequence. Frequency methods provide a more effective way of computing the surprise of a user-specified comparison unit between a training and test sequence. The TARZAN algorithm [257] uses suffix trees in order to compute the surprise level for a user-specified comparison unit. Proximity methods are also used for point-anomaly detection with the use of k-nearest neighbor methods [97], k-medoid based clustering [84, 85], and probabilistic suffix tree-based clustering [481]. Markov Models for the detection of significant episodes and change points in sequences are proposed in [199, 414].

Hidden Markov Models are a popular method for finding combination outliers. A tutorial on Hidden Markov Models may be found in [379]. While they support the direct determination of the determination of the outliers with the use of the underlying fit probabilities, methods are also designed to mine the state transition sequences with window-based methods for anomaly detection [163, 508]. Hidden Markov Models have also been used for point anomaly detection of small sequences [89, 413] by creating a mixture of hidden markov models in order to create the underlying clusters.

Outlier detection is also a challenging problem in the context of complex sequences in which individual points are sets rather than symbols. Most methods for novelty detection in text, categorical and market basket streams can be generalized to this case [26, 96]. The techniques for first-story detection in text can also be applied to set-based sequential data [37, 38, 515].

Supervised methods are used frequently in a variety of applications such as intrusion detection, when labeled data is available [153, 182, 183, 468, 449]. A review of sequence classification methods have be found in [475]. Position classification in sequences is useful for many natural language applications [41, 283], which specific properties of positions in sentences need to be learned. The most common method for sequence classification is use of feature extraction methods such as k-gram extraction and wavelets [12, 299]. Hidden markov models are a natural technique for the problem of classification [14, 423]. An experimental evaluation of methods for sequence classification may be found in [139].

8.   Exercises

1. Consider the discrete sequence denoted by ACGTACTGACGTACG-TATGT.

- Construct an order-1 Markovian Model in order to determine the position outliers. Which position is found as the outlier?

- Now create an order-2 Markovian Model in order to determine the position outliers. Which position is found as the outlier?

2. Determine all order-1 rules from the discrete sequence of Exercise 1. Which position is found as the outlier?

3. For the discrete sequence of Exercise 1, determine all subsequences of length 2. Use a frequency-based approach to assign outlier scores to objects. Which pairwise sequence of length 2 should be considered a combination outlier?

4. Repeat Exercise 2 with sequences of lengths 3, 4, and 5. What happens to the relative outlier scores with increasing subsequence length?

5. Repeat Exercises 3 and 4 with the use of a distance-based approach.

6. Construct a black box 2-state Hidden Markov Model in order to learn the state probabilities of the sequence in Exercise 1. Write a computer program to learn the state transition and symbol emission probabilities. Now apply the model to extracted subsequences of length 2. Which subsequences are predicted as the outliers by the model?

Chapter 10

SPATIAL OUTLIER DETECTION

"Time and space are modes by which we think and not conditions in which we live." – Albert Einstein

1. Introduction

Spatial outliers are objects which have *behavioral* attribute values that are distinct from those of their surrounding *spatial* neighbors. Thus, *spatial continuity* plays an important role in the identification of anomalies. This is an analogous principle to the concept of temporal continuity, which was discussed in the chapters on time series outlier detection. One of the most fundamental rules of spatial data is as follows [455]:
" Everything is related to everything else, but nearby objects are more related than distant objects."

Spatial data does not contain only spatial attributes, just as temporal data does not necessarily contain only temporal attributes. Instead, spatial locations form the *contextual points* at which other behavioral attributes of interest are measured. Thus, two kinds of attributes may be available:

- *Behavioral Attributes:* This is the attribute of interest which is measured for each object. For example, this could correspond to sea surface temperatures, wind speeds, car speeds, disease outbreak numbers, the color of an image pixel, etc. It is possible to have more than one behavioral attribute at a spatial location in given application.

- *Contextual Attributes (Spatial Location):* This is the location of interest at which the behavioral attribute is measured. Typically, this would contain two or three dimensions, when the data is expressed in terms of coordinates. In some cases, the contextual

attributes may be more complex, and may be expressed at the granularity of a *region of interest*, such as a county, zip-code etc. Alternatively, in an imaging application, the contextual attributes may correspond to individual pixels.

Spatial data shares a number of similarities with time-series data, in which one or more properties of interest (behavioral attributes) are measured at a given moment in time (contextual attribute). In fact, in *spatiotemporal* data, the contextual attributes may also contain a temporal component. This can be used to determine important spatiotemporal anomalies (or events) based on the underlying dynamics. For example, the dynamics of behavioral attributes such as humidity, wind speeds, sea surface temperatures and pressure can be used in order to identify and predict anomalous weather events. In such cases, both spatial *and* temporal continuity can play an important role in the prediction. It is also possible for the data to be *purely* spatiotemporal, in which no other behavioral attributes are present, and the trajectories of objects are measured over time. In such cases, no attribute needs to be treated as behavioral, since a joint analysis of both components provides the best insights in many applications. In some cases, it may be helpful to treat the temporal component as the contextual attribute, and the spatial components as the behaviorial attributes. For example, in a two-dimensional *real-time* trajectory mining application, this can be modeled as a bivariate time series, in which the evolving X-coordinate and Y-coordinate values are individual time series. In the *offline* trajectory shape analysis scenario, anomalies may correspond to unusual shapes, irrespective of their temporal provenance. The latter case is mostly a spatial analytics scenario, and the temporal aspects of the problem are limited. Therefore, trajectory-based applications can be modeled in multiple ways, depending upon the needs of the underlying application.

Spatial data is common in many real applications, such as the following:

- *Meteorological Data:* Numerous weather parameters are typically measured at different geographical locations, which may be used in order to predict anomalous weather patterns in the underlying data [510].

- *Traffic Data:* Moving objects may be associated with many parameters such as speed, direction etc. The location of an object is its contextual attribute. In many cases, such data is also spatiotemporal, since it has a temporal component. Finding anomalous behavior of moving objects [83] can provide numerous insights.

- *Earth Science Data:* The land cover types at different spatial locations may be the behavioral attributes. Anomalies in such patterns provide insights about anomalous trends in human activity such as de-forestation or other anomalous vegetation trends [287].

- *Disease Outbreak Data:* Data about disease outbreaks is often aggregated by spatial locations such as zip-code and county. Anomalous trends in such data [465] can provide information about the causality of the outbreaks.

- *Medical Diagnostics:* MRI and PET scans are spatial data in two or three dimensions. The detection of unusual localized regions in such data can help in detecting diseases such as brain tumors, the onset of alzheimer disease, and multiple sclerosis lesions [374, 206, 466, 418].

- *Demographic Data:* Demographic attributes such as age, sex, race, and salary can be used in order to identify demographic anomalies. Such information can be useful for target-marketing applications.

As in the case of temporal data, *abrupt changes in the behavioral attribute, which violate spatial continuity* provide useful information about the underlying contextual anomalies. For example, consider a meteorological application, in which sea surface temperatures and pressure are measured. Unusually high sea surface temperature in a very small localized region is a hot-spot which may be the result of volcanic activity under the surface. Similarly, unusually low or high pressure in a small localized region may suggest the formation of hurricanes or cyclones. In all these cases, spatial continuity is violated by the attribute of interest. Such attributes are often tracked in meteorological applications on a daily basis. In Figure 10.1, a color coded map of the sea surface temperatures on October 1, 2012 from the *NOAA Satellite and Information Service* is illustrated. Unusually high temperature anomalies are illustrated in red, whereas unusually low temperature anomalies are illustrated in blue.

In the context of *spatiotemporal* data, both spatial and temporal continuity is used for the purposes of outlier analysis. For example, a sudden change in the velocity of a few cars in a small localized region may suggest the occurrence of an accident or other anomalous event. Similarly, evolving events such as hurricanes and disease outbreaks are spatiotemporal in nature. Spatio-temporal methods for outlier detection [113, 114] are significantly more challenging because of the additional challenges of modeling the temporal and spatial components jointly.
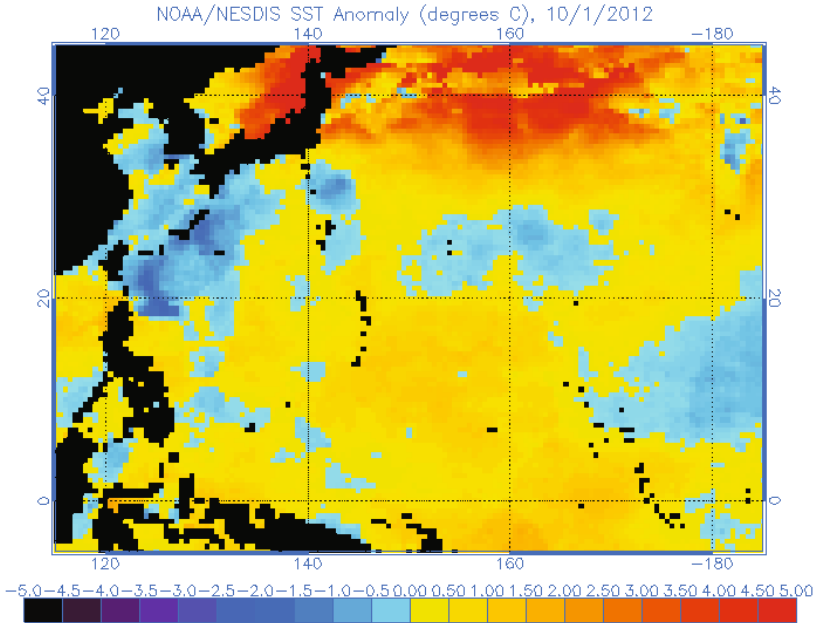
Figure 10.1.   Sea surface temperature anomalies. Source: NOAA Satellite and Information Service

There are two main characteristics of spatial data, which are commonly used in outlier detection algorithms:

- *Spatial Autocorrelations:* This corresponds to the fact that behavioral attribute values in spatial neighborhoods are closely correlated with one another. However, unlike temporal data, where future values of the time-series are unknown, the values in all spatial directions of a data point can be used.

- *Spatial Heteroscedasticity:* This corresponds to the fact the variances of the behavioral attribute depend on spatial location [433].

While the first property is the primary criterion for outlier analysis, the second has also proven to be useful in many scenarios. This is because when certain regions are likely to have greater variance as a matter of expectation, then abrupt changes in those regions are less likely to be significant. Such insights have lead to local methods [433], which are based on ideas derived from local density-based methods (LOF) [78].

Numerous methods have been proposed in the literature for detecting spatial outliers. The primary ones among them use *variations of the behavioral attribute* within a neighborhood in order to define outliers. Such

outliers use either multidimensional analysis methods or graph-based methods. In addition, many of the temporal auto-correlation methods discussed in the previous chapter can also be generalized to the spatial domain, when the data is completely specified over the various dimensions.

Most of the work on spatial outliers is about finding *abrupt changes* which violate spatial auto-correlations. Such outliers are *contextual* outliers. While the standard statistical tests for deviation detection are useful in this case, it is sometimes useful to intuitively visualize the key outlier points. The spatial nature of the data also lends itself to more intuitive visualization methodologies such as visualization. Two examples of such methodologies are *variogram clouds* and *pocket plots* [203, 354]. The former will be described in detail in this chapter.

As in the case of time-series databases, it is also useful to find unusual *shapes of patterns implied by the distribution of the behavioral attribute* in a database of multiple spatial distributions. For example, the color distribution in an image or MRI scan may correspond to an unusual shape, when compared to other images in the database. Such an image may be of interest for further analysis. Such outliers are *collective* outliers in the context of spatial data.

Supervised methods are also very useful in the spatial domain, where it is desirable to determine unusual shapes from multiple spatial patterns. For example, while many conditions such as weather patterns of interest, or brain tumors in MRI scans may be rare on a *relative* basis, a significant amount of training data may be available on an *absolute* basis for modeling purposes. In medical applications, large numbers of pathological examples are sometimes available for modeling purposes. Similarly, many examples of pathological patterns of unusual shapes may be available in meteorological and earth science applications. In such cases, it is useful to utilize supervision for the purposes of outlier detection. Supervised methods are particularly useful in the context of outlier detection in such cases, because of the unusually high complexity of a database containing multiple spatial patterns. Such methods are closely related to topics such as image classification. The topic of image classification is a large area of interest in its own right. While this is beyond the scope of this book, some discussion of related work will be provided in this chapter.

A close relationship exists between temporal and spatial outlier detection, because both methods use concepts of *behavioral attribute continuity with respect to one or more contextual attributes*. The main difference lies in the fact that spatial contextual attributes are often multidimensional, whereas time is a single attribute. Furthermore, time is uni-

directional, where only values in the past are known, whereas spatial attributes are known in the different directions of all axes. Nevertheless, in many applications, these differences are not significant enough to invalidate the applicability of temporal methods. While recent work has adapted temporal techniques to some spatial applications such as anomalous image shape detection [469], many other temporal techniques have the potential for use in the spatial domain. This chapter will point out the different temporal techniques, which are also applicable to the spatial domain. It should be noted that in some cases, these temporal methods are indeed not applicable, especially when the spatial contextual attribute cannot be expressed in terms of a comprehensive set of coordinates in a multidimensional plane. For example, the spatial attribute may be specified with a rough granularity, such as a county or zip-code, or may be available only for a small subset of points in the spatial plane.

This chapter is organized as follows. In the next section, neighborhood-based algorithms for outlier analysis will be studied. Both multidimensional and graph-based methods will be studied in this section. Auto-regressive models for anomaly detection are presented in section 3. Visual methods for detecting spatial outliers with variogram clouds are addressed in section 4. Unusual shape discovery in multidimensional spatial data will be addressed in section 5. Methods for spatiotemporal outlier detection are presented in section 6. The use of supervision for anomaly detection in spatial data is studied in section 7. The conclusions and summary are presented in section 8.

2.   Neighborhood-based Algorithms

Neighborhood-based algorithms can be very useful in the context of a wide variety of tasks. In these algorithms, abrupt changes in the spatial neighborhood of a data point are used in order to diagnose outliers. Such algorithms depend upon the exact way in which the spatial neighborhood is defined, the function used to combine these neighborhood values into an expected value, and the computation of the deviations from the expected values. The neighborhood may be defined in many different ways [3, 268, 317, 401–404], depending upon the nature of the underlying data.

- *Multidimensional Neighborhoods:* In this case, the neighborhoods are defined on the basis of multidimensional distances between data points.

- *Graph-based Neighborhoods:* In this case, the neighborhoods are defined by linkage relationships between spatial objects. Such

neighborhoods may be more useful in cases, where the location of the spatial objects may not correspond to exact coordinates (eg. county or zip code), and graph-representations provide a more general modeling tool.

This section will study method for neighborhood-based outlier detection with the use of multidimensional and graph-based methods.

2.1 Multidimensional Methods

While traditional multidimensional methods can also be used to detect outliers in spatial data, such methods do not distinguish between the contextual attributes and the behavioral attribute. Therefore, such methods are not optimized for outlier detection in spatial data, especially in cases where the outliers are defined on the basis of the behavioral attribute.

Numerous methods have been defined, which use the spatial neighborhood of the data with the use of multidimensional distances on the spatial (contextual) attributes. Thus, the contextual attributes are used for determining the k nearest neighbors, and the deviations on the behavioral attribute values are used in order to predict outliers. A variety of distance functions can be used on the multidimensional spatial data for determination of proximity. The choice of the distance function is important, because it defines the choice of the neighborhood which is used for comparison with the true value. For a given spatial object o, with behavioral attribute value $f(o)$, let $o_1 \ldots o_k$ be its k-nearest neighbors. Then, a variety of methods may be used to compute the predicted value $g(o)$ of the object o. The most straightforward method is the mean:

$$g(o) = \sum_{i=1}^{k} f(o_i)/k$$

Alternatively, $g(o)$ may be computed as the median of the surrounding values of $f(o_i)$, in order to reduce the impact of extreme values. Then, for each data object o, the value of $f(o) - g(o)$ represents a deviation from predicted values. The extreme values among these deviations may be computed using a variety of methods discussed in Chapter 2. These are reported as outliers.

2.1.1 Local Outliers. An observation in [433] is that all local deviations are not equally important from the perspective of outlier analysis. For example, consider the case where the sea-surface temperatures are being measured at different spatial locations. In some spatial

regions, the changes in temperatures may naturally show larger variations than others. Therefore, the same variation cannot be treated with equal importance in all regions. Specifically, the outlier scores in high variance regions need to be suppressed. In such cases, it may be useful to quantify the changes around a data point in a local way. For example, instead of using the value of $f(o) - g(o)$ as discussed above, it is possible to use a normalized value of $\frac{f(o)-g(o)}{L(o)}$, where $L(o)$ represents a *spatially local* quantification of the deviations around o. For example, $L(o)$ could represent the standard deviations of the behavioral attribute values in the spatial neighbors of o.

In practice, a variety of different methods could be used in order to characterize the local deviations around the spatial object o. The work in [433] has also defined a deviation measure $SLOM$ which is based on the LOF methods for defining local spatial outliers. This approach is sensitive to the spatial heteroscedasticity of the data, in which the behavior of the spatial locality is carefully accounted for in constructing the outlier score.

2.2   Graph-based Methods

In graph-based methods, spatial proximity is modeled with the use of links between nodes. Thus, nodes are associated with behavioral attributes, and strong variations in the behavioral attribute across neighboring nodes are recognized as outliers. Graph-based methods are particularly useful when the individual nodes are not associated with point-specific coordinates, but may correspond to regions of arbitrary shape. In such cases, the links between nodes can be modeled on the basis of the neighborhood relationships between the different regions. Graph-based methods define spatial relationships in a more general way, since semantic relationships can also be used to define neighborhoods. For example, two objects could be connected by an edge, if they are in the same *semantic* location such as a building, restaurant, or office. In many applications, the links may be weighted on the basis of the strength of the proximity relationship. For example, consider a disease outbreak application in which the spatial objects correspond to county regions. In such a case, the strength of the links could correspond to the length of the boundary between two regions.

Let S be the set of neighbors of a given node i. Then, the concept of spatial continuity can be used in order to create a *predicted* value of the behavioral attribute based on those of its neighbors. The strength of the links between i and its neighbors can also be used in order to compute the predicted values as either the weighted mean or median on

the behavioral attribute of the k nearest spatial neighbors. For a given spatial object o, with behavioral attribute value $f(o)$, let $o_1 \ldots o_k$ be its k linked neighbors based on the relationship graph. Let the weight of the link (o, o_i) be $w(o, o_i)$. Then, the linkage-based weighted mean may be used to compute the predicted value $g(o)$ of the object o.

$$g(o) = \frac{\sum_{i=1}^{k} w(o, o_i) \cdot f(o_i)}{\sum_{i=1}^{k} w(o, o_i)}$$

Alternatively, the weighted median of the neighbor values may be used for predictive purposes. Since the true value of the behavioral attribute is known, this can be used in order to model the deviations of the behavioral attributes from their predicted values. As in the previous case, the value of $f(o) - g(o)$ represents a deviation from the predicted values. Extreme value analysis can be used on these deviations in order to determine the spatial outliers. This process is identical to what was discussed before for the multidimensional case. As in all outlier analysis algorithms, a variety of extreme-value analysis methods of Chapter 2 can be used on these deviations in order to determine the outliers. The nodes with high values of the normalized deviation may be reported as outliers.

2.3 Handling Multiple Behavioral Attributes

In many cases, multiple behavioral attributes may be associated with the contextual attributes. For example, in a meteorological application, both temperature and pressure values may be available with the spatial attributes. In these cases, the deviations may be computed on each behavioral-attribute, and then these values need to be combined into a single deviation value, which provides the final outlier score. For this purpose, any of the multivariate extreme value analysis methods in section 3 of Chapter 2 may be used. In particular, the work in [112] has proposed the use of the Mahalanobis distance-based method of Chapter 2 for extreme value analysis. However, it is also possible to use other depth-based, or angle-based methods discussed in that chapter in order to determine the underlying outliers.

3. Autoregressive Models

Spatial data shares a number of similarities with temporal data. Both kinds of data measure a behavioral attribute (eg. temperature) with respect to a contextual attribute (eg. space or time). In many scenarios, spatial data is available in the form of coordinates, and the values of the behavioral attribute may be available at *each possible spatial reference*

point in the grid. Such data arises commonly in weather contour maps, images, MRI scans etc. In cases, *where the data is completely specified at most points in the grid*, it is possible to use auto-regressive models in order to determine unusually large deviations in the data, in a way which is completely analogous to the temporal scenario.

Let X_{t_1,t_2} be the value of the behavioral attribute at the spatial location (t_1, t_2). In the temporal auto-regressive model, the predicted value of the behavioral attribute is based on a 1-dimensional window of *past* history of length p (see section 2.1 of Chapter 8). In the 2-dimensional spatial scenario, this can be generalized to a square window of size $(2 \cdot p + 1) \times (2 \cdot p + 1)$, with p coordinates in either direction. More generally, in the case of 3-dimensional spatial data, one can use a cube of size $(2 \cdot p + 1) \times (2 \cdot p + 1) \times (2 \cdot p + 1)$. As in the case of the 1-dimensional auto-regression for temporal data in section 2.1 of Chapter 8, a 2-dimensional model can be defined as follows.

$$X_{t_1,t_2} = \sum_{i=-p}^{p} \sum_{j=-p}^{p} a_{ij} \cdot X_{t_1-i,t_2-j} + c + \epsilon_{t_1,t_2}$$

The value of a_{00} is always set to 0, and is missing from the above summation, since a spatial value cannot be used to predict itself. The values of a_{ij} need to be learned from the underlying training data. Thus, such an equation can be created for each value of (t_1, t_2). When the number of spatial-coordinates available is much larger than $(2 \cdot p + 1) \times (2 \cdot p + 1)$, this is an over-determined system of equations, and can be solved in a similar way with *least-squares regression*, as discussed in the methods of section 2.1 of Chapter 3. Thus, the process of determining the regression coefficients is very similar to the case of temporal data.

In the above system of equations, the value of c is a constant, and the value of ϵ_{t_1,t_2} represents the noise, or the *deviation* from the expected values. Large absolute values of this deviation represent the anomalies in the underlying data. Therefore, the extreme value analysis techniques of Chapter 2 can be used in order to determine those deviations which vary significantly from the norm. These values are assumed to be independent identically distributed random variables, which are drawn from a normal distribution. Thus, the extreme value analysis methods of Chapter 2 can be used in order to detect the anomalies.

The afore-mentioned discussion provides a generalization of Autoregressive (AR) models from temporal to spatial data for illustrative purposes. In practice, it is possible to generalize *all* the regression models (ARMA, ARIMA, PCA) to the spatial scenario, by using the appropriate slice of values from the spatial data. As in the temporal case, it is even possible to create multivariate spatial regression models, where

multiple behavioral attributes are available. Typically such behavioral attributes may be correlated with one another (eg. temperature and humidity), and it is desirable to determine unusually large local deviations with the help of multivariate correlations. Some of these generalizations are presented as exercises for the reader at the end of this chapter.

While the autoregressive nature of spatial data is very widely recognized, such models have rarely been used for anomaly detection in the spatial literature. This is partially a result of the high computational complexity of auto-regressive models with an increasing number of coefficients. Such models also cannot easily handle spatial data which is incompletely specified by spatial location, region-based locations or semantic locations. Nevertheless, such models can be very useful in many scenarios such as image analysis or weather patterns, where large amounts of reasonably complete data are available for analysis. In such cases, the statistical robustness of these methods is likely to be higher than simpler neighborhood-based models.

4. Visualization with Variogram Clouds

A number of visualization techniques such as pocket plots and variogram clouds are used in order to visualize spatial outliers. The former will be discussed here in detail, because of their relative popularity. Since spatial outliers are based on *disagreement* in the continuity of the behavioral attribute *in relation* to the spatial attribute, a natural method to visualize this would be to create a scatter plot between the pairwise spatial distances and the pairwise behavioral attribute (square) deviation. The spatial distance is simply the euclidian distance between a pair of points. The behavioral attribute deviation is defined as the half the square distance between the behavioral attribute values. A scatter plot is created between the spatial distances on the X-axis, and the behavioral square deviations on the Y-axis, for every pair of points in the data set. The idea is that smaller spatial distances will likely correspond to smaller behavioral attribute variances and vice-versa. In particular, large variations of the behavioral attribute for smaller spatial distances should be considered deviants. Such points on the variogram cloud can be traced back to the original data to determine pairs of points which are spatially close, but behaviorally different.

In order to illustrate the impact of outliers on variogram clouds, an example will be used. First, the data set for the variogram clouds of Figure 10.2 will be described. In this case, a grid of 100 points on the spatial plane are used with coordinates drawn from $X, Y, = 0.1, 0.2 \ldots 1.0$. The
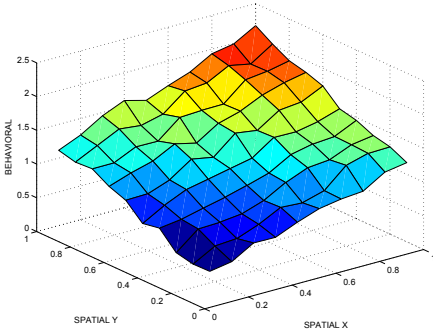
(a) Smooth spatial variation
with no outlier

(b) Variogram Cloud
with no outlier

(c) Smooth spatial variation
with added outlier
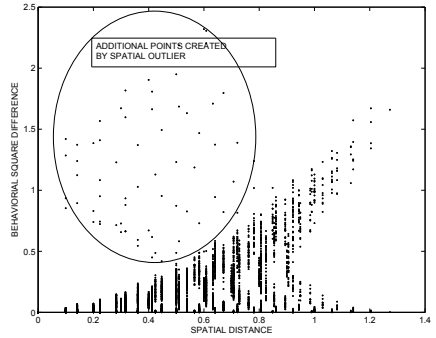
(d) Variogram Cloud
with added outlier

Figure 10.2.   Effect of adding spatial outlier to variogram cloud

value of the behavioral attribute Z was generated as follows:

$$Z = X + Y + \epsilon$$

Here ϵ is a small amount of noise, which was randomly generated from the uniform distribution in $[0, 0.2]$. This spatial variation of the attribute is quite smooth, since the noise is small relative to the global variation in values of the behavioral attribute. The spatial profile of the generated data is illustrated in Figure 10.2(a), and the corresponding variogram cloud is illustrated in Figure 10.2(b). It is evident that low values of the spatial distance always corresponding to low deviations of the behavioral attribute. While it is possible for high spatial deviations to be related to low behavioral deviations, the converse is not true.

Subsequently, a single outlier is added to the data by distorting the behavioral attribute of one of the spatial values in the grid of Figure 10.2(a). The corresponding outlier is shown in Figure 10.2(c), and is marked explicitly. Note that the spatial data sets in Figures 10.2(a) and 10.2(c) are virtually identical, with the only difference between them being the outlier created by a distorted behavioral attribute value. The corresponding variogram cloud is illustrated in Figure 10.2(d). It is evident that in this case, a new set of points have been added to the variogram cloud in which significant behavioral deviations exist even at low spatial distances. Multiple such deviant points are created corresponding to the different data points in the immediate spatial locality of the added outlier. Such points can easily be isolated visually and linked back to the original points in the data. Thus, this approach provides an easy visual and intuitive way to isolate the spatial outliers in the data set.

One challenge of creating a variogram cloud is the high computational complexity. Note that a single point exists in the variogram cloud for each pair of data points in the original data. Therefore, the number of points in the variogram cloud scales quadratically with the number of points in the spatial data. This can make the approach rather slow, when the number of data points is large. In practice, it is difficult to create a variogram plot for situations in which the data contains a few hundred thousand spatial data points. This can be a significant problem, since spatial data sets are often quite large in practice.

One observation about the variogram cloud is that it is not always necessary to represent *every pair* of points on the plot. Data points which are spatially very far away add little insights about the outlier behavior. Therefore, each spatial dimension can be discretized into ranges, and this creates a 2-dimensional grid in the data. The pairwise relationships between all spatial points *within* this grid can be used in order to create

the variogram cloud. This significantly reduces the computational complexity of creating the variogram cloud. For example, consider the case, where the original data set contains N points, which are discretized into a $t \times t$ grid with approximately[1] N/t^2 data points in each. Then, the computational complexity of creating a variogram cloud for each grid is $O(N^2/t^4)$. Of course, since there are a total of t^2 grids, the aggregate computational complexity is $O(N^2/t^2) < O(N^2)$. This provides a speedup factor of $O(t^2)$. Of course, in this case an optimistic scenario was assumed where the data points were uniformly distributed into the grid structure. It can be shown theoretically that a speedup factor of at least t can be obtained with this approach. This is because the speed up achieved with a grid partitioning into $t \times t$ ranges will always be better than the discretization along only one dimension into t ranges with an equal number of data points. A significant speedup may be obtained even for modest values of t, without significant reduction in the quality of the visual discrimination between the outliers and the normal points.

5.   Finding Abnormal Shapes in Spatial Data

The problem of finding unusual shapes in spatial data finds numerous applications such as image analysis. For example, the detection of unusual shapes from brain PET scans or MRI scans can help detect conditions such as tumors, alzheimer and sclerosis [374, 466], or can help identify anomalous conditions such as hurricanes from weather maps. For example, consider the satellite image illustrated in Figure 10.3. The anomalous shape in the image corresponds to hurricane *Fran*, which was a large destructive hurricane, which hit Cape Fear in North Carolina on September 1996. The hurricane can easily be identified by its characteristic shape in the satellite image. However, such a shape may not appear in other similar satellite images on normal days, and is therefore an unusual event. Another example from the medical domain is illustrated in Figure 10.4, where the PET scans from a normal person and an alzheimer patient are presented. The colored regions correspond to the uptake of the radioactive tracer administered in a PET scan (behavioral attribute). It is evident that this behavioral attribute shows very different spatial behavior for normal and diseased individuals.

In their simplest form, shapes can be modeled by the contours (or boundaries) of regions with particular ranges of behavioral attribute val-

---

[1]In practice, the different grid regions may contain a different number of data points because of spatial correlations. However, in many applications such as image data, pixels may be available for every spatial coordinate. Therefore, the division into grids will create a uniform division of the data points.

Figure 10.3. NASA Satellite Image of Hurricane *Fran*: The anomalous shape is characteristic of a hurricane
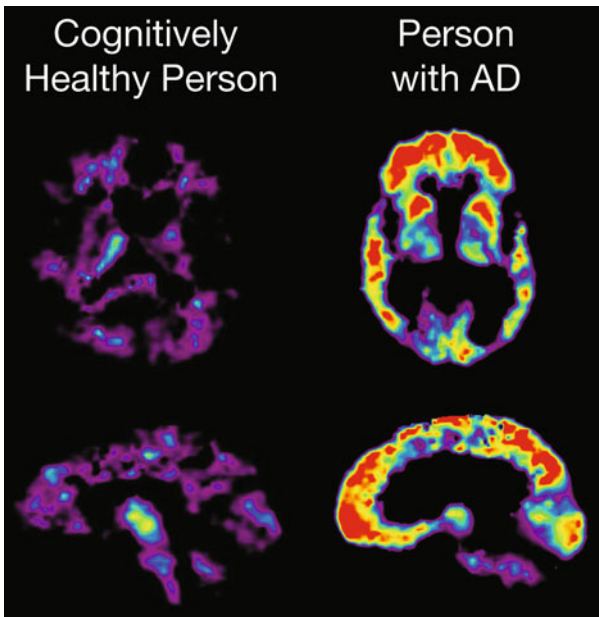
Figure 10.4. PET Scans of brain for cognitively healthy person versus an alzheimer patient: Image courtesy of the National Institute on Aging/National Institutes of Health

ues in the data. For example, in the case of Figure 10.3, the boundaries of such regions can be extracted by direct analysis of sensor and satellite readings such as pressure, cloud cover, temperature, wind speed, and humidity or from the (already processed) color histogram of the corresponding image.

A key simplification for shape analysis is that the contours of an object can be represented as a synthetic time-series. One possible way to achieve this is to use the distance from the centroid of the object to the boundary of the object, and compute a sequence of real numbers derived in a clockwise sweep of the boundary [504]. This yields a time series of real numbers, and is referred to as the *centroid distance signature*. This transformation can be used to map the problem of mining shapes to that of mining time-series, a domain which is much more easier to address from an analytical perspective. For example, consider the elliptical shape illustrated in Figure 10.5(a) with centroid denoted by X. Then, the time-series representing the distance from the centroid, by using 360 different equally spaced angular samples, is illustrated in Figure 10.5(b). In this case, the sample points are started at one of the major axes of the ellipse. If the sample point starts at a different place, or if the shape is rotated (with the same angular starting point), then this causes a cyclic translation of the time-series. The resulting time-series may be normalized in different ways depending upon the needs of the application:

- If no normalization is performed, then the outlier analysis approach is sensitive to the absolute sizes of the underlying objects. This may be the case in many medical images such as MRI scans, in which all spatial objects are drawn to the same scale.

- If all time series values are multiplicatively scaled down by the same factor to unit mean, then such an approach will allow the matching of shapes of different sizes, but will discriminate between different levels of relative variations in the shapes. For example, two ellipses with very different ratios of the major and minor axes will be discriminated well.

- If all time series are translated to zero mean and multiplicatively scaled to unit variance (as is normally done for time-series analysis), then such an approach will match shapes where *relative* local variations in the shape are similar, but the overall shape may be quite different. For example, such an approach will not discriminate very well between two ellipses with very different ratios of the major and minor axes, but will discriminate between two such shapes with different relative local deviations in the boundaries.

(a) Elliptical Shape   (b) Distance from Centroid
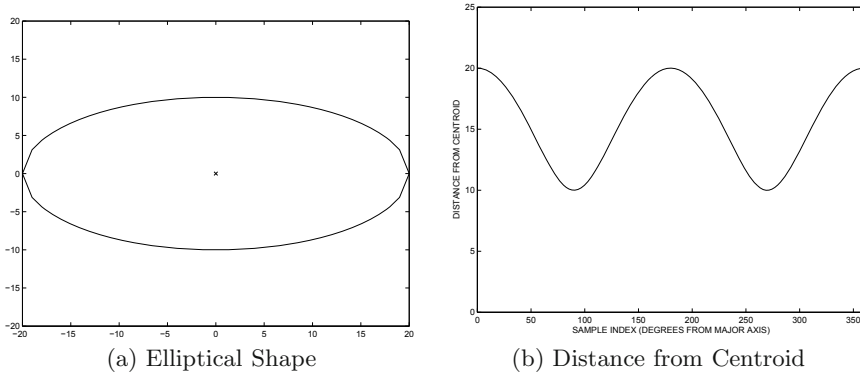
Figure 10.5.   Conversion from shapes to time-series

The only exception is a circular shape, which appears as a straight line. Furthermore, noise effects in the contour will be differentially enhanced in shapes which are less elongated. For example, for two ellipses with similar noisy deviations at the boundaries, but different levels of elongation (major to minor axis ratio), the overall shape of the time-series will be similar, but the local noisy deviations in the extracted time series will be *differentially* suppressed in the elongated shape. This can sometimes provide a distorted picture from the perspective of shape analysis. A perfectly circular shape may show unstable and large noisy deviations in the extracted time-series because of image rasterization effects. The solution proposed in [469] is to treat circular shapes specially, though the unintended effects of such normalization may have unusually complex effects across a broader spectrum of shapes.

In general, it may be advisable to pick the normalization method in an application-specific way.

The problem of shape analysis is further complicated by the effect that transformations such as rotations can have on the underlying data. For example, consider the images illustrated in Figure 10.6. All images correspond to the same object, but two of them are rotated with respect to the original shape, and the last is a mirror image of the original shape. It is clear that the rotation makes it much more difficult to match the two images, if the time-series representation does not account for the rotation or the mirror image effects of the representation. Errors in matching the two shapes also lead to errors in outlier detection, especially when the outlier detection process uses a proximity-based method. It is important to note that all applications do not necessarily require the accounting of rotations. For example, in an MRI scan, where the correct orientation of

the scan is known, such rotational transformations may not be needed. However, in the following, the most general case, which accounts for rotations will be discussed.

An immediate observation is that *a rotation of the shape leads to a linear cyclic shifting of the time series generated by using the distances of the centroid of the shape to the contours of the shape.* For a time series of length n denoted by $a_1 a_2 \ldots a_n$, a cyclic translation by i units leads to the time series $a_{i+1} a_{i+2} \ldots a_n a_1 a_2 \ldots a_i$. Then, the *rotation invariant euclidian distance* $RIDist(T_1, T_2)$ between two time series $T_1 = a_1 \ldots a_n$ and $T_2 = b_1 \ldots b_n$ is given by the minimum distance between T_1 and all possible rotational translations of T_2 (or vice-versa). Therefore, the following is true:

$$RIDist(T_1, T_2) = \min_{i=1}^{n} \sum_{j=1}^{n} (a_j - b_{1+(j+i) \bmod n})^2$$

Note that the reversal of a time-series corresponds to the mirror-image of the underlying shape. Therefore, mirror images can also be addressed by using this approach.

The shape discords can then be determined by computing the series whose kth nearest neighbor distance to its closest neighbor is as large as possible. The top n such shapes need to be found. As in all distance-based algorithms, a brute-force approach on a database with N shapes would require $O(N^2)$ distance computations, unless pruning methods are used.

The major difference between this problem and the unusual time-series shape discovery problem discussed in section 3 of Chapter 8 is that the rotational invariant distances are used instead of the euclidian distances. Furthermore, the distances are computed on whole time-series instead of on subsequences. While it may be possible in theory to use the method of Chapter 8, by making some modifications to address rotational invariance, longer lengths of whole sequences (compared to subsequences), may cause greater challenges in pruning. For example, rotational variations can be addressed by explicitly incorporating rotational variations of the time-series into the database, just as subsequences of a time-series are incorporated into the database for subsequence discord discovery in section 3 of Chapter 8. Care needs to be taken in avoiding self-similarity from the same shape during the distance computations, just as self-similarity is avoided in time series discord discovery. Therefore, the techniques in section 3 of Chapter 8 can be used in theory in order to find discords. Of course, the addition of multiple rotational variations of the shapes to the database is likely to slow down the discovery process. It also leads to some redundancy in the representation,
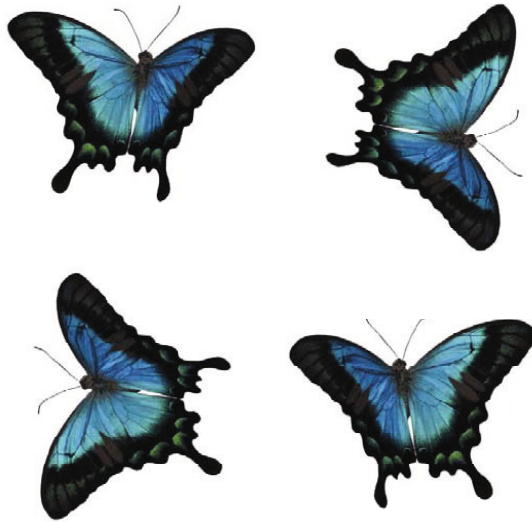
Figure 10.6. Rotation and mirror-image effects on shape matching for outlier analysis

because all rotational variations of the same object will have the same outlier score.

The work in [469] uses a different pruning method based on LSH-approximations [230] of the symbolic aggregate approximations of the time-series. The overall organization of the approach is similar to the algorithm discussed in section 3 of Chapter 8. Both methods first sort the objects by approximate outlier tendency in order to perform the outlier search in an ordered way, which optimizes the pruning behavior. For each object, pruning is performed with approximate nearest neighbor distances. However, the specific technique used for pruning is different in the two scenarios.

A nested loop approach is used to implement the method. The algorithm examines the candidate shapes iteratively in an outer loop, and progressively improves the estimate of each candidate's k-nearest neighbor distance in an inner loop. The inner loop essentially computes the distances of the other shapes to the candidate. At the end of the execution of a candidate-specific inner loop, the approach then either includes the candidate in the current set of top-n outlier score estimates, or discards the candidate at some point during the computation of its k-nearest neighbor in the inner loop. This is referred to as early inner loop termination. This inner loop can be terminated early, when the currently approximated k-nearest neighbor distance for that candidate shape is less than the score for the nth best outlier found so far. Clearly,

such a shape cannot be an outlier. In order to obtain the best pruning results, the candidate shapes in the outer loop need to be heuristically ordered, so that the earliest shapes examined have the greatest tendency to be outliers. The pruning performance is also best, when the points are ordered in the inner loop, such that the k-nearest neighbors of the candidate shape are found early. It remains to explain how the heuristic orderings required for good pruning are achieved.

As in the case of time-series subsequences, each time series is mapped onto an LSH word with the use of Symbolic Aggregate Approximation. Assume that the resulting SAX words have length m. Locality sensitive hashing [230] randomly samples $r < m$ distinct positions in the SAX representation. Therefore, two SAX words which are more similar are more likely to map to the same string. This is also referred to as the *locality sensitivity property* of the LSH-hashing approach, and the similarity can be robustly quantified by examining the mapping behavior over multiple hash functions. However, this does not account for the rotational invariance of the matching process. In order to account for the possible rotations, a rotational invariant LSH function is defined. This function first picks $r < m$ position indices randomly, and then samples these r position indices from all possible m rotations of the SAX word. Clearly, similar shapes will lead to LSH-based collisions, even in the presence of rotations. The LSH-hashing process is repeated with multiple hash functions in order to provide greater robustness to the collision-based counts.

For each SAX word, a count is maintained of its number of LSH-based collisions. This provides approximate information about its outlier score. Shapes with smaller counts need to be processed first as candidates in the outer loop, since they have greater likelihood of being outliers. Furthermore, shapes which collide with one another frequently in LSH-based hashing are more likely to be nearest neighbors. Therefore, shapes which have the largest number of collisions with the current outer loop candidate are examined first in the inner loop for distance computations. This provides the heuristic order of processing in the inner loop. The reader is referred to [469] for a detailed description of the algorithm.

6.   Spatio-temporal Outliers

Spatio-temporal data is very common in many real applications in which behavioral attribute values are continuously tracked at different spatial locations. For example, consider a chemical factory dumping chemicals in a river. In such cases, the concentrations of chemicals in the water cannot be described by using either only spatial or temporal

contextual attributes. Thus, the contextual attributes need to contain *both* spatial and temporal components. Spatiotemporal data is extremely common in all forms of sensor data, in which behavioral attribute readings are continuously transmitted by sensors at different spatial locations. An example is provided in [472], where precipitation data from different spatial locations and times is aggregated. It is desirable to determine localized spatial regions which are also close together in time, whose precipitation values are significantly different from their "neighboring" values. So how should neighboring values be defined in the case of spatiotemporal data?

Virtually all the spatial methods discussed in earlier sections of this chapter can be generalized to spatiotemporal data, as long as the concept of neighborhood is properly defined in order to make it relevant for the spatiotemporal scenario:

- Spatial methods can be used on temporal snapshots of the data in order to determine the relevant outliers at different instants. However, such an approach is incomplete, because it fails to identify violations of temporal continuity.

- Some algorithms have been proposed in order to separately identify spatial outliers and temporal outliers, and then combining the results in order to provide the spatiotemporal outliers [71]. However, the decoupling of spatial and temporal aspects of the problem at an earlier stage is obviously a sub-optimal solution.

- Spatio-temporal neighborhoods of data points may be used in order to determine predicted values. Thus, the only difference from purely spatial methods, is that the expanded set of contextual attributes are now used in order to define the neighborhoods for analysis and prediction. As in the previous case, deviations from the predicted values can be used in order to determine outliers. In some techniques such as neighborhood methods, the challenge is to combine the (contextual) distances along the spatial and temporal dimensions in a meaningful way. One simple way of achieving this would be to normalize the standard deviation across each of the contextual attributes to one unit before computation of distances. If desired, weights can be used in order to provide more importance to one or more of the contextual attributes.

The last of the above methods is the most general, because it can detect significant changes *both* across spatial and temporal attributes in an integrated and meaningful way. It is also important to note that spatial and temporal continuity may not be equally important, depending upon

the underlying application. For example, in an application where precipitation level is the behavioral attribute [472], spatial continuity may be slightly more important than temporal continuity. In such cases, appropriate scaling can be performed on the different dimensions, in order to define neighborhoods in a way which provides greater importance to one or more contextual attributes.

6.1   Spatiotemporal Data: Trajectories

A special case of spatiotemporal data is one in which no behavioral attributes are present, and the data comprises a set of moving object trajectories. Such data can be treated as a form of bivariate temporal data, by treating the X-coordinates and Y-coordinates of each object as the behavioral attributes, and time as the only contextual attribute. This results in two related time series at the same instants. Thus, the methods for temporal data analysis can be applied very effectively to such cases. Such analysis, when applied to single time-series, can identify sudden *changes* in trajectory directions and velocity. This can be very useful in detecting information about significant changes in cyclone or hurricane trajectories [94]. In other cases, a database of multiple trajectories may be available, and it is desirable to determine unusual shapes of trajectories. The temporal component is less important in this case, since the trajectories may have been created at different times. In such cases, it is possible to use subsequence analysis on these time-series in order to determine those trajectories which behave very differently from the remaining series by determining time-series of unusual shapes [304]. However, unlike the univariate scenario [304], spatial time-series are at least bivariate, and it is much harder to find unusual shapes in terms of the *combination behavior of the two time series*.

For the first case of real-time change analysis, the prediction-based outlier detection methods discussed in section 2 of Chapter 9 can be applied separately on each of the X-coordinate and Y-coordinate time series. This results in a residual value along each of the two coordinates. If each of these residuals is modeled as a normal distribution, then the sum of the squares of the Z-values of these residuals is a χ^2 distribution with two degrees of freedom. This can provide an outlier score, along with a corresponding probability value.

While real-time *change* analysis of such scenarios can be performed more effectively by using temporal modeling, *unusual shape* detection of trajectories can be best performed by abstracting out the temporal component, and performing the spatial analysis directly on the trajectories. In such cases, each spatial object has a shape, and the difference of this

shape to its nearest neighbor trajectories are used in order to determine outliers. Since such trajectories may contain a large number of time-stamps, it may often be difficult to determine outliers on the entire sets of trajectories. In such cases, unusual subsequences of the trajectories may be used in order to identify outliers. This case is similar to that of identifying unusual shapes in images, which is discussed in section 5 of this chapter.

Some specific methods such as TROAD have also been proposed in the literature [292] for unusual shape detection in trajectories. In particular, the partition-and-detect framework [292] first partitions the trajectories into a set of sub-trajectories. Note that this is somewhat analogous to the concept of partitioning time series into subsequences (or finding outliers in subspaces of numerical data), since outliers cannot easily be determined on the full series (with high implicit dimensionality). The sub-trajectories are created with a two-level partitioning which is allowed to be coarse-grained at the higher levels, and fine-grained at the lower level. Subsequently, those sub-trajectories, which are not similar to other ones in the data are reported as the outliers. The similarity is measured with the use of both distance-based and density-based methods. Note that the choice of the distance function is critical, and can regulate the nature of the outlier found. For example, a distance function which is sensitive to the *location* of the trajectory is likely to find an outlier based on location of the trajectories. On the other hand, a distance function which is sensitive to the angle between trajectory segments is likely to be sensitive to directions of movement. The precise definition of the distance function is application dependent, though a variety of such functions can be used in conjunction with the partitioned set of sub-trajectories.

The work in [292] defines a t-partition as a line segment from the trajectory. Intuitively, this can be considered analogous to comparison-unit schemes discussed in Chapter 9, which are used in the context of sequence data. A t-partition is said to be outlying using the variation[2] of the k-nearest neighbor distance definition, first proposed in [261]. Intuitively, a t-partition is considered an outlier, if a sufficient number of trajectories in the database are not close to it. The definition of closeness is based on measuring the portion of the trajectory, which is close to the t-partition. As in comparison-unit schemes for discrete sequences, the results from the different "units" (or partitions) are combined together to declare a trajectory as an outlier, if a sufficient number of its

---

[2]That variation fixes the nearest neighbor distance, and computes the required value of k rather than the other way around.

partitions are also outlying. Furthermore, the locality sensitive density-based approach of [78] has also been generalized to this case, by creating a density-sensitive outlier score for the trajectories.

6.2   Anomalous Shape Change Detection

In spatial data such as weather data, PET scans, and MRI scans, unusual changes in the contours of the shapes may be used in order to predict anomalous events. For example, the formation of a hurricane or a tumor over multiple time stamps will show up as an unusual change in the shapes of the corresponding image representations of the weather data or the MRI scan. The determination of such changes is more complex than those of detecting unusual *point changes* in the data. However, the detection of unusual point changes can be a first step towards detecting regions of anomalous change in the data, by clustering the change points in the spatial data. Not all regions of change may necessarily correspond to anomalies. For example, increasing age may create certain characteristic change contours in an PET scan, which should be considered normal. In practice, this problem is not very different from finding unusual shapes in the original data, with the main difference being that the contours of the shapes are constructed on the basis of the changes in the behavioral attributes between two snapshots. The normally occurring changes in the data over time will usually be quite different from the anomalous changes. Therefore, a differencing operation on two temporal snapshots of the data may be required as a pre-processing step, before applying outlier analysis algorithms. A detailed description of many such change analysis methods may be found in [92].

7.   Supervised Outlier Detection

In many applications, a significant amount of training data may be available in order to determine anomalies. Such supervision could occur in either spatial data (with contextual attributes and behavioral attributes), or spatiotemporal data such as trajectory data. In all cases, supervision can be used in order to greatly enhance the effectiveness of the outlier analysis process.

7.1   Supervised Shape Discovery

Spatial data is particularly common in many forms of image data such as weather maps, PET scans or MRI scans. For example, consider the case of MRI scans, where 3-dimensional images of the brain may be available for analysis. The anomalies in the data such as tumors and lesions may show up as characteristic regions in the data, which are

rare but are nevertheless indicative of specific kinds of abnormalities. In such cases, previous examples of anomalous and normal scans may be available for the purposes of training. While unsupervised anomaly detection can help outlier analysis up to a point, the use of supervision can increase the sophistication of the analysis by revealing specific *kinds* of abnormalities. In most applications, at least semi-supervision is used, where examples of normal spatial profiles are available for analysis. The collection of normal examples is typically not very difficult in most application-specific scenarios, since copious examples of normal instances are usually available.

For all forms of shape classification, the actual *representation* of the shape is the most important step. For example, the *centroid distance signature* discussed in this chapter [504] is one possible way of representing the shapes, but by no means the only one. A thorough review of shape representation techniques may be found in [504]. The shape to time-series transformation discussed in section 5 of this chapter can be used in order to transform the shape classification problem to the time-series classification problem. Any of a number of methods (such as subsequence-based k-nearest neighbor methods) can be used for time-series classification in this case. Numerous methods for time-series classification may be found in the literature [343, 490]. These methods typically try to determine discriminative shapes of the series (or shapelets) which distinguish the normal and abnormal series. In the context of spatial data, such abnormal series are typically derived from abnormal shapes from a spatial perspective. In the *semi-supervised* case, the distances of the test series to examples of normal profiles can be used in order to create outlier scores for the underlying series. The only distinction from the available methods for time-series analysis is that care must be taken in order to account for different rotational variants of the shape in particular application-specific scenarios.

The problem of supervised classification of unusual shapes is also closely related to the problem of detecting and recognizing specific shapes in images. This problem has been studied extensively in the field of computer vision and image analysis. The problem of supervised shape recognition is an important area of research in its own right, and is beyond the scope of this book. The reader is referred to [54, 92, 316, 504] for a detailed description of such methods for image classification, analysis and change detection in the image domain. The major modification to these methods is the incorporation of rare class detection and cost-sensitive methods into these algorithms, using the methods of Chapter 6. Since many of the algorithms discussed in Chapter 6 are meta-algorithms, they

can be used in conjunction with any of the classification techniques in the literature.

7.2   Supervised Trajectory Discovery

In many cases, supervision may be available in the form of labels associated with trajectories. For example, consider a case where the trajectories of a large number of ships are available, and it is desirable to identify the suspicious ones based on their trajectory patterns. In some cases, previous examples of anomalous trajectories may be available. These can be used in order to detect significant anomalous patterns in the underlying data. This is a homogeneous attribute scenario, since the unusual shapes are based purely on the spatial and temporal attributes, rather than on a behavioral attribute.

The ROAM method [300] uses a discrete *symbolic* approximation of the trajectories, which converts the numerical coordinate sequence into a symbolic sequence based on the directions of movement and significant changes in this direction. For example, motifs could correspond to *right-turn*, *u-turn* or *loop*. Every movement pattern can be described as a sequence of these primitive movement patterns. The important motifs can be mined directly from the data by using a clustering approach. If desired, additional meta-attributes may be associated with the symbols corresponding to characteristics of the movement such as the speed. This is however different from the concept of behavioral attributes, since these attributes do not play the behavioral role in the learning process.

Once the discrete representation has been created, the sequences together with their labels can be fed to any sequence-based classifier, which identifies how different sequences are related to the class labels. While the ROAM method was applied in the context of supervised models, it is important to note that the feature transformation used in this work can also be used in the context of unsupervised scenarios.

8.   Conclusions and Summary

The problem of spatial outlier detection arises in many domains such as demographic analysis, disease outbreaks, image analysis, and medical diagnostics. Spatial outlier detection shares significant resemblance with temporal outlier detection in terms of the effects of contextual attributes on the continuity of the behavioral attributes. Therefore, a number of methods in the temporal domain can be used for outlier detection in the spatial domain. Spatio-temporal outlier detection is even more complex and challenging, since it combines spatial and temporal characteristics effectively for outlier analysis.

Spatial data can often be treated as an abstraction of image data, when the spatial data is specified in a complete way. In such cases, numerous methods for image analysis can be used for outlier detection. In fact, in many applications such as MRI scans and weather maps, such data are indeed expressed as images. The analysis of such data involves the determination of unusual shapes from the distribution of the spatial attributes. Such analysis can be performed both in the unsupervised and supervised scenarios.

9. Bibliographic Survey

The problem of finding spatial outliers is different from that in multidimensional data because of the different kinds of attributes which are present in spatial data. The most common kinds of methods for finding spatial outliers use changes in the spatial proximity in order to determine outliers [3, 268, 317, 401–404]. Spatial proximity can be defined either with the use of multidimensional distances, or graph-based distances. Spatial distances are more relevant when the contextual attributes are expressed in terms of coordinates. On the other hand, when the reference attributes correspond to spatial regions or semantic locations, graph-based methods are more relevant, since distances and proximity can be expressed as general functions across links. A random walk approach to determine free form spatial scan windows is discussed in [234]. The application of outlier detection to heterogeneous neighborhoods is discussed in [235]. The work in [473] introduces a spatial likelihood ratio test in order to determine local grid regions in which the variation of the behavioral attribute is different from the remaining data in a statistically significant way. Furthermore, such methods can also be used in the context of multiple behavioral attributes [112]. Spatial data also shows local heterogeneity because of different levels of variance in different parts of the data. Therefore, a local method for spatial outlier detection was proposed in [433].

The standard auto-regressive models for temporal data [387] can be extended to spatial data, when the behavioral attribute values are completely specified over all the different reference values. This is often the case with many forms of image data. The problem of unusual shape detection in images is an important one from the perspective of outlier analysis. Some recent work [469] has been performed on finding unusual shapes in images in an efficient way. Supervised methods for shape detection and change analysis are also widely available in the literature [54, 92, 316, 504]. The work in [206] uses Multivariate Gaussian Markov Random Fields in order to find unusual shapes in medical image data.

Spatial data is closely related to temporal data in the context of the continuity shown by the behavioral attributes. Numerous methods for auto-regressive modeling [387] can also be generalized to the case of spatial data. A significant amount of data in spatial domains also has a temporal component, when the attributes are tracked at multiple time-stamps. This requires methods for spatiotemporal outlier detection [113, 114]. An application of spatiotemporal outlier detection to precipitation data is discussed in [472]. A method for detecting flow anomalies in the context of sensors which located upstream or downstream from one another is discussed in [251]. When the differences in the values of the sensors exceeds a given threshold, it is flagged as a spatiotemporal anomaly. A method for explicitly quantifying the level of local change in a spatiotemporal data stream is proposed in [16]. This method also has the ability to perform online processing, and is discussed in detail in Chapter 8. Methods for detecting anomalies in vegetation data with the use of Principal Component Analysis (PCA) are discussed in [287].

The detection of outliers in trajectories can be modeled either spatially or temporally. Therefore, both spatial and temporal methods are relevant to this case. Significant *changes* in trajectory directions is useful for many applications such as hurricane tracking [94]. In such cases, the trajectory can be treated as bivariate temporal data, and change analysis can be applied to this representation. For this purpose, the prediction-based deviation detection techniques of the previous chapter can be helpful. The works in [83, 181] determine anomalies in moving object streams in real time, by examining patterns of evolution. On the other hand, the detection of anomalous trajectory *shapes* is a very different problem. The earliest methods for trajectory shape outlier detection were proposed in [263]. However, this method transforms the trajectories into point data by using a set of features describing meta-information about the trajectories. Unsupervised methods for trajectory outlier detection, which actually use the sequence information explicitly were first investigated in [344, 292]. The work in [344] uses the fourier transform in order to represent the trajectories in terms of the leading coefficients, and find anomalies. In the second method [292], trajectories are divided into different line segments and anomalous patterns are identified in order to determine outliers. Supervised methods for anomaly detection in trajectory data may be found in [300]. These methods transform the data into discrete sequences, and a classifier is learned in order to relate the trajectories to the class labels. Another method proposed in [302] proposes methods for finding outliers in vehicle traffic data. However, these methods are not designed for determining outliers on individual

objects, but are designed for finding anomalous traffic regions (or road segments) on the basis of aggregate spatial traffic characteristics.

10.   Exercises

1. Construct the closed form solution to the AR regression model proposed in this chapter. Use the methods proposed in Chapter 3 for this purpose.

2. Construct PCA models for relating multiple behavioral attributes at the same spatial location. Use analogous models to those discussed in Chapter 8 for this purpose.

3. Construct PCA models for relating multiple behavioral attribute values over spatially local slices of size $p \times p$. Use analogous spatial models to the time-series models proposed in Chapter 8 for this purpose.

4. What is the time complexity of the methods proposed in Exercises 2 and 3.

5. Create a generalization of the time-series shape detection algorithm discussed in section 3 of Chapter 8 [258] to the spatial shape detection scenario. Refer to the details in [258] for specific details of pruning based on Symbolic Aggregate Approximation.

6. Implement the algorithm developed in Exercise 6 using a C++ implementation. Test it over benchmark data sets discussed in [469].

7. Implement the algorithm discussed in this chapter for unusual shape detection. Refer to [469] for specific details of LSH-based pruning. Test it over benchmark data sets discussed in [469]. How does the speed compare to the algorithm developed in Exercise 7.

Chapter 11

OUTLIER DETECTION IN GRAPHS AND NETWORKS

"In nature, we never see anything isolated, but everything in connection with something else which is before it, beside it, under it and over it."– Johann Wolfgang von Goethe

1.   Introduction

Graphs represent one of the most powerful and general forms of data representation, which can express diverse data, ranging from multi-dimensional entity-relation graphs, the web, social networks, communication networks, and biological and chemical compounds. Broadly speaking, two kinds of graphs arise often in real domains:

- The data may contain many small graphs, drawn over a small base domain of labeled nodes. Some examples of this scenario include chemical and biological compounds. The labels correspond to the chemical elements. Individual graph objects are defined as outliers based on the model of normal graph objects in the database.

- The data may be represented as a single large graph. Examples include the web, social and information networks. In some cases, such as the web and social networks, the nodes may correspond to distinct identifiers such as URLs, actors, or IP addresses. In other cases, the node identifiers are not unique. For example, if the nodes on the web are annotated by their subject category rather than their URL, the node labels are not unique. In other cases, no node labels may be available at all. These scenarios are somewhat different, which may allow the definition of different kinds of node, linkage or subgraph outliers.

In addition, other cases exist, which may be closer to one of the afore-mentioned scenarios. For example, multiple small graphs may be extracted from a larger network. An example of such a scenario is a bibliographic network, in which a publication may be represented as a small graph over a larger bibliographic co-authorship network. In this problem, individual small graphs may be defined as outliers based on their linkage relationships. As discussed in section 3.2.1, the methods for outlier analysis in a single large graph can be easily generalized to this case.

These different kinds of data require different methods for outlier analysis. For example, in the case of small graphs, a single object may be represented as an outlier. However, in large graphs, the outliers are defined as portions of the network, which may be nodes, edges or even subgraphs.

In temporal graphs, the structure of the network may change over time. Such scenarios typically occur in large scale web, social or communication networks, and therefore belong to the second or third cases discussed above. Outliers in temporal data may correspond to significant *changes* in specific structural aspects of the network such as the communities, shortest paths, or other local structural properties. Temporal graphs represent one of the most challenging cases for outlier analysis, because of the many different ways in which outliers may be defined. The following phrase from Chapter 1 is restated here: "*The more complex the data, the more the analyst has to make prior inferences of what is considered normal for modeling purposes.*" For example, in a temporal network, an outlier node could be a node with unusually high degree, unusual connectivity structure, unusually *changing* degree, unusually *changing* community structure, unusually *changing* distances to other nodes, or unusual relationships of node content to linkage structure. There are virtually an unlimited number of ways that outliers could be defined. Even within the context of a specific outlier type such as a node outlier, the appropriate model of regularity could be based on its degree, neighbor set, edge weight distribution etc. Therefore, it is somewhat perplexing in an unsupervised application, how one should define an outlier.

In such scenarios, it is important to define change analysis and outlier detection problems from an application-specific perspective, since there is no uniform definition of an outlier. Specific applications may provide better guidance about outliers. For example, in a spam detection application, the degree distributions of nodes can provide insights about outliers. In a network de-noising application, the linkage connectivity structure can be used to determine outlier links. Therefore, application-

specific guidance is critical in defining outliers meaningfully. This can be considered a mild form of supervision, since domain specific knowledge is encoded into the outlier analysis process. This chapter will focus on a few common definitions of outliers in network data because of their utility in a number of network-centric scenarios.

Specifically, the following aspects of outlier detection in graphs and networks will be studied:

- The problem of outlier detection in many small graphs will be introduced. Such graphs occur commonly in the domain of chemical data, and are therefore useful in determining unusual structural combinations. Because such graphs are small, this case is an instantiation of point-anomaly detection.

- Different models for outlier analysis will be studied in the context of a single large graph. These cases arise commonly in the context of the web, communication networks, and social networks. However, the complexity of a large network allows the definition of outliers in a variety of interesting ways.

- The issue of temporal outlier detection in graphs will be studied. These correspond to significant local and global structural changes in graphs, which are abrupt and unexpected.

Different variations of the aforementioned scenarios will be addressed, such as the incorporation of node content or domain knowledge into the outlier analysis process.

This chapter is organized as follows. Section 2 addresses the problem of outlier detection in many small graphs. Section 3 discusses the problem of outlier detection on a single large graph. The case of many small graphs super-imposed on a single large graph will also be discussed within this section. Methods for incorporating node content in outlier analysis are discussed in section 4. The problem of change analysis and outlier detection in temporal graphs is discussed in section 5. The conclusions and summary are presented in section 6.

2.   Outlier Detection in Many Small Graphs

An interesting case for outlier analysis arises when the data contains many small graphs. Some examples include chemical compounds, biological networks, XML graphs, RFID graphs, and entity-relation graphs. In some of these domains, the graphs could become quite large, though in other specific cases, the graphs could be of relatively modest size. A complicating factor in these domains is that the labels on nodes may

typically be repeated, as a result of which the matching can be performed in a variety of different ways. This is referred to as *the challenge of isomorphism.* Therefore, similarity computations can sometimes be difficult. A discussion of these issues is provided in [457].

The problem of outlier detection in such cases is similar to multidimensional point anomaly detection, where each graph object is treated as an individual point. In this context, the easiest class of models to generalize are proximity-based models. This is because the problems of clustering and similarity search have been widely studied in domains such as chemical compound mining and XML graph analysis [13, 457]. In the case of clustering methods [13], the clusters are defined as sets of graphs which overlap with the frequent subgraph patterns in the underlying data set. Since outliers are defined in a complimentary way to clusters, they are defined as graphs which do not overlap well with the frequent subgraph patterns in the data. Thus, a natural approach for outlier detection in such scenarios is to determine the frequent subgraphs in the underlying data, and determine those graphs which do not contain these frequent patterns. This approach is analogous to determining the outliers in transaction data with the use of pattern mining. Outliers can be defined as graphs which do not contain a significant number of frequent subgraph patterns. Therefore, an approach analogous to that discussed in section 5.1 of Chapter 7, for outlier detection in transaction data [209] can also be used for the case of graphs. The same quantifications based on support can be used in this case, except that the support is defined on subgraph frequent patterns, rather than transaction frequent patterns.

The use of a k-nearest neighbor outlier detection algorithm is also a viable alternative in this case, because of a wide variety of available algorithms for similarity search in graphs [457]. Numerous similarity functions are commonly used such as the graph edit distance, largest common subgraph, largest matching node set, in order to perform the similarity search. While these methods correspond to generic extensions of multi-dimensional outlier detection algorithms, not much work has been specifically targeted towards outlier detection in this domain.

3.   Outlier Detection in a Single Large Graph

In large graphs, unusual structural characteristics can be used to define outliers in different regions of the network. Such outliers can be defined in a variety of ways, such as node outliers, linkage outliers, or subgraph outliers. Each of these different kinds of outliers will be discussed in some detail. Throughout this section, it will be assumed that

a single large network or graph is available for analysis. This graph is denoted by $G = (V, E)$. Here, V denotes the set of vertices, and E denotes the set of edges.

3.1 Node Outliers

Node outliers can be defined in a network in a wide variety of ways. The key is to extract features from the neighborhood of a node, and then define the outliers in terms of these feature. The work in [33] defines a set of features which are extracted from the 1-step neighborhood (also known as *egonet*) of node i. The specific features which are extracted are as follows:

- ■ (Node Feature n_i) The number of nodes in the 1-step neighborhood of node i (which is the same as the degree).

- ■ (Edge Feature e_i) The number of edges between all nodes in the 1-step neighborhood of node i.

- ■ (Weight Feature w_i) For weighted graphs, the features comprise the weight of all edges in the 1-step neighborhood of node i.

- ■ (Spectral Feature λ_i) The principal eigenvalue of the weighted subgraph in the 1-step neighborhood of node i.

These features can then be organized into carefully chosen pairs, which are shown to obey a number of interesting power laws. Data points which deviate from these power laws are flagged as outliers. Some examples of useful pairs for anomaly detection were as follows:

- ■ Node Feature vs Edge Feature: Extremely dense neighborhoods of a node correspond to near-cliques, whereas extremely sparse neighborhoods of a node are stars. By plotting the relationship between the node an edge features, it is possible to detect near cliques and stars. In general, the *expected relationship* between these features is $e_i \propto n_i^\alpha$, where $\alpha \in (1, 2)$.

- ■ Weight Feature vs Edge Feature: If edges are received unusually frequently in the neighborhood of a node, this will result in a high value of the weight feature relative to the edge feature. This corresponds to the *heavy vicinity* of a node. The expected relationship between these features is $w_i \propto e_i^\beta$, where $\beta \geq 1$, but usually ranged in $(1, 1.29)$ according to the experiments in [33].

- ■ Spectral Feature vs Weight Feature: This detects a single dominating heavy edge in the neighborhood of a node. The expected relationship between these features is $\lambda_i \propto w_i^\gamma$, where $\gamma \in (0.5, 1)$.

The above rules are fairly simple to implement, and provide an effective characterization of the underlying outliers. It should be noted that many features can be mined from the neighborhood of a graph, each of which provides a different notion of an outlier. The aforementioned possibilities simply reflect three *instantiations* of features extracted from a graph neighborhood, so that deviations from power-law models can be used in order to define outliers. Power laws correspond to specific *domain knowledge* about the behavior of *typical* networks. As discussed frequently in the book, the use of domain knowledge is often very useful for meaningful outlier analysis.

On the other hand, not all features extracted may satisfy such power laws. In general, a wide variety of features can be extracted from the neighborhood of nodes. This can be used to define a multi-dimensional feature space, in which the features extracted from each node correspond to a multi-dimensional data point. For example, the shape of the degree distribution of the neighbors of a node provides a different characterization of the outlier behavior. A general meta-algorithm for node outlier analysis in networks is as follows:

1. Extract features $f_1 \ldots f_r$ from the k-hop neighborhood of a node. Create a new multi-dimensional data point $(f_1 \ldots f_r)$ specific to that node.

2. Apply any point anomaly detection algorithm to the extracted multi-dimensional data set, and report unusual data points as outliers.

Virtually an unlimited number of different features could be extracted from a node. The precise features extracted should depend on the application at hand. For example, in a social network, it could be a vector containing the number of messages exchanged with each neighboring node, and in a spam detection application, it could correspond to the degree of the node. Therefore, the key in effective node outlier detection is to extract these features in a way which is sensitive to the particular application at hand.

3.2   Linkage Outliers

Linkage outliers are defined as edges which lie across dense partitions in the network. In other words, if the nodes in the graph are clustered, then the edges across these node clusters are defined as outliers. Unfortunately, there is no single way to define the clusters in a network. In many cases, the clustering may be imperfect, and may merge two sets of nodes which should ideally be in different clusters. In order to address

these issues, the method in [15] defines a clustering which is based on randomized sampling of the edges in the network. Since the connected components in edge samples are known to create *weak clusters* of densely connected nodes [253], such an approach is an excellent strategy for link outlier estimation. Furthermore, this randomized sampling is repeated in order to create the clusters in different ways. For each clustering, a likelihood fit is defined for an edge. The median of these likelihood fits is defined as the final anomaly score.

Consider a partitioning of the nodes denoted by $\mathcal{C} = C_1 \ldots C_{k(\mathcal{C})}$. The number of node partitions in $\mathcal{C}$ is denoted by $k(\mathcal{C})$. Each set C_i represents a disjoint subset of the nodes in V. The likelihood fit is defined with respect to a *structural generation model* of edges with respect to node partitioning $\mathcal{C}$. This defines the probability of an edge between a pair of partitions. An edge with lower probability is defined as an outlier.

DEFINITION 11.1 (EDGE GENERATION MODEL [15]) *The structural generation model of a node partitioning $\mathcal{C} = \{C_1 \ldots C_{k(\mathcal{C})}\}$ is defined as a set of $k(\mathcal{C})^2$ probabilities $p_{ij}(\mathcal{C})$, such that $p_{ij}(\mathcal{C})$ is the probability of a randomly chosen edge in the network to be incident on partitions i and j.*

The probabilities $p_{ij}(\mathcal{C})$ are estimated from the underlying network. Specifically, the fraction of edges between any pair of partitions in the network can be used in order to estimate this probability. The partition identifier for node i is denoted by $I(i, \mathcal{C})$, and it takes on a value between 1 and $k(\mathcal{C})$.

DEFINITION 11.2 (EDGE LIKELIHOOD FIT) *Consider an edge (i, j), a node partition $\mathcal{C}$, and edge generation probabilities $p_{ij}(\mathcal{C})$ with respect to the partition. Then, the likelihood fit of the edge (i, j) with respect to the partition $\mathcal{C}$ is denoted by $\mathcal{F}(i, j, \mathcal{C})$ and is given by $p_{I(i,\mathcal{C}), I(j,\mathcal{C})}$.*

The likelihood fit of an edge can be defined with respect to any partition in the network. As discussed above multiple clustering models are created, in order to increase the robustness of likelihood estimation. Since each partitioning provides a different way to construct the generation model, it provides a different way of estimating the edge generation probabilities. By combining these different ways of estimation, the variations which are specific to a particular partitioning can be smoothed out, and a robust estimation of probabilities can be provided. The r different ways of creating the partitions are denoted by $\mathcal{C}_1 \ldots \mathcal{C}_r$. Specifically, the ith clustering $\mathcal{C}_i$ contains $k(\mathcal{C}_i)$ different clusters. The composite edge-likelihood fit from these r different ways of creating the partitions is defined as the *median* of the edge likelihood fits over the different partitions.

DEFINITION 11.3 (EDGE LIKELIHOOD FIT (COMPOSITE)) *The composite edge likelihood fit over the different clusterings $\mathcal{C}_1 \ldots \mathcal{C}_r$ for the edge (i, j) is the median of the values of $\mathcal{F}(i, j, \mathcal{C}_1) \ldots \mathcal{F}(i, j, \mathcal{C}_r)$. This value is denoted by $\mathcal{MF}(i, j, \mathcal{C}_1 \ldots \mathcal{C}_r)$.*

A variety of randomized clustering methods can be used in order to create the different partitions. For example, any off-the-shelf randomized algorithm can be used for partitioning purposes. A method has also been proposed [15] in order to maintain the clusterings for a *stream of edges* with an online reservoir sampling approach. Thus, this approach can be used in order to determine edge anomalies in graph streams. The reservoir sampling approach will be discussed in detail in section 5.1.

3.2.1   Community Linkage Outliers.   In some network applications, many small graphs may be superimposed on a single large graph. For example, in a bibliographic network, a small graph object may represent a publication. In such cases, it is desirable to identify objects, for which the linkage structure is anomalous compared to the linkage structure of other objects in the network. This case is often not very different from determining linkage outliers. The main difference is that the likelihood scores for the different edges in the object need to combined into a single likelihood score [15].

The likelihood fit for a graph object G is the product of the likelihood fits of the edges in G. In order to fairly compare between graphs which contain different numbers of edges, we put the fraction $1/|G|$ in the exponent, where $|G|$ is the number of edges in the incoming graph stream object. In other words, we use the geometric mean of the likelihood fits of different edges in the incoming graph stream object. Therefore, we define the object likelihood fit as follows.

DEFINITION 11.4 (GRAPH OBJECT LIKELIHOOD FIT) *The likelihood fit $\mathcal{GF}(G, \mathcal{C}_1 \ldots \mathcal{C}_r)$ for a graph object G with respect to the partitions $\mathcal{C}_1 \ldots \mathcal{C}_r$ is the geometric mean of the (composite) likelihood fits of the edges in G.*

$$\mathcal{GF}(G, \mathcal{C}_1 \ldots \mathcal{C}_r) = \left[\prod_{(i,j) \in G} \mathcal{MF}(i, j, \mathcal{C}_1 \ldots \mathcal{C}_r) \right]^{1/|G|} \tag{11.1}$$

Objects which have extremely low likelihood fits are reported as community linkage outliers [15]. In many cases, any particular linkage in the set may not have a low fit, but the overall fit, when evaluated over the entire set, may be very low. Thus, such outliers are examples of *collective* outliers, whereas node and linkage outliers are *contextual* outliers.

Strictly speaking, this measure can be defined for any set of nodes in a single large network, rather than a specified set of nodes in the small graph superimposed over a larger network. In such cases, an additional challenge is to *discover* the appropriate sets of nodes which are connected together in an anomalous way. This is a much more difficult problem with the use of only network structure information, unless additional content-based supervision is available. A closely related notion of community outliers was proposed in [180], which finds such anomalous communities in a single large network. However, node content is used in order to provide additional supervision for the discovery of such anomalous sets of nodes.

3.2.2 Matrix Factorization Methods. Matrix factorization provides a natural way to determine anomalies in data which is represented in the form of a 2-dimensional matrix. Note that all forms of graphs can be represented in the form of an adjacency matrix A. The entries of the adjacency matrix may wither be binary, or they may reflect weights on the edges. Furthermore, the matrix may not be square, depending upon the nature of the graph. In the case of a bipartite graph with p and q nodes on either size, the matrix can be most efficiently [1] represented as a $p \times q$ matrix, which is not necessarily square. Therefore, the problem of finding linkage anomalies in such graphs can be stated as the problem of determine atypical entries in a matrix of values.

PROBLEM 3.1 *Given a matrix A of size $p \times q$ determine anomalous entries in the matrix.*

Matrix factorization provides a natural technique for finding linkage anomalies in such networks. Specifically, the matrix A is factorized into two *low rank* matrices of U and V of size $p \times r$ and $r \times q$ respectively, for some relatively small value of r.

$$A \approx U \cdot V \qquad (11.2)$$

The smaller the value of r, the greater the level of approximation in the aforementioned factorization. The determination of U and V is achieved by constructing the residual matrix R, and optimizing an objective function such as the sum of the squares of the values in the residual matrix.

$$R = A - U \cdot V \qquad (11.3)$$

---

[1]This allows the most efficient representation, rather than using a $(p + q) \times (p + q)$ matrix with large blocks of zero entries.

Different objective functions such as the Frobenius norm (sum of squares of entries of R), can be used [452] in order to determine the optimal values of U and U. Clearly, large *absolute* values of the residual correspond to linkage anomalies. Any form of extreme value analysis may be used on the matrix R in order to determine the linkage anomalies. In order to improve interpretability of the anomaly detection technique, it is possible to impose different kinds of non-negativity constraints, such as those on the factorized matrices U and V, or on the residual matrix itself. Details of the reasons for the greater interpretability of non-negative matrix factorizations are provided in [452].

3.2.3   Spectral Methods.   The matrix factorization methodology discussed above can be (conceptually) replicated with the use of spectral methods. Let Q be the $n \times m$ node-link adjacency matrix of the network containing n nodes and m edges. This matrix may be either un-weighted or weighted, depending upon whether or not the edges are weighted. Then $A = Q \cdot Q^T$ is an augmented adjacency matrix with the diagonal entries containing either the degrees (un-weighted case) or the sum of the squares of the weights. Then the matrix A is positive semi-definite, as are all matrices of the form $Q \cdot Q^T$. Therefore, it can be diagonalized as follows:

$$A = P \cdot D \cdot P^T \qquad (11.4)$$

Here P is a matrix, whose columns contain the orthonormal eigenvectors, and D is a diagonal matrix. Let D_r be the rank r truncated matrix of D, in which only the top r eigenvalues are retained, and the remaining are reset to zero. Then, the corresponding approximated version A_r of the original matrix A is as follows:

$$A_r = P \cdot D_r \cdot P^T \qquad (11.5)$$

As before, the residual matrix may be computed as follows:

$$R = A - A_r \qquad (11.6)$$

Entries with large residual values may be declared anomalies. When diagonal entries have large residuals compared to other diagonal entries, the corresponding nodes may be considered anomalous. These methods are intuitively very similar to the matrix factorization methods discussed above, except that PCA is used for determining the residuals. Of course, the actual values of the residuals will be somewhat different, since the models are not mathematically identical. The interpretability of such methods is lower, since it is no longer possible to impose non-negativity constraints on the residuals.

3.3 Subgraph Outliers

A subgraph outlier is defined as a part of the graph which exhibits unusual behavior with respect to the normal patterns in the full graph. Subgraph outliers cannot be defined meaningfully in a large graph, unless there are repetitions in the labels on the nodes. For example, in a web or social network graph, in which all node identifiers are distinct, it is much harder to define a notion of regularity. However, if the nodes are associated with labels drawn from the relatively small subset of possibilities, it is much easier to define subgraph outliers by finding subgraphs which relate these labels to one another in unusual ways. Alternatively, the nodes may not have labels associated with them, and the matching is based purely on structural similarity. Either of these scenarios can be addressed by the work in [349], which designs information-theoretic models for subgraph outlier detection.

The approach proposed in [349] is based on the Minimum Description Length (MDL) principle for outlier detection. The broader principles underlying the approach are based on the SUBDUE system for subgraph pattern mining. In the subdue system, a commonly occurring pattern is one which provides a small description length for the purposes of representation. For example, if S is a substructure which occurs repeatedly in a graph G, then by compressing that substructure into a single vertex, the graph is compressed (corresponding to a smaller description length), and the overall graph can be represented in terms of the description of the compressed graph and the smaller substructure. Therefore, the frequent nature of a substructure S in graph G may be leveraged in order to describe the compressed representation of the graph concisely in terms of the description length $DL(G|S)$ of the compressed graph and that of the repeatedly occurring substructure S. The conciseness is achieved, because the repeatedly occurring substructure needs to be described only once. This description length $F1(S, G)$ is defined as follows:

$$F1(S, G) = DL(G|S) + DL(S) \tag{11.7}$$

Subgraphs which are very common (and therefore not outliers) will have low values of $F1(S, G)$. One possible solution is to determine those subgraphs which have *high values* of $F1(S, G)$ and flag them as outliers. Such an approach does not seem to work very well for anomaly detection because it does not discriminate between the varying sizes of subgraphs. For example, subgraphs with only 1 node will often have high values of the anomaly score according to this criterion. Therefore, a different heuristic criterion was proposed in [349], which is based on the spirit of the original definition for pattern frequency, but normalizes for subgraph

size as well. Specifically, this definition is as follows:

$$F2(S, G) = Size(S) * Instances(S, G) \qquad (11.8)$$

Here, $Size(S)$ represents the number of nodes in the subgraph S, and $Instances(S, G)$ represents the number of instances of subgraph S in the graph G. Subgraphs with low value of $F2(S, G)$ are reported as outliers.

A second approach is based more directly on the SUBDUE method [123]. The SUBDUE method is designed to determine the common substructures in the graph in an iterative way. Specifically, this method compresses the common substructures in the graphs, and replaces them with new nodes. Substructures which are more common are compressed in earlier iterations. The key insight in detecting anomalous subgraphs is determine whether significant portions of the subgraph were compressed in early iterations. If this is *not* the case, it implies that the subgraph contains few common substructures. Such a subgraph should be considered anomalous. Therefore, the anomaly score A of a subgraph S is defined in terms of the iteration index i of the SUBDUE method as follows:

$$A = 1 - \sum_{i=1}^{n} \frac{(n-i+1)}{n} \cdot \frac{DL_{i-1}(S) - DL_i(S)}{DL_0(S)} \qquad (11.9)$$

The anomaly score always lies in the range $(0, 1)$. High values indicate that the object is anomalous. The description length $DL_i(S)$ of a subgraph typically reduces with increasing iteration index i, because of the compression of nodes inside it. Note that the term $\frac{DL_{i-1}(S) - DL_i(S)}{DL_0(S)}$ corresponds to the fraction of compression in the ith iteration, and the term $\frac{(n-i+1)}{n}$ provides greater weight to a compression occurring in an earlier iteration. This is because more common graphs are compressed in earlier iterations. Therefore, the value of the anomaly score will be impacted by high quickly the nodes of the subgraph S become compressed because of the membership of common patterns in them. This provides a heuristic definition of the anomaly score, based on the MDL principle.

4.   Node Content in Outlier Analysis

The incorporation of node content can have significant benefits in outlier analysis, particularly in terms of discovery of outlier links between nodes. This is because in most domains such as the web, social networks, and information networks, linkages between nodes are highly correlated to content similarity between nodes. For example, web pages with similar content are more likely to link with one another. In social networks, network participants are more likely to be linked with other

participants with similar interests. On the other hand, nodes with certain kinds of content are very unlikely to link with one another. For example, an official US Government web site is unlikely to link to a web page containing certain kinds of questionable content. Thus, there are two kinds of outlier linkages which can be discovered:

- Noisy outlier linkages can degrade the quality of a variety of network mining algorithms. Therefore, it may be useful to detect and remove such linkages before performing the mining. This problem is referred to as *network denoising* [178, 378].

- Truly anomalous linkages can provide insights about unusual connections among nodes. This may have applications beyond noise removal, since the formation of anomalous connections is indicative of interesting facts about the underlying system.

Three methods have recently been proposed for incorporating node content in the outlier analysis process. These methods use different levels of consistency between the content and clustering structure in order to determine anomalies in the linkage structure.

The first method [178] models the similarity in the feature vectors at the nodes to an intuitive notion of tie-strength across links. This tie-strength is modeled as a weight vector, and represents the consistency of links with their neighborhood feature vector. Each node i has a feature vector f_i associated with it. Let F_i be a $k \times d_i$ matrix which has one row for each of the k features, and one column for each of the d_i neighbors of node i. Let w_i be a non-negative column vector of length d_i which represents the tie string of the node i to each of its neighbors. Then, $F_i \cdot w_i$ represents the set of predicted features of node i with the use of the weight vector in conjunction with the features at the neighbors of node i. The error of this prediction is given by $||F_i \cdot w_i - f_i||_2^2$. In addition, in order to incorporate different levels of sparsity for learning the weight vector, a Lagrangian term is incorporated into the objective function. The following objective function is minimized:

$$\min_{w_i \geq 0} \sum_i (||F_i \cdot w_i - f_i||_2^2 + \lambda \cdot ||w_i||_1) \qquad (11.10)$$

Links which correspond to very small values of the weights (or zero weights) are assumed to be outliers. Here λ is a parameter which regulates the tradeoff between the sparsity of the weight vector and the learned accuracy. Larger values of λ will result in more sparse weight vectors, and therefore fewer links will be included. Note that this is a convex non-linear programming problem. A variety of off-the-shelf solvers [90] can be used in order to solve this problem.

The second method [378] uses a heterogeneous random field framework in order to determine the consistency of links with the clustering structure and the network. Therefore, this model associates a binary variable $n(i, j)$ in order to determine whether or not a link should be considered anomalous. Similarly, a variable is associated with each node, which indicates its affinity to each cluster. Then, the two variables are *simultaneously* learned in order to determine the clusters in the network and the outlier links simultaneously. The criteria for learning these variables incorporates consistency in the clustering structure of the network with the node content. The key idea is to create clusters and infer the anomalousness of links simultaneously in a *mutually consistent way*. This ensures that the clustering structure is consistent with the overall network structure, and the content at the different nodes. At the same time, the linkages which are not consistent with the learned variables are declared as anomalous. A heterogeneous markov random field (HMRF) is used in order to learn these variables. Interested readers are referred to [378] for details.

Finally, a method [180] known as *community outlier detection* determines outlier communities in the network by using a combination of content-based and structural analysis. Linkage outliers try to find *a pair* of unusually connected nodes in the network. Community-outliers can be considered unusual subgraphs in the network, and are therefore a generalization of link outliers, by allowing more than two nodes in defining a set of unusually connected nodes. The technique in [180] proposes a method for efficient determination of community outliers in information networks. This technique also uses a heterogeneous markov random field (HMRF) in order to determine interesting sets of nodes which are inconsistent with the observed labels. Interested readers are referred to [180] for details.

5.   Change-based Outliers in Temporal Graphs

Many entities such as the web, social, and information networks are temporal in nature, in which new linkages continuously form over time. In certain kinds of networks such as communication networks, the underlying edges may be received very fast. This results in a fast edge stream. In such cases, unusual and abrupt changes in the structure of the network should be detected and reported as anomalies. As discussed in earlier chapters, outlier detection and change analysis are two tightly integrated problems [160, 479], especially when the changes are abrupt, and reflect the occurrence of unusual events in the underlying generating process. In the context of graphs and networks, the evolution patterns

correspond to different kinds of structural pattern changes. Some examples are as follows:

- *Linkage Anomalies:* The arrival of new edges which are inconsistent with the past history of the graph stream may be considered anomalous. For example, the arrival of a link between two regions of the network which are normally sparsely connected to each other based on previous history may be considered anomalous. This is a similar definition to the concept of linkage anomalies in the static case, except that a link should be compared only with respect to the portion of the graph, which has been received till the current time.

- *Community Evolution:* In this case, significant and unusual changes in the community structure are tracked [17, 192, 429], and reported as anomalies.

- *Distance Evolution:* In this case, pairs of nodes with unusually large changes in the distances are determined [193], and reported as anomalies.

- *Pattern-based Evolution:* In this case, significant local patterns of changes in the network are characterized as evolution rules [64].

Each of these different kinds of temporal anomalies provide different kinds of insights, and will be discussed in this section.

5.1 Stream Oriented Processing for Linkage Anomalies

The methods in sections 3.2 were presented for static data sets. These methods are also applicable[2] for anomaly detection in an edge stream. The only difference is that the likelihood estimate for a linkage anomaly needs to be computed with respect to the previous history of the graph stream. Furthermore, the cluster partitioning for computing the likelihood fits needs to be maintained using an online approach. A *fully* stream-oriented approach does not necessarily work with snapshots but maintains a continuous summary of the entire data stream, so that the arrival of unusual edges can be detected immediately.

As was discussed earlier in section 3.2, cluster-based partitions need to be maintained dynamically from the edge stream[3] in order to perform

---

[2]In fact, these quantifications were originally proposed in the context of stream processing [15], but are also applicable to any kind of data.

[3]The work in [15] uses a stream of composite objects, though this presentation simplifies it to edge streams. Either scenario can be handled by the approach with small modifications.

the linkage anomaly detection. It is well known that the use of edge sampling [253] can be used to create dense partitions. For example, a sample of edges from a stream implicitly creates a set of clusters in terms of the connected components in this sample. Such connected components are much denser than randomly picked node sets in the graph, because of the inherent bias of edge sampling [253]. A major challenge arises in adapting the minimum 2-way cut methods of [253] to a more general stream scenario, while maintaining specific *structural properties* of the k-way cut partitions. For example, one possible structural constraint would be to ensure a minimum number of points in each cluster, or to constrain the total number of clusters. Clearly, a random edge sample may not satisfy such constraints. Reservoir sampling [454] is a methodology to dynamically maintain an unbiased sample from a stream of elements. The method of [15] extends this method to an unbiased sample of a structured graph, so that many natural and desirable structural properties of the sample are maintained. This goal is achieved with the help of a *monotonic set function* of the underlying edges in the reservoir. A monotonic set function is defined on the sample as follows.

DEFINITION 11.5 (MONOTONIC SET FUNCTION) *A monotonically non-decreasing (non-increasing) set function is a function $f(\cdot)$ whose argument is a set, and value is a real number which always satisfies the following property:*

- *If S_1 is a superset (subset) of S_2, then $f(S_1) \geq f(S_2)$*

The monotonic set function can be useful for regulating the structural characteristics of the graph over a given set of edges. Some examples of a monotonic set function with corresponding structural properties are as follows:

- The function value is the number of connected components in the edge set S (monotonically non-increasing).

- The function value is the number of nodes in the the largest connected component in edge set S (monotonically non-decreasing).

Properties such as the above are very useful for inducing the appropriate partitions with robust structural behavior. In some cases, it is possible to use *thresholds* on the above properties, which are also referred to as *stochastic stopping criteria.*

Let $\mathcal{D}$ be a set of edges. Consider a restricted bunch of subsets of $\mathcal{D}$, in which the edges are sorted, and can be added to S only in *sort order priority*; in other words, an edge cannot be included in a subset S, if all

the elements which occur before it in the sort order are also included. Clearly, the number of such subsets of $\mathcal{D}$ is linear in the size of $\mathcal{D}$.

DEFINITION 11.6 (SORT SAMPLE WITH STOPPING CRITERION) *Let $\mathcal{D}$ be a set of edges. Let $f(\cdot)$ be a monotonically non-decreasing (non-increasing) set function defined on the edges. A sort sample S from $\mathcal{D}$ with stopping threshold α is defined as follows:*

- *All edges in $\mathcal{D}$ are sorted in random order.*

- *The* smallest *subset S from $\mathcal{D}$ among all subsets which satisfy the sort-order priority* is picked, such that $f(S)$ is at least (at most) α.

This means that if the last added element is removed, then that set (and all previous subsets) will not satisfy the stopping criterion. As a practical matter, the set which is obtained by removing the last added element is the most useful for processing purposes. For example, if $f(S)$ is the size of the largest connected component, the stopping criterion determines the smallest sample S, such that the size of the largest connected component is at least a user-defined threshold α. By dropping the last edge (v, w) which was added to S, it is guaranteed that the size of the largest connected component in $S - \{(v, w)\}$ is less than α. Correspondingly, a *penultimate set* for a sort sample is defined as follows.

DEFINITION 11.7 (PENULTIMATE SET) *The penultimate set for a sort sample S is obtained by removing the last element in the sort order of sample S.*

For the case of a *fixed data set*, it is fairly easy to create a sort sample with a structural stopping criterion. This is achieved by sorting the edges in random order and adding them sequentially, until the stopping criterion can no longer be satisfied. However, in the case of a data stream, a random sample or reservoir needs to be maintained *dynamically*. Once edges have been dropped from the sample, it becomes a challenge to compare their sort order to new incoming edges.

The key idea is use a *fixed random hash function*, which is computed as a function of the node labels on the edge, and remains fixed over the entire stream. This hash function is used to create a sort order among the different edges. This hash function serves to provide *landmarks* for incoming edges when they are compared to the previously received edges from the data stream. Furthermore, the use of a hash function *fixes the sort order among the edges throughout the stream computation*. The fixing of the sort order is critical in being able to design an effective structural sampling algorithm. Therefore, for an edge (i, j) the hash

function $h(i \oplus j)$ is computed, where $i \oplus j$ is the concatenation of the node labels i and j. The use of a sort on a random hash function value induces a random sort order on the stream elements. Furthermore, a stopping criterion on the sorted data set *translates* to a threshold on the hash function value. This provides an effective way to control the sampling process. In other words, it is desirable to determine the smallest threshold q, such that the set S of edges which have hash function value at most q satisfy the condition that $f(S)$ is at least (at most) α. The key observation here is that the value of the threshold q varies over the life of the data set, as more edges arrive, and a smaller *fraction* of the edges in the stream need to be included in the reservoir in order to satisfy the structural constraint. Intuitively, this "smaller" fraction translates to lower hash thresholds. In fact, the following result has been explicitly shown in [15].

THEOREM 11.8 (HASH THRESH. MONOTONICITY) *The stopping hash threshold is monotonically non-increasing over the life of the data stream.*

This result implies that edges which have not been included in the current sample will *never* be relevant for sampling over the future life of the data stream. Thus, there is no risk that any stream edge which was discarded will become useful later. The current sample is the only set needed for any future decisions about reservoir sample maintenance. The key here is to find ways to dynamically update the hash threshold and the stream sample continuously so as to maintain the structural constraints.

A simple algorithm can be designed in order to maintain the reservoir dynamically. The *current hash threshold* is maintained dynamically to make decisions on whether or not incoming elements are included in the reservoir. For each incoming edge, the hash function is applied, and it is added to the reservoir, if the hash function value is less than the current threshold value. The addition of an edge will always result in the stopping criterion being met because of set function monotonicity. However, the set may no longer be the *smallest sort-ordered set* to do so. Therefore, edges may need to be removed in order to make it the smallest sort-ordered set to satisfy the stopping criterion. In order to achieve this goal, the edges in the reservoir are processed *in decreasing order of the hash function value* and removed, until the resulting reservoir is the smallest possible set which satisfies the stopping constraint. The corresponding hash threshold is then reduced to the largest hash function value of the *remaining edges in the reservoir* after removal. Clearly, the removal of edges may result in a reduction of the hash threshold in each iteration. However, it will never result in an increase in the threshold,

because all the added edges had a hash function value lower than the threshold in the previous iteration. The penultimate set derived from the sort sample is always used for the purposes of maintaining the cluster partition as the set of underlying connected components.

The above description explains the maintenance of a single reservoir (and corresponding partition). For robustness, a set of r different reservoirs is maintained, which corresponds to r different partitionings of the data. Therefore, r different hash functions are used in order to create the different reservoir samples. These are used to define the r different partitionings of the nodes as required by the outlier modeling algorithm. The *penultimate sets* of the r different reservoirs are denoted by $S_1 \ldots S_r$. These will be used for the purpose of inducing the r different partitionings denoted by $\mathcal{C}_1 \ldots \mathcal{C}_r$. Correspondingly, the outlier score of an edge can be computed from the likelihood fits, as discussed in section 3.2 of this chapter.

5.2 Outliers based on Community Evolution

The most common form of evolution is based on significant changes in the structure of the communities of the underlying network. In many social and information networks, the changes in patterns of activity may lead to the formation and disappearance of communities in the network. An analysis of the community evolution structure provides numerous insights about sudden changes in membership of nodes, the formation or disappearance of clusters, the erratic membership of nodes in clusters, and a general change in the overall clustering quality of the data. In the field of community evolution, there are two kinds of algorithms which are commonly proposed in the context of evolving networks.

- The first class of algorithms addresses how to dynamically maintain the set of communities in an evolving network. An example of such an approach is provided in [99].

- The second class of algorithms addresses how to determine the *key changes* in the community structure. While the first class of algorithms are often used for determining the change points, this may not always be the case. The determination of the key points of change is best performed by algorithms which are optimized to this scenario.

Since the second class of algorithms is more relevant to change analysis and outlier detection, this chapter will focus on those algorithms.

5.2.1   Online Analysis of Community Evolution in Graph Streams.   One of the earliest works on community evolution in graph streams was presented in [17]. It is assumed in this model, that the number of edges (i, j) which have arrived till time t is denoted by $w_{ij}(t)$. Each edge (i, j) in this model is assumed to be an interaction between actors i and j. The goal of this work is to determine communities which change significantly over time, such as highly expanding communities or highly contracting communities. This is achieved with the use of the concept of a *differential graph*.

The differential graph is defined over a particular interval (t_1, t_2), and is defined as a fractional rate at which the level of interaction along an edge has changed during the interval (t_1, t_2). In order to generate the differential graph, the *normalized graph* at the times t_1 and t_2 is constructed. The normalized graph $G(t) = (N(t), A(t))$ at time t is denoted by $\overline{G(t)}$, and contains exactly the same node and edge set, but with different weights. Let $W(t) = \sum_{(i,j) \in A} w_{ij}(t)$ be the sum of the weights over all edges in the graph $G(t)$. Then, the normalized weight $\overline{w_{ij}(t)}$ is defined as $w_{ij}(t)/W(t)$. The normalized graph therefore contains the fraction of interactions of each edge.

The differential graph is constructed from the normalized graph by using a subtractive process on the normalized graphs at snapshots t_1 and t_2. Therefore, the differential graph $\Delta G(t_1, t_2)$ contains the same nodes and edges as $\overline{G(t_2)}$, except that the differential weight $\Delta w_{ij}(t_1, t_2)$ on the edge (i, j) is defined as follows:

$$\Delta w_{ij}(t_1, t_2) = \overline{w_{ij}(t_2)} - \overline{w_{ij}(t_1)} \tag{11.11}$$

In the event that an edge (i, j) does not exist in the graph $\overline{G(t_1)}$, the value of $w_{ij}(t_1)$ is assumed to be zero. Because of the normalization process, the differential weights on many of the edges may be negative. These correspond to edges over which the interaction has reduced significantly during the evolution process. For instance, in a bibliographic network, when the *rate of authoring new publications* between a pair of authors reduces over time, the corresponding weights in the differential graph are also negative.

Once the differential graph has been constructed, it is desired to determine node clusters with high evolution. A natural solution would be find the clustered subgraphs with high absolute weights in the differential graph. However, in a given subgraph, some of the edges may have high positive weight while others may have high negative weight. Therefore, such subgraphs correspond to the entity relationships with high evolution, but they do not necessarily correspond to entity relationships with

the greatest increase or decrease in interaction level. Instead, it is desired to determine subgraphs such that most interactions within that subgraph are all either highly positive or all highly negative. This is a much more difficult problem than that of finding clusters within the subgraph $\Delta G(t_1, t_2)$. Methods for finding such subgraphs are proposed in [17]. One unique characteristic of this framework is that it directly uses the edge frequency *change* in order to model evolution. Most of the other community evolution methods focus on structural changes in the underlying network.

5.2.2   GraphScope.   A method known as *GraphScope* proposed in [429] has designed a method to perform 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. Such graphs are useful for modeling a wide variety of directional interactions, such as users with web sites, clients with hosts etc. Furthermore, since these interactions are dynamic, the patterns in the interactions will change over time.

The work in [429] uses a combination of dynamic community detection and information theoretic methods in order to determine the key change points in the underlying data. The Minimum Description Length (MDL) principle is used in order to determine the change points in the data. Intuitively, a change point is one which significantly increases the encoding cost in order to represent the stream.

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. It has been shown in [429], that such change points correspond to drastic discontinuities in the network, which can be regarded as temporal outliers. Readers are referred to [429] for details.

5.2.3   Integrating Clustering Maintenance with Evolution Analysis.   While evolutionary clustering [99, 116] is widely studied in the community detection literature, a recent method [192] also inte-

grates the clustering process with evolution analysis. The integration of an evolution analysis procedure into the clustering process is critical in determining relevant change points in the data. The work in [192] uses a probabilistic clustering model in order to learn the clusters from the underlying data.

The *ENetClus* method [192] generalizes the probabilistic *NetClus* [435] model to the temporal scenario. This is a soft clustering model, which assigns probabilities of membership of each node to different clusters. The idea in this model is to perform the clustering on temporal snapshots of the data. On each snapshot, a probabilistic assignment is learned with the use of the *NetClus* algorithm. The final probabilistic assignment in a given snapshot is used as an initialization point (prior) for the next iteration. This ensures that continuity is maintained among the clusters, and the clusters found in the next snapshot can be directly compared to their counterpart in the current snapshot. A number of evolution metrics are then proposed in order to 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 (by using the methods discussed in Chapter 8) can be reported as anomalous changes in the network. A subset of the more important temporal change quantifications introduced in [192] are presented below:

- *Cluster Membership Consistency:* Since the *ENetClus* algorithm determines probabilities of membership for each node to the different clusters, the vector of probabilities of membership of the different nodes to a particular cluster, can be compared to the corresponding vector in the next snapshot. The cosine similarity between these vectors can be reported as the change value. This provides a global quantification of the change in cluster memberships.

- *Cluster Snapshot Quality:* The ratio of intra-cluster similarity to inter-cluster similarity is used as a quantification of the quality of the clusters. Significant changes in the clustering quality between successive snapshots is indicative of the fact that the inherent clustering tendency of the network has changed significantly over time. Thus, this form of change provides a more global understanding of the clustering tendency over different snapshots of the data. For

example, if a network receives a significant number of spam links in a short period of time (an anomalous event), this could abruptly disrupt the clustering tendency of the data set. This will show up as a sudden change in the time series representing a quantification of the clustering quality.

- *Cluster Novelties, Merge, Split and Disappearance:* The formation of new clusters represents a novelty in the data. Similarly, significant structural changes may occur, corresponding to cluster merge, split or disappearance. Each of these different structural anomalies is quantified in [192].

- *Temporal Object Stability:* Objects which move across different clusters over time are inherently unstable, and should therefore be considered outliers. The fraction of time-stamps at which the membership of the cluster remains the same between successive snapshots is a definition of temporal object stability. The inverse of this quantity is the outlier score for the object.

- *Temporal Object Sociability:* Objects which inherently belong to many different clusters are considered sociable. In the context of a soft clustering algorithm, this means that the membership probability vector for the object is distributed across different clusters. This definition of sociability is different from the degree of a node, since it is performed in the context of the community behavior, which provides more intuitive insights into aggregate sociability trends. This can be quantified by using the gini-index G, or the entropy E for the object in terms of cluster membership probabilities. Then, if $p_1 \ldots p_k$ be the membership probabilities for an object in the k different clusters, the underlying sociability can be defined in two different ways:

$$G = 1 - \sum_{i=1}^{k} p_i^2$$

$$E = -\sum_{i=1}^{k} p_i \cdot \log(p_i)/k$$

Larger values of each of these quantifications indicate greater sociability. For example, in a bibliographic network, this corresponds to authors who collaborate across many different research areas. Sudden changes in the sociability can provide an understanding of the significant changes in the collaboration patterns of a particular author.

5.2.4   Spectral Methods.   Spectral methods are closely related to community detection, and are often used to cluster networks [20]. These methods are also closely related to principal component analysis, though the precise matrix representation of the network similarity structure or the technique used for principal component analysis may vary with the specific application.

One major challenge in integrating temporal analysis with community detection algorithms is that the communities may sometimes change significantly on small variations of the graph, because of the *specific nature of the underlying algorithm*, rather than the network itself. This creates challenges for change analysis, because it becomes difficult to discern when a community change is caused by changes in graph structure, and when it is caused by the specifics of the underlying algorithm.

The major advantage of spectral methods is that they use the aggregate correlation structure of the linkages in the network. Such measures are extremely robust to small changes in the underlying network, and a significant change usually reflects a corresponding change in the structure of the network. While spectral methods can be implemented in a variety of ways, a simple method is to use principal component analysis on its augmented adjacency matrix. Let Q be a $n \times m$ node-link incidence matrix in a network containing n nodes and m edges. This is binary matrix containing only 0 or 1 values. A unit value implies that the corresponding edge is incident on that node. Then, the matrix $A = Q \cdot Q^T$ represents an augmented adjacency matrix, where the diagonal entries are the degrees on the nodes, and all other entries have 0-1 values depending upon whether or not a corresponding edge is present. In many interaction networks, weights are naturally associated with the edges. In such cases, the original node-link incidence matrix Q contains the weights instead of unit values. The weighted adjacency matrix $A = Q \cdot Q^T$ can also be defined in a similar way.

The matrix A is guaranteed to be positive semi-definite, since this is a property of all matrices of the form $Q \cdot Q^T$. Therefore, the matrix A can be diagonalized as follows:

$$A = P \cdot D \cdot P^T$$

Here P is an orthonormal matrix, whose columns contain the unit eigenvectors of A, and D is a diagonal matrix containing the eigenvalues. The eigenvector corresponding to the largest eigenvalue provides the principal directions of correlation. Significant changes in this vector over the graph snapshots over different periods of time, may correspond to anomalous behavior. Such an approach has been used in [229] in order to determine significant changes in temporally evolving graphs.

While the aforementioned method is a simple generalization of principal component analysis, other spectral methods commonly use the Laplacian of the similarity matrix. These methods are more directly related to the communities in the network [20] can be used in a similar way. A method for incorporating temporal smoothness in spectral clustering algorithms is discussed in [116]. While this method is not designed explicitly for change detection, it can be used for change detection, since an approximate mapping can be found between clusters at different time snapshots. This is because of the incorporation of the temporal smoothness criterion, which allows a clear mapping between clusters at different snapshots. A specific application of this kind of approach to the monitoring of evolution in blog communities is discussed in [348].

5.3   Outliers based on Shortest Path Distance Changes

Most real-world graphs such as the web, social networks and information networks experience significant changes in terms of the pairwise distances between nodes in the network. For example, it has been shown in [53] that most real graphs such as the web and social networks have shrinking diameters over time. This is because edges are continuously added to such networks, which makes them more dense.

In this context, *sudden and abrupt* changes in pairwise distances between nodes are indicative of unusual events in a network. For example, in a bibliographic network such as *DBLP* [518], 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. This problem was first proposed in [193].

A straightforward solution to this problem is to solve the all-pairs shortest path problem [32] at two snapshots t_1 and t_2. The pairs of nodes for which the distances have changed very significantly are reported as the anomalies. However, such an algorithm requires the computation and storage of all-pairs shortest paths, which can be impractical for larger graph applications. Virtually all social, communication, information and web networks are very large. For example, in a network containing 10^8 nodes, the number of possible pairs is 10^{16}. Even the storage of the pairwise distances is beyond the capability of most commodity

hardware. The computation of such node pairs is also impractical from the point of view of efficiency.

Therefore, it is important to design methods which can *efficiently* find the top-k distance changes in a heuristic way. A key observation in [193] is that edges which lie on the shortest paths between many pairs of nodes in either snapshot are important edges, the addition or deletion of which can significantly change the shortest path distances. Therefore, a randomized algorithm is proposed in [193] in order to find such edges. This is then leveraged in order to determine the significant nodes pairs between which the greatest change has occurred. While the determined node pairs are heuristic in nature, a high amount of precision and recall is achieved by this approach.

5.4   Temporal Pattern-based Outliers

Finally, many forms of pattern changes in a network may be characterized in the form of evolution rules. In the framework presented in [64], nodes and edges are associated with labels associated with specific properties of the network. Furthermore, edges contain the time-stamps corresponding to their first appearance. Patterns are defined as subgraphs, which have similar structure and labels on nodes at different time stamps, and the same relative offsets of the time-stamps. This defines significant temporal patterns or *graph evolution rules* in the underlying data. Evolution rules do not necessarily represent outliers, since they correspond to frequent temporal patterns in the data. On the other hand, the formation of a new evolution rule at a given time may be considered a temporal novelty, and may be reported as an outlier.

6.   Conclusions and Summary

The problem of network outlier detection is particularly important because of the ubiquity of different kinds of graphs and networks in a wide variety of real domains. Graphs can either occur as multiple entities of small size, or as a single large graph. Furthermore, since the nature of graphs is complex, the outliers can be defined in a wide variety of ways, depending upon how regularity is defined. Outliers can be defined in terms of nodes, edges, subgraphs, evolving edges, evolving subgraphs, or evolving distances. In the network context, the number of possible definitions of outliers are virtually unlimited. Therefore, it is critical to use application-specific properties to define outliers in networks in a meaningful way.

7. Bibliographic Survey

Outlier detection can be defined in the context of many small graphs, or in the context of a single large graph. The former case occurs frequently in scenarios such as chemical and XML data, in which unusual objects need to be determined [457, 13], Two common methods can be used. Distance-based methods can be easily generalized to this case [457], by defining appropriate similarity functions between the graphs. Alternatively, clustering or frequent pattern mining methods [13] can be used in order to identify significant anomalies.

Outliers in graphs can be defined in the form of node outliers, linkage outliers or subgraph outliers. The wide variety of ways in which outliers can be defined in networks is a direct result of the complexity of the data. Even in the context of particular kinds of outliers such as node outliers [33], it has been shown that a wide variety of definitions are possible, depending upon how the features are defined within the locality of a node. Linkage outliers are defined as edges which lie across dense clusters in networks [15]. A method for finding linkage outliers with the use of the Minimum Description Length (MDL) principle is proposed in [100]. The MDL principle is also useful for finding subgraph outliers [349]. Other work along a similar direction was presented in [148]. The use of eigenvector analysis in order to compare subgraphs with background behavior of the full graph and correspondingly declare them as anomalous was proposed in [333]. The use of such an approach for threat detection in social network for applications such as counter-terrorism is discussed in [334].

In many practical scenarios, content is available at the nodes. Some examples of such scenarios include social networks, information networks, and the web. Such content can be used in order to significantly enhance and sharpen the outlier analysis process. The work in [178] uses the similarity between the features at a node and its neighbors in order to determine linkage outliers. The work in [378] determines outlier links simultaneously with the underlying communities, with the use of a heterogeneous markov random field (HMRF) approach. Finally, the work in [180] determines community outliers, which are sets of linked that are mutually inconsistent with the underlying content. The reverse problem of determining attribute outliers by examining the hierarchical structure of data such as XML has been addressed in [264]. This is because the hierarchical structure of XML data provides semantically meaningful neighborhoods in which outliers may be found.

Significant research has also focussed on the problem of evolutionary graphs. Outliers can be defined in an almost unlimited number of

ways in temporal graphs, because of the different combinations of time and structure, which can be used in order to define regularity. Some of the earliest work focusses on measuring similarities between successive snapshots of graphs with the use of different similarity functions [365]. Another method which uses graph matching between successive snapshots for anomaly detection is discussed in [405]. This creates a time-series which can be analyzed with standard auto-regressive moving average (ARMA) methods for finding the outliers. The work in [229] uses spectral methods in order to determine anomalies in time-series of graphs. The principal component is chosen as the activity vector for that graph. This graph is then represented as a time series of activity vectors, which creates a data set of activity vector values. The principal left singular vector of this data set provides the significant direction of correlation. The activity vector for a new graph is computed, and the corresponding angle with the principal left singular vector provides the outlier score.

The work in [432] uses compact matrix decomposition in order to approximate the adjacency matrix of large sparse graphs. The primary idea underlying the work is that it is harder to approximate anomalous graphs than normal graphs. Therefore, the approximation error for each graph in a sequence of graphs is constructed. Anomaly detection is performed on this time-series of values. Other dimensionality reduction methods which are used for anomaly detection in graphs include the use of non-negative matrix factorization [452].

The use of community evolution methods is very common, since communities capture the broad patterns in the network. Therefore, a change in the community structure is used in order to model significant evolution [17, 116, 192, 194, 195, 322, 429–431]. The work in [375] uses the history of node's neighborhood in order to detect anomalies. Other common methods used are shortest-path distance change metrics [193], and pattern-based methods [64]. Some of these methods [429, 431] are specifically applicable to bipartite graphs. The determination of significant evolution in graphs can be useful in the context of a wide variety of applications such as monitoring blog communities [348], or mining traffic flow data sets [336]. In the latter case, values are associated with edges, corresponding to traffic flows. Anomalous regions are found in the network, by using the values on these edges.

In many evolutionary graphs, a significant amount of content may be available. Some examples of such data include social streams such as those created by *Twitter*. Therefore, anomaly detection in such scenarios may need to combine both linkage and content. A number of recent methods have also shown how to determine evolutionary outliers

in such social media streams [27, 362, 439]. These methods may use both linkage and content. Some methods focus on linkage [439], whereas others focus on content [362], and yet others on a combination of the two [27]. The detection of anomalous meetings between a group of people in a social network is addressed in [408] with the use of an Expectation Maximization approach.

8. Exercises

1. Download the *DBLP* bibliographic network [518]. Construct the co-authorship network, where the weights on the edges correspond to the number of publications. Extract the features n_i, e_i, w_i and λ_i, as discussed in the node outlier section of this chapter. Create pairwise plots between:

   - n_i and e_i
   - w_i and e_i
   - λ_i and w_i

   Determine the points which deviate significantly from the least squares fit for each pair. Which nodes are determined as the outliers?

2. Use an edge sampling approach in order to create k connected components in the network, where $k = 100000$. Repeat the process 100 times. Which edges have end points that repeatedly lie in different partitions?

3. Construct a *DBLP* co-authorship network for each year since 1990. Construct a normalized dot-product between the edge sets in successive years. Which are the significant change points along the different years?

4. Repeat Exercise 3 by performing PCA on the un-weighted augmented adjacency matrix graph snapshot obtained from each year of the *DBLP* data set.

5. Associate a number with each edge of *DBLP*, corresponding to the number of publications between each pair of authors. Repeat Exercise 3 by performing PCA by using these weights in the analysis.

Chapter 12

APPLICATIONS OF OUTLIER ANALYSIS

"The study and knowledge of the universe would somehow be lame and defective, were no practical results to follow."– Marcus Tullius Cicero

1. Introduction

Outlier analysis has numerous applications in a wide variety of domains such as the financial industry, quality control, fault diagnosis, intrusion detection, web analytics, and medical diagnosis. The applications of outlier analysis are so diverse, that it is impossible to cover all possibilities exhaustively in a single chapter. Therefore, the goal of this chapter is cover a wide spectrum of problem domains at a higher level, and how they map to the different techniques covered in this book. The practical issues and challenges in the context of real data sets will also be covered. This will provide a broader understanding of the issues involved in *problem domain to technique mapping*. The main applications domains covered in this chapter are as follows:

- Quality Control Applications

- Financial Applications

- Web Log Analytics

- Intrusion Detection Applications

- Medical Applications

- Text and Social Media Applications

- Earth Science Applications

In addition, a section will also be devoted to miscellaneous types of data, such as images and trajectory data. Within each domain and problem formulation, the diversity of the problems and data types are significant. Therefore, a few "core" formulations will be studied for each application domain, rather than the wide gamut of specific and detailed variations. The formulations will be mapped to broad classes of techniques covered in this book. The goal of the chapter is to teach practitioners how to *use* outlier analysis methods, by defining an appropriate mapping from problem formulation to the specific class of techniques. Some discussion will also be provided about the specific challenges of different problem domains. A discussion of the related research will also be integrated into the discussion of each application.

At this stage, a few practical insights about outlier analysis will be mentioned. These insights seem to be common across a wide variety of problem domains, and therefore provide an understanding of the areas which would benefit the most from further research:

- *Dependency-oriented data is ubiquitous:* While traditional multidimensional outlier detection is applicable in many domains, an increasing number of domains generate *dependency oriented data* such as time series, sequences, spatial data, or network data. Such data is extremely complex, and much more challenging to analyze, as compared to multidimensional data. This is because anomalies are usually defined in a contextual or collective sense in such data. The ability to distinguish between noise and anomalies is limited, because large amounts of data are required in order to obtain a sufficient level of statistical significance about the frequency properties of collective groups of data items.

- *Supervision is often critical in distinguishing between noise and application-specific anomalies:* A significant amount of research has been devoted towards unsupervised outlier analysis in the literature. However, when examining *what really works* in many applications, the presence of supervision is critical. This is because outliers found by unsupervised methods often correspond to noise, and may greatly outnumber the number of interesting anomalies. In such cases, the incorporation of supervision is critical. Even when labels are not available, *indirect* supervision can be incorporated into unsupervised methods by using domain knowledge during feature extraction and specific details of algorithm design. Furthermore, active learning methods can be used in order to efficiently *create* labels, by combining unsupervised methods with human feedback.

This chapter is organized as follows. Quality control applications are discussed in section 2. A discussion on financial applications is provided in section 3. Methods for using outlier analysis in web log analytics are discussed in section 4. The problem of outlier analysis in the context of security and intrusion detection applications is studied in section 5. Medical applications are studied in section 6. Text and social media applications are studied in section 7. Earth science applications are presented in section 8. A number of miscellaneous applications are studied in section 9. A broader discussion of the key guidelines for practitioners is provided in section 10. A description of the software resources available for the practitioner are provided in section 11. The conclusions are provided in section 12.

2.   Quality Control and Fault Detection Applications

Quality control applications arise often in the context of manufacturing. Outliers can be detected in such applications either in terms of the characteristics of individual objects, or in terms of the aggregate characteristics of the manufacturing process. Some examples of applications in such domains are discussed below.

APPLICATION 2.1 (QUALITY CONTROL) *A manufacturing process is designed to produce widgets, which are defective with probability p_i. A specific batch of widgets of size n contains q defective widgets. The goal is to determine the probability that the manufacturing process is behaving in an anomalous way.*

Discussion: This is one of the standard formulations for quality control analysis. In many variations of this problem, specific parameters of the widget (eg. physical parameters) may be tracked, and compared to an expected mean and standard deviation. In some cases, multiple parameters may be tracked simultaneously. Depending upon the specific parameters being tracked, a variety of extreme value analysis methods from Chapter 2 can be used.

- The Chernoff bounds discussed in section 2 of Chapter 2 can be used in order to provide tight bounds on the tail probabilities, when specific fractions are being tracked (eg. fraction of defective widgets).

- The Hoeffding inequality discussed in section 2 of Chapter 2 can be used in order to provide tight bounds, when strict upper and lower limits exist on tracked values.

- The *t*-value and normal value distributions discussed in section 2 of Chapter 2 can be used in order to provide approximations for the level of significance of an aggregate value (eg. fraction of defective widgets in a sample).

- In cases, where multiple parameters are tracked (eg. a combination of length, width, weight), the multivariate distance distribution methods discussed in section 3.4 of Chapter 2 may be used, either on the individual parameters (dimensions), random subsets of parameters (dimensions), or the entire set of parameters (dimensions). It is often the case that the anomalous behavior may be reflected simultaneously in many dimensions, as a result of which the use of multivariate analysis may provide better insights.

                                                                        ■

While aggregate analysis is an important application in quality control, numerous scenarios require the examination of a specific object for faults. This is related to the problem of fault diagnosis.

APPLICATION 2.2 (FAULT DETECTION AND SYSTEMS DIAGNOSIS)
A running engine or industrial system is continuously being monitored on a variety of parameters such as rotor speed, temperature, pressure, performance etc. It is desired to detect a fault in the engine system as soon as it occurs.

Discussion: The data in this domain is usually in the form of sensor data, in which continuous sensor values are tracked over time. Thus, methods for time-series analysis may be used in this scenario. However, the kind of methods being used may depend upon the specific application at hand.

- In many applications, extreme values of the sensor data may correspond to anomalies. For example, very high temperature or pressure may precede the bursting of a pipe. In such cases, even the simple extreme value analysis methods of Chapter 2 may be used without accounting for the temporal aspect of the data. However, in order to perform early detection, sudden and unusual changes are more relevant. For example, a sudden and unusual *rise* in temperature may be relevant, even when the *absolute* value of the temperature is not high. In such cases, the abrupt change detection methods discussed in section 2 of Chapter 8 may be used.

- In many applications, thousands of the time-series may be monitored, and unusual deviations in *specific* combinations of time-

series provide information about *different kinds of* anomalous behavior. Section 2 of Chapter 8 discussed the case where a sensor failure scenario could be distinguished from a pipe rupture scenario. Such scenarios almost always require supervision in order to provide the learning necessary for specific kinds of anomalies.

■ Unusual *shapes* in the time-series may often provide clues about anomalous behavior. For example, an unusual vibration in the engine system may cause oscillations in the pressure values, which are abnormal. In such cases, the unusual time-series shape detection methods of section 3 in Chapter 8 can be used. These methods can also be generalized to multivariate time-series by transformation to trajectory data, as discussed in section 3.4 of Chapter 8.

■ Unusual shapes can also be related to *specific* diagnosis of system faults by using supervised methods. In such cases, training data relating similar faults with the corresponding time series may be available. Supervised methods for unusual shape detection in time series are discussed in section 3.5 of Chapter 8.

The work in [198] designs a method for novelty detection, such as the detection of shorted turns in the field-windings of operating synchronous turbine-generators. Signature patterns of signals are extracted from the running motor, and are compared with the normal signals in order to detect novelties. A probability density estimation method for detection of abnormal conditions in engineering is discussed in [138]. A kernel method is used for the density estimation process. Kernel density-estimation methods are discussed briefly in Chapter 5 of this book.

The method in [358] uses principal component analysis to transform the data, and then determine the anomalies, by determining the points which lie far away from the primary hyper-planes of projection.

Many of the above methods can also be applied to problems such as structural damage detection, in which faults in mechanical units may be diagnosed with the use of different kinds of time-series data. PCA methods have also been used anomaly detection in spacecraft components [172]. A method which combines unsupervised and supervised learning methods for fault detection in automobile data is discussed in [173]. A detailed discussion of the use of the wavelet transformation for machine health monitoring is provided in [361]. The use of neural networks for motor fault detection is discussed in [118]. A supervised method for motor bearing damage detection was proposed in [398]. In many cases, streaming and online detection methods are desirable [9].

■

The diagnosis of computer systems also requires real-time anomaly detection techniques. However, the data in such cases is often discrete, and the methods used are typically similar to those in intrusion detection and security applications. This will be discussed in a later section of this chapter. For example, methods for system monitoring of large computer clusters are discussed in [390].

A related topic is that of structural defect detection, which attempts to determine structural defects in 2-d or 3-d objects such as a fabric, or a beam [101, 436, 437]. In such cases, measurements may be associated with each spatial location. For example, for the case of fabric fault detection, a 2-d image of the fabric may be analyzed for faults [101].

APPLICATION 2.3 (STRUCTURAL DEFECT DETECTION) *Given a 2-d or 3-d surface associated with measurements at each spatial location over time, the goal is to determine significant structural defects, either at a given snapshot in time, or significant changes which occur over time.*

Discussion: The nature of the measurements in this case are specific to the problem domain. This problem is inherently spatiotemporal, since multiple spatial measurements are available at different instants in time. Different methods are possible for defining outliers.

- Unusual (spatial) changes in the attribute values on the basis of spatial locations can be used in order to detect anomalies. For example, the neighborhood algorithms discussed in section 2 of Chapter 10 can be used in order to detect outliers.

- Unusual shapes in the images implied by the attribute values often provide insights about significant patterns of defects in the data. These methods are discussed in section 5 of Chapter 10.

- The spatial analysis in the two cases can be combined with temporal analysis in order to determine significant changes in the underlying data. This corresponds to the methods discussed in section 6 of Chapter 10.

- In many cases, previous examples of specific defects may be available. In such cases, the supervised techniques discussed in section 7 of Chapter 10 may be used.

Methods for structural defect detection are discussed in [101, 224, 436, 437].

■

A broad review of fault detection methods is provided in [453].

3.   Financial Applications

Financial fraud is one of the more common applications of outlier analysis. Such outliers may arise in the context of credit card fraud, insurance transactions, and insider trading. This section will discuss a number of financial applications in the context of outlier analysis.

APPLICATION 3.1 (CREDIT CARD FRAUD) *A credit card company maintains the data corresponding to the card transactions by the different users. Each transaction corresponds to a set of attributes corresponding to the user identifier, amount spent, geographical location etc. The card company may also have labeled data containing previous examples of fraudulent transactions. It is desirable to determine fraudulent transactions from the data.*

Discussion: One desirable aspect of credit card applications is that labeled data is often available in order to relate the transactions with the underlying anomalies. Nevertheless, both supervised and unsupervised methods can be used for anomaly detection in such cases.

Many domain specific characteristics of the data are used for fraud detection. For example, it is well known that large absolute values of the transaction amounts may correspond to anomalies. The most common technique is to build user profiles on short segments of transaction sequences. Typically, the ordering among a short segment of the transactions is immaterial. If desired, a single transaction of the user can also be used. Either a single transaction or a short sequence of transactions can be converted into a feature vector, which is compared to the user's profile. The key is to design a similarity function, which can encode the wide diversity of attribute types, the collective profile within a short segment, and domain-specific knowledge (eg. higher values of transactions or sudden bursts of high-value transactions are more likely to be fraudulent). It is also possible to use geographical location, in order to determine the anomalousness of a sequence of transactions from the same user with respect to other sequences in the same spatial location.

The major challenge with anomaly detection in credit card data, is that false positives are extremely common, and false negatives are expensive, even when rare. In other words, the ROC curve usually suggests very noisy behavior of purely unsupervised methods. The quality of the inference can be improved in two ways, both of which incorporate some form of supervision:

- Domain specific knowledge needs to be encoded into the similarity function, in order to account for the differential nature of fraudulent transactions.

■ When labels are available, supervision should be used in order to relate the profiles to fraudulent behavior.

In many cases, the automated analysis is combined with manual inspection in order to determine significant cases of fraud. Recently, discrete sequence methods of Chapter 9 based on Hidden Markov Models have also been used [422]. In these methods, symbolic values are extracted by discretizing the credit card amounts. A survey on supervised fraud detection methods is provided in [367]. The issue of class imbalance in supervised fraud detection methods is discussed in [368]. A wide variety of methods [35, 106, 184, 367, 425, 426] are available for credit-card fraud detection, though the general experience has been that supervised methods are the most effective. This is not surprising, since supervised methods are better able to distinguish between true anomalies and noise.

■

APPLICATION 3.2 (INSURANCE CLAIM FRAUD) *In this case, claims are made by different entities on the basis of insurance policies. Significant anomalies need to be discovered from the data on this basis.*

Discussion: While this application shares some resemblance to credit card fraud detection at a higher level, it is also significantly different in many ways. For example, user-specific profiles cannot be constructed, since a particular user may rarely make a claim. On the other hand, repeated claims by a single user is often an indicator of fraud, and should be incorporated as a feature during pre-processing. Unlike credit fraud applications, geographical location is often *contextually* not relevant.

The problem is essentially an application of multidimensional (point) anomaly detection, once a multidimensional representation of the claims has been created. The key step is to extract the correct features from the insurance claim documents, which can be used in order to create an unsupervised or supervised anomaly detection system. Feature extraction in insurance claim scenarios is highly domain specific, since it requires the identification of indicators which are highly specific to the particular kind of claim. For example, in a life-insurance scenario, a low lag between the initiation of the policy, and the death of the subject is sometimes correlated with homicide. In a medical insurance claim scenario, the statistical distribution of the claims over different types of diseases coming from a single medical provider may be skewed, when the provider is engaging in fraud. Depending upon the application, such features need to be extracted in a domain-specific way. Therefore, any of the methods in Chapters 2, 3, 4, 5 and 6 may be used, once the feature extraction phase has been performed.

Labels are usually available since previous examples of fraud are available. In order to obtain high quality prediction, it is critical to encode the information about previous examples, either in the form of the learning algorithms discussed in Chapter 6, or indirectly by using feature representations which distinguish fraudulent claims from normal ones. Methods for insurance fraud detection are discussed in [137, 367, 456]. In particular, a comprehensive bibliography may be found in [367].

■

Many financial organizations also track the user behavior at their online web sites. These correspond to discrete sequences, which can be analyzed in order to determine significant instances of fraud. These cases will be analyzed in section 4 of this chapter on web log analytics.

APPLICATION 3.3 (STOCK MARKET ANOMALIES) *The financial tickers of the different stocks and options correspond to time-series data streams. In some cases, significant anomalies may be created by external events. The* early *detection of such events may be useful in the determination of* unknown *influencing factors such as insider trading, or automated stock trading glitches (eg. the flash crash of May 2010).*

Discussion: In many cases, external information such as news streams (eg *Google News*) are also available for event detection. This is particularly useful for applications such as insider trading detection, where unusual temporal ordering between events in the news and events in the stock tickers can be used in order detect insider trading. For example, if an anomalous change (in value or transaction volume) of the relevant stock ticker precedes an event for the ticker in the news stream, then this can be used as an indicator of insider trading.

In other cases, such as the unusual behavior of the stock market during the flash crash of May 6, 2010, direct time-series analysis may be used. Such methods are discussed in Chapter 8. Both deviation-based contextual point anomalies and time-series shape-based collective anomalies provide insights about the unusual interactions. Deviation-based anomalies are more useful for early detection, whereas shape-based anomalies are more useful for detailed diagnosis, which is slightly delayed. In many cases, it may be desirable to use streaming methods [9] in order to determine the anomalies in real time. Methods for early detection of insider trading in financial markets are discussed in [142]. Some of the methods for multidimensional change detection discussed in Chapter 8 are useful for tracking other aspects of stock activity such as the specific stock orders, or the volume distribution of stock orders. Methods for distribution change detection in stock order data streams are discussed in [313].

Another interesting method for detecting *regime anomalies* [61] from time-series data streams can be used in order to detect how the dependencies among different streams has changed suddenly. This provides an idea of scenarios, where the relationships among the different stocks has changed over time.

■

Financial entities often interact with one another. Examples include producers and suppliers, customers with sellers, and customers with each other. In such cases, anomalous patterns of interaction can provide useful insights. This leads to the interesting problem of detecting anomalies in financial and customer interaction networks.

APPLICATION 3.4 (FINANCIAL INTERACTION NETWORKS) *A set of financial entities V are continuously interacting with one another over time. Values on the edges may correspond to the intensity or volume of the interactions. Unusual anomalies may need to be determined in such cases.*

Discussion: Financial interaction networks are ubiquitous, and the interactions between different financial participants are often tracked in order to obtain competitive knowledge about the interactions. In some cases, such as mobile phone fraud, the interactions may not specifically correspond to financial transactions, but an interaction between two customers. In most of these cases, values on the edges are available for analytical purposes. The temporal change detection methods in section 5 of Chapter 11 can be used in order to determine relevant regions of change in the network. The challenge in using such methods is that the values on the edges are often critical to the anomaly detection process. For example, high values on the transactions may correspond to anomalies. It is relatively easy to generalize the spectral methods of Chapter 11 to include the values on the edges in the analysis. Such methods are discussed in [229, 429, 430, 432]. Different methods for fraud detection in the context of mobile phone networks are discussed in [225, 350, 367, 368, 444].

■

4.   Web Log Analytics

Web logs often contain significant information about security breaches and other kinds of anomalous activity. For example, a bank may keep logs of its web site accesses. Unusual patterns of accesses may correspond to anomalous activity.

APPLICATION 4.1 (WEB LOG ANOMALIES) *Given a sequence of accesses in a web log, the goal is to determine the unusual patterns of accesses in this log.*

Discussion: Web logs are typically pre-processed into a set of user-specific discrete sequences. These discrete sequences correspond to the identifiers of the pages accessed by the users. Many challenges arise during pre-processing, since users can often be distinguished only at the level of their IP-addresses. Nevertheless, even in such cases, user-sessions can often be mined from the logs. The initial phase of pre-processing is crucial in such applications, because a single undifferentiated log is provided. This log needs to be decomposed into user-sessions, and then further decomposed into test sequences, and comparison units, as discussed in Chapter 9. General issues related to web log data preparation are discussed in [122].

In many cases, additional domain knowledge is available about *relevant* sequences. For example, a repeated sequence of accesses to *login* and *password* pages may be indicative of anomalous behavior. In other cases, examples of anomalous discrete sequences may be available. Where possible, such domain knowledge should always be used, because it significantly improves the quality of the underlying results.

Numerous methods discussed in Chapter 9 are applicable to this case, depending upon the kinds of outliers that need to be found.

- Position outliers can be used in order to determine unexpected accesses. These are contextual anomalies detected by the method, and correspond to a single unpredictable access, which is an outlier because of its *relationship* to adjacent and neighboring values. Markovian and rule-based models are typically used for outlier detection in discrete sequences. Such methods are useful for early anomaly detection, when a single unexpected web access is sufficient to arouse suspicion.

- Combination outliers are useful for determining unusual subsequences in the test sequence. This can be achieved using either unsupervised or supervised methods. In the case of supervised methods, the extraction of relevant features such as k-grams is crucial for effective anomaly detection. In the case of unsupervised techniques, window-based nearest neighbor and Hidden Markov Models are typically used.

Methods for anomaly detection in web logs are discussed in [140, 275].

Web logs are often analyzed in the context of a wide variety of security and intrusion detection algorithms. These methods will be discussed in this section. In the former case, operating system call traces are analyzed on a particular computer system, whereas in the latter case, the network data is analyzed for anomalies.

5.   Intrusion and Security Applications

Intrusions correspond to different kinds of malicious security violations in a computer system. The data is typically streaming, and arrives at a high rate. Intrusions correspond to anomalous events, which need to be inferred from the underlying data. Denning [136] classified intrusions into *host-based intrusion detection systems*, and *network-based intrusion detection systems*. These are somewhat different cases both from the perspective of data representation and temporal locality. In general, the former involves the analysis of discrete sequences with high temporal correlations, whereas the latter involves the analysis of multidimensional streams with (relatively) limited temporal correlations. Therefore, different models are typically used in these cases. Each of these cases will be discussed in this section.

APPLICATION 5.1 (HOST-BASED INTRUSIONS) *Operating   system   call traces are available in a computer system, which are symbolic sequences. Anomalous subsequences in these traces correspond to malicious computer programs. It is desired to determine anomalous sequences from these traces.*

Discussion: The data in this case is similar to web logs at a conceptual level, in that it corresponds to symbolic sequences. In this case, operating system call traces are used instead of web log traces. The calls could correspond to either operating system calls or user calls. Thus, the calls form the base alphabet Σ over which the mining is performed. Different kinds of programs execute different sequential combinations of calls. Therefore, the sequential ordering of the calls provides critical information in order to distinguish between normal and malicious programs. These calls could either be at the user command level, or they could be at the operating system level. The latter is much more granular, and that can sometimes make it more difficult to mine such sequences. Some examples of user level calls in an operating system are illustrated in Table 12.1.

During the feature extraction phase, the logs are transformed into symbolic sequences. In many cases, when the commands are coming from multiple sources, they may need to be separated out into their different hostnames. For example, in the case of Table 12.1, the sequences

Time	Hostname	Command	Arg1	Arg2
AM	Sayani-T61	mkdir	dir1	
AM	Guardian	more	file1	
AM	Guardian	cp	file1	file2
AM	Sayani-T61	cd	dir1	
AM	Guardian	find	dir2	-print
AM	Sayani-T61	vi	file1	

Table 12.1. Some examples of user commands in a UNIX system

of commands from the hostnames *Guardian* and *Sayani-T61* need to be separated out into different sequences in order to examine the malicious behavior of a particular host. This is similar to web log analytics, in which sequences which are specific to each web user are constructed in the pre-processing phase. Sometimes, the choice of the symbols used in the sequence may also depend upon the application at hand. For example, should only the user command be used, or should some combination of user command and argument be used? These choices are highly application-specific, and the quality of the final results may be sensitive to such choices.

Subsequently, any of the discrete sequence methods discussed in Chapter 9 can be used. In fact, the different kinds of scenarios are very similar to those in Application 4.1 on web log analytics. The reader may refer to the details of the specific methodologies used in Application 4.1. In spite of the very different data domains in these cases, it is interesting to see that the underlying methods are often interchangeable.

A comparison of different methods for intrusion detection in these scenarios is provided in [130]. Numerous methods have been proposed in the literature for this scenario, and are discussed in [130, 155, 163–167, 182, 183, 222, 284–286, 294–296].

∎

A second class of methods is that of network intrusion detection systems, in which the intrusions are inferred from network data. The data on the network can be of different types, depending upon the level of abstraction at which it is presented. For example, the data could correspond to the underlying packets on the network, and the intrusions may result in subtle changes in the multidimensional features extracted from these packets.

APPLICATION 5.2 (NETWORK INTRUSION DETECTION) *Given a stream of network packets or data records, the goal is to determine network intrusions.*

Duration	Protocol	Service	Src_Bytes	Dest_Bytes
5	tcp	telnet	183	3855
5	tcp	telnet	183	3855
0	tcp	http	298	2239
0	udp	private	105	146
0	udp	domain_u	44	44
0	tcp	http	188	2199
0	icmp	ecr_i	508	0

Table 12.2.   Examples of five basic features from the network connection records from the *KDD Cup 1999 Network Intrusion Data Set* [169]

Discussion: The temporal relationships between data records is much weaker in this case, than in the case of host-based systems where the sequential ordering of calls is critical in identifying intrusions. Furthermore, each individual record in this case is multidimensional, and contains the features extracted from the unit of network data (eg. packet), or raw *tcpdump* data. For example, a common feature used is the number of bytes transferred, which is continuous. Other attributes are discrete. Thus, this problem can be modeled as a multidimensional stream of records, containing both continuous and categorical attributes. An example of a network intrusion data set is illustrated in Figure 12.2. Only five of the basic features are shown. This is the well known *KDD Cup 1999 Intrusion detection data set* [169], which contains a combination of symbolic and continuous attributes. These features of three types corresponding to the basic characteristics of the connections (service, protocol, bytes transferred etc.), the content characteristics of the connections suggested by domain knowledge (eg. number of "hot" indicators, failed login attempts), and the traffic characteristics (eg. number of connections, or the number of connections with specific kinds of errors).

Most of the unsupervised multidimensional outlier detection methods can be generalized to this case. Furthermore, in cases where stream processing is required, the multidimensional streaming outlier detection methods of Chapter 8 may be used. In particular, aggregate change-points may often be helpful in identifying network-wide traffic anomalies. Such change points often correspond to network intrusions and attacks [162, 280, 281]. Since the data is often of a mixed nature, many of these algorithms need to be modified using the general methodologies discussed in Chapter 7 for adapting unsupervised algorithms to mixed data sets.

In some cases, a subset of the data may be labeled, and may correspond to either normal data, or intrusions. The labeled intrusions can

never be exhaustive, since new intrusions may always arise over time. Nevertheless, labeled data about existing intrusions is useful for identifying repeating attacks. At the same time, it is important to also detect novel classes as new intrusions arise. Such scenarios can be addressed using the streaming and supervised novel-class detection methods discussed in section 4.3 of Chapter 8. Methods for network intrusion detection are discussed in [43, 44, 55–57, 119, 155, 236, 297, 276, 277, 320, 321, 351, 399, 448, 489]. A comparative study of network intrusion detection schemes may be found in [290].

■

6.   Medical Applications

Medical applications typically uses different kinds of diagnostic tools for predictive modeling. Two of the most common kinds of data which are encountered for predictive modeling of medical data are in the form of sensor data (eg. ECG), and spatial data (eg. PET scans). Both of these are different kinds of contextual data. Each of these different cases will be addressed by a different application definition.

APPLICATION 6.1 (MEDICAL SENSOR DIAGNOSTICS) *Given a set of sensor readings from a given patient, the goal is to determine if the patient has a disease condition.*

Discussion: Both supervised and unsupervised methods can be used in order to process medical data. For the unsupervised case, the problem formulation is similar to that of fault diagnosis, except that the nature of the sensor readings are specific to the medical domain. Therefore, all the methods from Chapter 8 can be used for this case as well. In the simplest case, extreme or unexpected value analysis on medical time-series or data distributions may be used [226, 288, 384, 415] in order to determine anomalous values. An approach which uses a probabilistic mixture model is discussed in [438]. Time-series containing subsequences of unusual shapes [304] may also be useful for identifying more complex anomalous conditions.

In the context of medical data, the cost of missing a positive is high, and the diagnosis needs to be specific. Furthermore, any anomalous conditions may need to be reported in real time. In many cases, a specific diagnosis may be distinguished from other spurious diagnosis by using the signals from multiple sensor data streams. In this context, a supervised method for deviation-based anomaly detection in multivariate time series data streams was proposed in [9]. Methods for shape-based supervised anomaly detection are discussed in Chapter 8.

Supervised methods are particularly desirable in the medical domain because of the specificity requirements of a diagnosis. In many cases, expert knowledge [40] may need to be combined with the mining algorithm in order to ensure the most effective results. Semi-supervised methods for medical time-series classification are discussed in [462]. The problem of supervised shape discovery in time-series is discussed in section 3.5 of Chapter 8.

■

Another common diagnostic tool used in the medical domain is that of *imaging*. In these cases, an MRI or PET scan is used to create a 2-dimensional or 3-dimensional image of a part of the body such as the brain. Anomalies in these shapes correspond to significant medical conditions.

APPLICATION 6.2 (MEDICAL IMAGING DIAGNOSTICS) *Given a multi-dimensional image of an affected body part, the goal is to determine whether the patient has a disease condition.*

Discussion: This is a classic example of a spatial application, which contains both contextual and behavioral attributes. The discovery of anomalies in such data has been addressed extensively in Chapter 10. The most crucial part of the feature extraction is to convert [504] the shape into a time-series using the methods discussed in section 5 of Chapter 10. Subsequently, a variety of unsupervised or supervised methods can be applied to the extracted time series. The problem typically reduces to one of the following formulations:

- Anomalous shapes may correspond to specific disease conditions such as tumors, multiple sclerosis lesions, mammography, or the degraded brain regions of an Alzheimer patient [374, 466, 418, 446]. In the semi-supervised case, examples of normal shapes may be available, and it may be desirable to determine shapes which are very different from these normal profiles. The method of [504] may be used to convert the shapes into time-series, and then anomaly detection can be applied to this time series. For example, the approach in [469] may be used in order to discover such anomalies. This method has also been described in section 5 of Chapter 10.

- In many cases, previous examples of anomalous regions may be available. This can be used in order to train a classifier to learn the relationship between the shape and the specific disease condition. A variety of shape learning methods [54, 92, 316, 504] are available

for labeled data. A brief review of such techniques is provided in section 7.1 of Chapter 10.

■ In some cases, a temporal sequence of images from the same patient over many years may be available, and it may be desirable to determine usual *changes* in the shapes of the images [68]. This corresponds to shape change detection methods discussed in section 6.2 of Chapter 10.

A variety of shape analysis methods have been used frequently in the medical domain for image diagnostics [68, 206, 374, 466, 393, 418, 446, 447].

■

7. Text and Social Media Applications

Text and social media applications are extremely common because of the ubiquity of text data in social interactions such as email, the web, and blogs. Some of these methods have already been discussed in Chapter 7.

APPLICATION 7.1 (EVENT DETECTION IN TEXT AND SOCIAL MEDIA) *Given a document corpus $\mathcal{D}$, the goal in unusual topic detection is to determine unusual documents, which differ significantly from the trend. In first story detection, a stream of documents is available, and it is desirable to determine unusual events corresponding to new topics in the stream of documents. In the context of social media applications, such streams may be generated by user activities such as tweets.*

Discussion: This scenario has already been discussed extensively in Chapter 7. Both supervised [484, 486] and unsupervised methods [26, 37–39, 77, 254, 420, 421, 485, 503, 515] may be used. The reader is referred to Chapter 7 for details.

A particularly important special case is that of social media streams in which the user activities such as tweets may provide early knowledge of unusual events. In many cases, unusual events in localized regions may show up in social media feeds well before traditional news media, because of the wider authorship of social media sites. Such events will typically be manifested as changes in the topical and linkage distributions of the social media feeds. Both supervised and unsupervised techniques are relevant in such cases, depending upon the availability of training data. In the supervised case, it may be desirable to determine events of specific types. Methods for finding unusual events or changes in text and social

streams are discussed in [27, 362, 458]. Both supervised and unsupervised methods for event detection in social media streams are discussed in [27].

∎

Another common application in email streams is to detect spam in the emails.

APPLICATION 7.2 (SPAM EMAIL) *Given a stream of emails, the goal is determine the subset of emails which correspond to spam.*

Discussion: While unsupervised methods for unusual topic detection can be used, the results are often likely to be inaccurate. While spam is still a small fraction of the mail in most cases, the volume is large enough to make it difficult to detect with unsupervised methods. This case is best addressed with the use of supervised methods, where the specific features of the emails are learned, and related to spam labels in the training data. Any of a variety of methods for text classification [21] can be used.

A lot of additional domain knowledge is available, which helps determine whether a particular email message is junk or not. For example, some common characteristics of the email which would make an email to be more or less likely to be junk are as follows:

- The domain of the sender such as *.edu* or *.com* can make an email to be more or less likely to be junk.

- Phrases such as *"Free Money"* or over emphasized punctuation such as "!!!" can make an email more likely to be junk.

- Whether the recipient of the message was a particular user, or a mailing list can influence the underlying likelihood.

The Bayes classifier for text provides a natural way to incorporate such additional information into the classification process, by creating new features for each of these characteristics. A method such as this has been discussed in [389]. A survey of methods for email spam filtering may be found in [67, 124].

∎

In many social networks, a significant percentage of the links are noisy, and may not provide any useful insights for analysis. Such links are often caused by spam links on the web to increase search engine rating, or low quality links across weakly related entities.

APPLICATION 7.3 (NOISY AND SPAM LINKS) *Given a social network with content at the nodes, determine the noisy and spam links with the use of structure and content information.*

Discussion: This problem is discussed extensively in the sections 3.2 and 4 of Chapter 11. The first section discusses methods for determining linkage outliers on the basis of structure only, whereas the second section discusses methods for determining linkage outliers with the use of both structure and content. Such edges in a social network correspond to relationships which are weak, and often harm the effectiveness of social media algorithms. Techniques for determining such outliers are discussed in [15, 161, 178, 378, 180]. In many cases, supervised methods may be used in order to learn link spam, when labeled information is available.

∎

Many forms of networks such as blogs, bibliographic and social networks contain both text and linkage information. It may be helpful to discover significant patterns of change in such networks.

APPLICATION 7.4 (ANOMALOUS ACTIVITY IN SOCIAL NETWORKS) *Given an evolving network with associated text content at the nodes, the goal is to determine the anomalous regions of activity or change in the network.*

Discussion: This problem is related to that of evolution of communities in the underlying network. It is possible to use a purely structural approach with the use of the community change analysis methods discussed in section 5 in Chapter 11. In the case of blogs and social networks, a significant amount of content is also available in the network. In such cases, community detection algorithms can be enhanced with the use of node content [11]. In addition, a variety of spectral methods [229, 429, 430, 432] can be used, when it is required to use only the linkage structure for analysis. A method which specifically monitors evolving blogs for significant change is discussed in [348].

Social networks are particularly useful tools for the discovery of threat activity such as terrorist interactions. A method for finding threat activity in social networks with the use of eigenspace analysis was proposed in [334].

∎

8. Earth Science Applications

Outlier detection is used in numerous weather, climate, or vegetation cover applications, where anomalous regions are detected in spatial

data either at a single snapshot or over time. Therefore, many of these applications are spatial or spatiotemporal in nature. For example, sea surface temperatures are often tracked in order to determine significant and anomalous weather patterns.

APPLICATION 8.1 (SEA SURFACE TEMPERATURE ANOMALIES) *The temperatures on the sea surface are tracked continuously over time. It is desired to determine:*

(a) *unusual localized spatial variations in temperature on the sea surface,*

(b) *regions on the sea surface containing unusual shapes with homogeneous temperature,*

(c) *sudden and unexpected changes in sea surface temperature which are local to a specific region, and*

(d) *the relationship of the spatial temperature patterns to known weather events for predictive modeling.*

Discussion: This is a typical spatial or spatiotemporal formulation, which arises often in the context of meteorological applications. In this case, the spatial and temporal coordinates are the contextual attributes, whereas the temperature is the behavioral attribute. It requires the determination of both contextual and collective outliers from the data. The determination of unusual spatial variations can be performed by finding neighborhood based outliers. Such outliers are discussed in section 2 of Chapter 10.

Unusual shapes on the sea-surface temperature profile can be performed by applying the unusual shape discovery method discussed in section 5 of Chapter 10. The temperatures may need to be discretized into buckets in order to convert the continuous values into discrete shape contours. The contour of a region with the same discretized value may be used for the anomaly detection process.

The neighborhood-based spatiotemporal change detection algorithms of section 6 in Chapter 10 may be used in order to determine significant outliers. These correspond to specific *points* in space and time at which there are unusual temporal *or* spatial variations. In some cases, it may be desirable to determine significant changes in the *shapes* of temperature patterns. The detection of significant changes in the patterns can be performed by using the shape change detection methods discussed in section 6.2 of Chapter 10.

Such characteristic patterns in different kinds of behavioral attributes such as temperature, pressure or humidity can often be related to unusual weather events such as cyclones. However, such anomalies are

best determined with the use of supervised methods in which previous training data relating the weather patterns to the spatiotemporal data is available. Supervised methods for classification of such data are discussed in detail in [54, 92, 316, 504].

■

While the aforementioned application was presented with the use of temperature as a behavioral attribute, a variety of other attributes such as pressure, or humidity may be tracked for anomaly detection. As an example, the work in [472] tracks precipitation patterns in order to determine unusual regions of change. Methods for finding region outliers in meteorological data are discussed in [510]. In many cases, multiple behavioral attributes may be available. This is a more challenging case, because it is desired to determine unusual combinations of behavioral attributes. In such cases, a simple approach is to perform the analysis separately on each attribute and combine the anomaly scores. The real-time spatiotemporal analysis of weather patterns can provide predictions of significant events such as hurricanes.

A closely related problem is that of *land cover anomalies*. The type of land cover or vegetation is often tracked with the use of remote sensing. Virtually, all the aforementioned methods for finding meteorological anomalies can also be applied to the problem of land-cover anomalies.

Application 8.2 (Land Cover Anomalies) *The land cover at different spatial locations are tracked continuously over time. It is desired to determine:*

(a) *unusual localized spatial variations in land cover type,*

(b) *regions containing unusual shapes with homogeneous land cover,*

(c) *sudden and unexpected changes in land cover which are local to a specific region, and*

(d) *the use of changes in spatial land cover patterns to uncover unusual and unknown geological, climate, human or wild life activity.*

Discussion: This case is virtually identical to that of uncovering unusual sea-surface temperature anomalies. The major difference is that the land-cover type is the behavioral attribute, which may be discrete. Therefore, the key is to design a similarity function which relates different land cover types to one another. For example, certain kinds of land cover are more likely to be adjacently located than others. In order to determine such similarity values, the methods [126] for contextual similarity discussed in section 4.2 of Chapter 6 may be used. Once such

contextual similarity measures have been determined, they can be used in order to determine significant point changes in the data. An example of a recent application to determine significant vegetation changes is discussed in [287]. Methods which correlate land cover changes with other kinds of parameters such as climate are discussed in [52, 385].

■

9.   Miscellaneous Applications

This section briefly covers miscellaneous applications for outlier detection, which do not belong to any of the aforementioned categories. Some of these applications are discussed below.

APPLICATION 9.1 (DATA CLEANING) *Given a data set, remove discordants from it. Correct any errors in the data if possible.*

Discussion: This is one of the classical applications of outlier analysis, and is often addressed effectively with unsupervised methods. Virtually all the unsupervised methods discussed in this book can be used for noise *removal.* Many of the autoregressive models introduced in Chapter 8 are used for removal of erroneous values from sensor data. For example, the removal of noisy links in social networks can be considered a data cleaning application. Such methods are discussed in Chapter 11.

For noise *correction*, methods such as PCA can provide the best insights. For example, it has been shown in [18, 355] that the use of PCA and SVD methods can be used in order to improve the representation quality of data sets for mining and retrieval. Methods for removing outliers in the context of regression analysis are discussed in [387].

■

A number of applications are designed to determine anomalies in spatiotemporal data. Such data may correspond to wild life movement patterns, or vehicular movement patterns.

APPLICATION 9.2 (TRAFFIC AND MOVEMENT PATTERNS) *Given a set of entities with trajectory patterns, determine significant outliers from the patterns.*

Discussion: This problem arises often in the context of either tracking wildlife with RFID tags, or tracking vehicles with GPS receivers. This problem can be addressed using either spatiotemporal trajectory mining methods, or by network mining methods, depending upon how the data is represented. For example, when it is desirable to determine anomalies *entities* in terms of movement patterns, the trajectory mining methods

[292] discussed in section 6.1 of Chapter 10 may be used. In some cases, supervision may be used [300] in order to improve the quality of the discovered patterns. Online algorithms for finding outliers in movement patterns are proposed in [83].

In many cases such as traffic data, the movement values are available on an aggregated basis because of privacy concerns. Furthermore, the movement patterns are often associated with a network corresponding to the road network in the underlying data. In such cases, only the flow values on the different road segments may be available. Significant regions of congestion or anomalous behavior may need to be identified. Methods for finding such anomalies are discussed in [336].

■

One of the most common domains for anomaly detection is image analytics. This was discussed earlier in the context of medical image diagnostics.

APPLICATION 9.3 (IMAGE DATA) *Given a set of (possibly labeled) images, or a temporal sequences of almost identical images, it is desirable to determine:*

(a) *images with anomalous shapes,*

(b) *in the cases, where temporal snapshots are available, the goal is to determine significant changes in the underlying patterns, and*

(c) *in the cases, where some images are labeled with a rare class, it is desired to predict this class with the use of the training data.*

Discussion: The techniques used for this case are similar to those discussed earlier in this chapter in the context of the medical image diagnostics application. Generic image representations also have a number of attributes such as color or texture, which can be leveraged in order to determine anomalies. As discussed earlier, image diagnosis methods are used frequently in the medical domain for determination of anomalies. In the context of anomalous shape discovery, the feature extraction for *shape representation* is the most important. An example was discussed in section 5 of Chapter 10, where it was shown how to convert a shape into a time-series for further analysis. An extensive discussion of such feature transformation methods is provided in [504]. General references on image outlier detection may be found in [68, 206, 374, 466, 393, 418, 446, 447].

■

Numerous other applications of outlier analysis may be found in the domains of aviation safety [128], internet routing updates [373], malicious URL detection [319, 509], disease outbreaks [465], spatial linking of criminal incidents [311], failure management of large computer clusters [390], astronomical data [147, 213, 477], disturbance events in terrestrial ecosystems [370] and biological sequences [308].

10.   Guidelines for the Practitioner

The examples discussed in this chapter illustrate that the diversity in applications in rather large across different domains. However, many of these models map into the same set of problems. Some critical observations which arise in the context of outlier detection are as follows:

- *Data normalization is important:* A common mistake which is made by many practitioners is to forget to normalize the data before applying outlier analysis algorithms. Consider an application containing an *Age* attribute (less than a hundred), and a *Salary* attribute (in order of tens of thousands). The use of proximity-based or linear models on such data (without normalization) will be dominated by the *Salary* attribute, and the *Age* attribute will be almost ignored. Typically, each attribute value needs to be divided by its standard deviation (over the entire data set). This ensures that the different attributes are given an equal level of importance in the outlier analysis process.

- *Noise vs interesting anomalies:* Most of the application domains contain data, which is noisy, incomplete or otherwise has errors. For example, sensor data often contains noise because of defects in transmission, or failure. As discussed in this chapter, data cleaning is itself a key application of outlier analysis. Therefore, it is critical to design a pre-processing phase which can filter out or correct such noise from the data, where possible. This can be achieved using a variety of domain-specific methods, which have knowledge of the noise generation process. For example, in the context of sensor data, such noise can often be corrected or filtered by a variety of methods [19]. Nevertheless, in many cases, the use of such filtering methods can also mask interesting anomalies in the data.

- *The feature extraction phase is crucial:* In many domains, the base data is not necessarily specified in a way which can be used directly with an outlier analysis application. For example, in an insurance application, the documents containing details of the claims may be available. In a credit card fraud application, raw transaction

data may be available. In such cases, feature extraction *should implicitly use domain knowledge* as far as possible. For example, when it is known that high values on the credit card transactions are more important, then the feature corresponding to transaction value should be incorporated. While numerous dedicated feature selection methods exist for other data mining problems such as clustering and classification, this does not seem to be the case for outlier analysis. This is because feature selection methods require the determination of aggregate trends relating the features to application-specific aspects of the data. On the other hand, since outliers are based on rare observations, aggregate trends are hard to determine in a way which would be relevant for outlier analysis. Therefore, the incorporation of a domain-specific understanding is often the only way by which meaningful features can be extracted for outlier analysis. A proper selection of features can also help in distinguishing noise from true anomalies.

- *Domain knowledge is often easy to incorporate into unsupervised algorithms:* One of the challenges of outlier analysis is that labeled data is rarely available. However, an *indirect* form of supervision is to incorporate domain knowledge into unsupervised algorithms. Typically such changes require minor modifications to the details of the underlying algorithm such as the similarity function design, or Hidden Markov Model design. Some examples of incorporating domain-specific knowledge are as follows.

  - In a credit card fraud application, the absolute amount spent is known to be a key indicator of fraudulent behavior. A k-nearest neighbor algorithm can be modified, so as to treat neighbors with higher or lower values of the amount spent in a purchase in a differential way. This knowledge can be incorporated directly into the distance function, without making any other change to the algorithm.

  - In a security monitoring application, specific sequences such as *login password login password* may be known to be indicative of attacks. Such sequences can be provided higher importance during the construction of the comparison units for sequence anomaly detection.

  - The states of a Hidden Markov Model should reflect an understanding of the process, rather than using a black box k-state model, in which all transition probabilities are learned.

■ *Labeled Data should be used where possible:* This is the easiest way to distinguish between noise and anomalies. *Even a small amount of labeled data* can *significantly* improve the effectiveness of outlier analysis algorithms. Unfortunately, in many real applications, labeled data is not available.

■ *Exploratory and visual analysis can be helpful at all stages of outlier analysis:* One challenge in outlier analysis is that it is often difficult to know which model may work most effectively for a given problem. Should a proximity-based model be used, should a linear model be used, or should a subspace model be used? In this context, visual analysis of the kind introduced at the beginning of Chapter 3, can provide some insights about the distribution of the data, and which model may work best for a particular application.

■ *A human in the loop can more easily generate labels in conjunction with unsupervised outlier analysis algorithms:* It is hard to generate labeled data in outlier analysis algorithms, because anomalies are rare, and therefore positive examples are often hard to obtain. Furthermore, the manual examination of large amounts of data for anomalies is akin to searching for a needle in a haystack. However, unsupervised and supervised algorithms can be used in an iterative way in conjunction with a human in the loop in order to generate labels. This corresponds to the active learning framework discussed in Chapter 6. Such frameworks can be extremely useful in converting unsupervised outlier detection problems to supervised rare class detection problems.

The above recommendations seem to suggest that the incorporation of supervision and domain knowledge is possible, even when fully labeled data is not available. A careful domain-specific design of the feature selection and algorithmic processes is critical in obtaining the most informative outliers.

11.   Resources for the Practitioner

Since the problem of outlier analysis is one of the key problems in data mining, a significant number of software resources exist for this problem. The software available for this problem is both commercial and open-source. Note that the outlier analysis problem is addressed using both supervised and unsupervised methods. A significantly larger number of packages exist for the supervised version of the problem, since it is directly related to the problem of classification, which is a much broader

field. In this section, the key resources from these two perspectives will be summarized.

A significant number of open source packages exist in the literature for different variations of the problem. A meta-repository containing a description of the different resources for both unsupervised and supervised learning is the *KDD Nuggets* website [519, 520]. The *Weka* repository [521] is a large general purpose repository, containing different kinds of data mining software for clustering, classification and outlier analysis. In addition, the *ELKI* repository [2] for outlier analysis contains an implementation of many of the advanced algorithms discussed in this book such as *LOF* and its variations, *LOCI*, EM-methods, distance-based methods, and subspace methods. Numerous methods for spatial outlier detection are contained in the same repository. Resources for different components of supervised and unsupervised outlier analysis are available from the UCR time-series classification and clustering page [522], and the Symbolic Aggregate Approximation [523] page.

A significant amount of commercial software is also available for outlier analysis. An example is the *IBM Proventia Network Anomaly Detection System* [524] for network intrusion detection. The *IBM SPSS Workbench* [525] has numerous tools, which can be used for outlier detection in both temporal and non-temporal data. In particular, *IBM SPSS Statistics* [526] contains a significant number of tools which can be used to build models for outlier analysis. The *Oracle Data Miner* [529] has significant data mining capabilities including anomaly detection. *WizSoft* software [530] has designed a software *WizRule*, which can be used for fraud and anomaly detection. *SAS* [527] has developed many different software packages for general statistical modeling and anomaly detection. A particularly relevant one is the *SAS Security Intelligence* [528] software, which is designed to address fraud, compliance and security issues. All of the above can handle both supervised and unsupervised scenarios. Furthermore, for the supervised case, a significant amount of classification software is available both on an open source and commercial basis. An exhaustive list of the different kinds of open source and commercial software may be found in [520].

12.   Conclusions and Summary

This chapter provides an overview of the applications of outlier analysis. The applications of outlier analysis are distributed across a wide variety of domains. Nevertheless, many of these domains map to similar formulations for modeling purposes. The goal of this chapter was to provide the practitioner an understanding of the methods used in dif-

ferent domains. Outlier analysis brings numerous challenges with it in different application domains, because of the difficulty in distinguishing between noise and anomalies. Typically, the incorporation of human feedback, domain knowledge and explicit supervision can address many of these challenges. It was also discussed in this chapter, how many of the generic outlier analysis methods can be adapted to specific domains.

References

[1] N. Abe, B. Zadrozny, and J. Langford. Outlier Detection by Active Learning, *ACM KDD Conference*, 2006.

[2] E. Achtert, A. Hettab, H.-P. Kriegel, E. Schubert, and A. Zimek. Spatial Outlier Detection: Data, Algorithms, Visualizations. *SSTD Conference*, 2011.

[3] N. R. Adam, V. P. Janeja, and V. Atluri. Neighborhood based detection of anomalies in high dimensional spatio-temporal sensor datasets. *ACM SAC Conference*, 2004.

[4] C. C. Aggarwal, and P. S. Yu. Outlier Detection in High Dimensional Data, *ACM SIGMOD Conference*, 2001.

[5] C. C. Aggarwal, C. Procopiuc, J. Wolf, P. Yu, and J. Park. Fast Algorithms for Projected Clustering, *ACM SIGMOD Conference*, 1999.

[6] C. Aggarwal, J. Han, J. Wang, and P. Yu. A Framework for Projected Clustering of High Dimensional Data Streams. In *VLDB Conference*, 2004.

[7] C. C. Aggarwal, and P. S. Yu. Finding Generalized Projected Clusters in High Dimensional Spaces, *ACM SIGMOD Conference*, 2000.

[8] C. C. Aggarwal. Re-designing Distance Functions and Distance-based Applications for High Dimensional Data, *ACM SIGMOD Record*, 2001.

[9] C. C. Aggarwal. On Abnormality Detection in Spuriously Populated Data Streams, *SIAM Conference on Data Mining*, 2005.

[10] C. C. Aggarwal. Data Streams: Models and Algorithms, *Springer*, 2007.

[11] C. C. Aggarwal. Social Network Data Analytics, *Springer*, 2011.

[12] C. C. Aggarwal. On Effective Classification of Strings with Wavelets, *ACM KDD Conference*, 2002.

[13] C. C. Aggarwal, N. Ta, J. Wang, J. Feng, and M. J. Zaki. Xproj: A Framework for Projected Structural Clustering of XML Documents. *ACM KDD Conference*, 2007.

[14] C. C. Aggarwal, and P. S. Yu. On String Classification in Data Streams, *ACM KDD Conference*, 2007.

[15] C. C. Aggarwal, Y. Zhao, and P. S. Yu. Outlier Detection in Graph Streams, *ICDE Conference*, 2011.

[16] C. C. Aggarwal. A Framework for Diagnosing Changes in Evolving Data Streams, *ACM SIGMOD Conference*, 2003.

[17] C. C. Aggarwal, and P. S. Yu. Online Analysis of Community Evolution in Data Streams, *SDM Conference*, 2005.

[18] C. C. Aggarwal. On the Effects of Dimensionality Reduction on High Dimensional Similarity Search, *ACM PODS Conference*, 2001.

[19] C. C. Aggarwal. Managing and Mining Sensor Data, *Springer*, 2013.

[20] C. C. Aggarwal, and C. K. Reddy. Data Clustering: Algorithms and Applications, *CRC Press*, 2013.

[21] C. Aggarwal, and C. Zhai. Managing and Mining Text Data, *Springer*, 2012.

[22] C. C. Aggarwal, A. Hinneburg, and D. Keim. On the Surprising Behavior of Distance Metrics in High Dimensional Space, *ICDT Conference*, 2001.

[23] C. C. Aggarwal, and P. S. Yu. Outlier Detection with Uncertain Data, *SDM Conference*, 2008.

[24] C. C. Aggarwal, Y. Xie, and P. S. Yu. On Dynamic Data-Driven Selection of Sensor Streams, *ACM KDD Conference*, 2011.

[25] C. C. Aggarwal, J. Han. J. Wang, and P. Yu. A Framework for Clustering Evolving Data Streams, *VLDB Conference*, 2003.

[26] C. C. Aggarwal, and P. Yu. On Clustering Massive Text and Categorical Data Streams, *Knowledge and Information Systems*, 24(2), pp. 171–196, 2010.

[27] C. C. Aggarwal, and K. Subbian. Event Detection in Social Streams, *SDM Conference*, 2012.

[28] R. Agrawal, T. Imielinski, and A. Swami. Mining Association Rules Between Sets of Items in Large Databases. *ACM SIGMOD Conference*, 1993.

[29] R. Agrawal, and R. Srikant. Fast algorithms for finding Association Rules in Large Databases, *VLDB Conference*, 1994.

[30] M. Agyemang, K. Barker, and R. Alhajj. A Comprehensive Survey of Numeric and Symbolic Outlier Mining Techniques, *Intelligent Data Analysis*, 10(6). pp. 521–538, 2006.

[31] A. Ahmad and L. Dey. A Method to Compute Distance between two Categorical Values of Same Attribute in Unsupervised Learning for Categorical Data Set. *Pattern Recognition Letters*, 28(1), pp. 110–118, 2007.

[32] R. Ahuja, J. Orlin, and T. Magnanti. Network Flows: Theory, Algorithms and Applications, *Prentice Hall*, 1993.

[33] L. Akoglu, M. McGlohon, and C. Faloutsos. Oddball: Spotting Anomalies in Weighted Graphs, *PAKDD Conference*, 2010.

[34] L. Akoglu, H. Tong, J. Vreeken, and C. Faloutsos. Fast and Reliable Anomaly Detection in Categorical Data, *CIKM Conference*, 2012.

[35] E. Aleskerov, B. Freisleben, and B. Rao. CARDWATCH: A Neural Network based Database Mining System for Credit Card Fraud Detection. *IEEE Computational Intelligence for Financial Engineering*, pp. 220–226, 1997.

[36] T. Al-Khateeb, M. Masud, L. Khan, C. Aggarwal, J. Han, and B. Thuraisingham. Recurring and Novel Class Detection using Class-based Ensemble, *ICDM Conference*, 2012.

[37] J. Allan, R. Papka, and V. Lavrenko. On-line new event detection and tracking. *ACM SIGIR Conference*, 1998.

[38] J. Allan, V. Lavrenko, and H. Jin. First story detection in TDT is hard. *ACM CIKM Conference*, 2000.

[39] J. Allan, J. Carbonell, G. Doddington, J. Yamron, and Y. Yang. Topic Detecting and Tracking Pilot Study Final Report, *CMU Technical Report, Paper 341*, 1998.

[40] F. Alonso, J. Caraca-Valente, A. Gonzalez, and C. Montes. Combining Expert Knowledge and Data Mining in a Medical Diagnosis Domain. *Expert Systems with Applications*, 23(4), pp. 367–375, 2002.

[41] Y. Altun, I. Tsochantaridis, and T. Hofmann. Hidden Narkov Support Vector Machines. *ICML Conference*, 2003.

[42] M. R. Anderberg. Cluster Analysis for Applications, *Academic Press*, New York, 1973.

[43] D. Anderson, T. Lunt, H. Javitz, A. Tamaru, and A. Valdes. Detecting Unusual Program Nehavior using the Statistical Components of NIDES, *Techical Report, SRI–CSL–95–06, Computer Science Laboratory*, SRI International, 1995.

[44] D. Anderson, T. Frivold, A. Tamaru, and A. Valdes. Next-generation Intrusion Detection Expert System (nides), Software Users Manual, Beta-update Release. *Technical Report SRI–CSL–95–07, Computer Science Laboratory*, SRI International, 1994.

[45] S. Ando. Clustering Needles in a Haystack: An Information Theoretic Analysis of Minority and Outlier Detection. *ICDM Conference*, 2007.

[46] F. Angiulli, and C. Pizzuti. Fast Outlier Detection in High Dimensional Spaces. *European Conferece on Principles of Knowledge Discovery and Data Mining*, 2002.

[47] F. Angiulli, F. Fassetti, amd L. Palopoli. Finding Outlying Properties of Exceptional Objects, *ACM Transactions on Database Systems*, 34(1), 2009.

[48] F. Angiulli and F. Fassetti. Detecting Distance-based Outliers in Streams of Data, *ACM CIKM Conference*, 2007.

[49] A. Arning, R. Agrawal, and P. Raghavan. A Linear Method for Deviation Detection in Large Databases. *ACM KDD Conference*, 1996.

[50] I. Assent, P. Kranen, C. Beldauf, and T. Seidl. AnyOut: Anytime Outlier Detection in Streaming Data, *DASFAA Conference*, 2012.

[51] I. Assent, R. Krieger, E. Muller, and T. Seidl. Subspace Outlier Mining in Large Multimedia Databases, *Parallel Universes and Local Patterns*, 2007.

[52] A. Auer Jr. Correlation of Land Use and Cover with Meteorological Anomalies. *Journal of Applied Meteorology*, 17(5) pp. 636–643, 1978.

[53] L. Backstrom, D. Huttenlocher, J. Kleinberg, and X. Lan. Group formation in large social networks: membership, growth, and evolution. *KDD Conference*, 2006.

[54] D. Ballard. Generalizing the Hough Transform to Detect Arbitrary Shapes, *Pattern Recognition*, 11(2), pp. 111–122, 1981.

[55] D. Barbara, Y. Li, J. Couto, J.-L. Lin, and S. Jajodia. Bootstrapping a Data Mining Intrusion Detection System. *Symposium on Applied Computing*, 2003.

[56] D. Barbara, J. Couto, S. Jajodia, and N. Wu. ADAM: A Testbed for Exploring the Use of Data Mining in Intrusion Detection. *ACM SIGMOD Record*, 30(4), pp. 15–24, 2001.

[57] D. Barbara, J. Couto, S. Jajodia, and N. Wu. Detecting Movel Network Intrusions using Bayes Estimators. *SIAM Conference on Data Mining*, 2001.

[58] V. Barnett and T. Lewis. Outliers in Statistical Data, *Wiley*, 1994.

[59] R. Baragona and F. Battaglia. Outlier Detection in Multivariate Time Series by Independent Component Analysis. *Neural Computation*, 19(1), pp. 1962–1984, 2007.

[60] S. Bay, and M. Schwabacher. Mining distance-based outliers in near linear time with randomization and a simple pruning rule. *ACM KDD Conference*, 2003.

[61] S. Bay, K. Saito, N. Ueda, and P. Langley. A Framework for Discovering Anomalous Regimes in Multivariate Time-series Data with Local Models. *Technical report, Center for the Study of Language and Information*, Stanford University, 2004.

[62] R. Beckman, and R. Cook. Outliers, *Technometrics*, 25(2), pp. 119–149, 1983.

[63] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R*-tree: An Efficient and Robust Access method for Points and Rectangles. *ACM SIGMOD Conference*, 1990.

[64] M. Berlingerio, F. Bonchi, B. Bringmann, and A. Gionis. Mining Graph Evolution Rules. *ECML/PKDD Conference*, 2009.

[65] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is "nearest neighbor" meaningful? *International Conference on Database Theory*, 1999.

[66] K. Bhaduri, B. Matthews, and C. Giannella. Algorithms for Speeding up Distance-based Outlier Detection. *ACM KDD Conference*, 2011.

[67] E. Blanzieri and A. Bryl. A Survey of Learning-based Techniques of Email Spam Filtering. *Artificial Intelligence Review*, 29(1), pp. 63–92, 2008.

[68] M. Bosc, F. Heitz, J.-P. Armspach, I. Namer, D. Gounot, and L. Rumbach. Automatic Change Detection in Multimodal Serial MRI: application to multiple sclerosis lesion evolution, *NeuroImage*, 20(2), 2003, Pages 643–656

[69] Y. Bilberman. A Context Similarity Measure. *ECML Conference*, 1994.

[70] P. Billingsley. Probability and Measure, Second Edition, *Wiley*, 1986.

[71] D. Birant, and A. Kut. Detecting Spatio-temporal Outliers in Large Databases, *Journal of Computing and Information Technology*, 14(4), pp. 291–297, 2006.

[72] D. Blei, and J. Lafferty. Dynamic topic models. *ICML Conference*, 2006.

[73] D. Blei, A. Ng, and M. Jordan. Latent Dirichlet allocation, *Journal of Machine Learning Research*, 3: pp. 993–1022, 2003.

[74] C. Bohm, K. Haegler, N. Muller, and C. Plant. Coco: Coding Cost for Parameter Free Outlier Detection, *ACM KDD Conference*, 2009.

[75] S. Boriah, V. Chandola, and V. Kumar. Similarity Measures for Categorical Data: A Comparative Evaluation, *SIAM Conference on Data Mining*, 2008.

[76] J. Branch, B. Szymanski, C. Giannella, R. Wolff, and H. Kargupta. In-Network Outlier Detection in Wireless Sensor Networks. *ICDCS Conference*, 2006.

[77] T. Brants, F. Chen, and A. Farahat. A System for New Event Detection. *ACM SIGIR Conference*, 2003.

[78] M. Breunig, H.-P. Kriegel, R. Ng, and J. Sander. LOF: Identifying Density-based Local Outliers, *ACM SIGMOD Conference*, 2000.

[79] M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. OPTICS-OF: identifying local outliers. *PKDD Conference*, 1999.

[80] L. Brieman. Bagging Predictors, *Machine Learning*, 24: pp. 123–140, 1996.

[81] M. R. Brito, E. L. Chavez, A. J. Quiroz, and J. E. Yukich. Connectivity of the Mutual *k*-Nearest Neighbor Graph in Clustering and Outlier Detection. *Statistics and Probability Letters*, 35(1), pp. 33–42, 1997.

[82] R. G. Brown, and P. Hwang. Introduction to Random Signals and Applied Kalman Filtering, *John Wiley and Sons*, 1997.

[83] Y. Bu, L. Chen, A. W.-C. Fu, and D. Liu. Efficient Anomaly Monitoring over Moving Object Trajectory Streams. *ACM KDD Conference*, 2009.

[84] S. Budalakoti, A. Srivastava, R. Akella, and E. Turkov. Anomaly Detection in Large Sets of High-dimensional Symbol Sequences, *NASA Ames Research Center*, Technical Report NASA TM-2006-214553, 2006.

[85] S. Budalakoti, A. Srivastava, and M. Otey. Anomaly Detection and Diagnosis Algorithms for Discrete Symbol Sequences with Applications to Airline Safety, *IEEE International Conference on Systems, Man, and Cybernetics*, 37(6), 2007.

[86] S Burdakis, A Deligiannakis. Detecting Outliers in Sensor Networks using the Geometric Approach, *ICDE Conference*, 2012.

[87] T. Burnaby. On a Method for Character Weighting a Similarity Coefficient, Employing the Concept of Information. *Mathematical Geology*, 2(1), 25–38, 1970.

[88] S. Byers, and A. Raftery. Nearest-Neighbor Clutter Removal for Estimating Features in Spatial Point Processes, *JASIS*, 93, pp. 577–584, June 1998.

[89] I. Cadez, D. Heckerman, C. Meek, P. Smyth, and S. White. Visualization of Navigation Patterns on a Web Site using Model-based Clustering, *ACM SIGMOD Conference*, 2000.

[90] P. H. Calamai. Projected gradient methods for linearly constrained problems. *Mathematical Programming*, 39: pp, 93–116, 1987.

[91] C. Campbell, and K. P. Bennett. A Linear-Programming Approach to Novel Class Detection, *NIPS Conference*, 2000.

[92] M. J. Canty. Image Analysis, Classification and Change Detection in Remote Sensing: with Algorithms for ENVI/IDL, *CRC Press*, 2006.

[93] C. Caroni. Outlier Detection by Robust Principal Component Analysis. *Communications in Statistics – Simulation and Computation*, 29: pp. 129–151, 2000.

[94] L. E. Carr and R. L. Elsberry. Monsoonal interactions leading to sudden tropical cyclone track changes, *Monthly Weather Review*, 123(2), pp. 265–290, Feb. 1995.

[95] K. Chakrabarti, S. Mehrotra. Local Dimensionality Reduction: A New Approach to Indexing High Dimensional Spaces. *VLDB Conference Proceedings*, 2000.

[96] S. Chakrabarti, S. Sarawagi, and B. Dom. Mining Surprising Patterns using Temporal Description Length. *VLDB Conference*, 1998.

[97] V. Chandola, V. Mithal, and V. Kumar. A Comparative Evaluation of Anomaly Detection Techniques for Sequence Data, *International Conference on Data Mining*, 2008.

[98] A. Chaudhary, A. S. Szalay, and A. W. Moore. Very Fast Outlier Detection in Large Multidimensional Data Sets. *DMKD Workshop*, 2002.

[99] D. Chakrabarti, and C. Faloutsos. Evolutionary Clustering, *ACM KDD Conference*, 2006.

[100] D. Chakrabarti. AutoPart: Parameter-Free Graph Partitioning and Outlier Detection. *PKDD Conference*, 2004.

[101] C.-H. Chan and G. Pang. Fabric Defect Detection by Fourier Analysis, *IEEE Transactions on Industry Applications*, 36(5), 2000.

[102] N. V. Chawla, N. Japkowicz, and A. Kotcz. Editorial: Special Issue on Learning from Imbalanced Data Sets, *ACM SIGKDD Explorations Newsletter*, 6(1), 1–6, 2004.

[103] N. V. Chawla, K. W. Bower, L. O. Hall, and W. P. Kegelmeyer. SMOTE: synthetic minority over-sampling technique, *Journal of Artificial Intelligence Research (JAIR)*, 16, pp. 321–356, 2002.

[104] N. Chawla, A. Lazarevic, L. Hall, and K. Bowyer. SMOTEBoost: Improving prediction of the minority class in boosting, *PKDD*, pp. 107–119, 2003.

[105] N. V. Chawla, D. A. Cieslak, L. O. Hall, and A. Joshi. Automatically Cuntering Imbalance and its Empirical Relationship to Cost. *Data Mining and Knowledge Discovery*, 17(2), pp. 225–252, 2008.

[106] P. K. Chan and S. J. Stolfo. Toward Scalable Learning with Non-uniform Class and Cost Distributions: A Case Study in Credit Card Fraud Detection. *KDD Conference*, pp. 164–168, 1998.

[107] V. Chandola, A. Banerjee, and V. Kumar. Anomaly Detection: A Survey. *ACM Computing Surveys*, 41(3), 2009.

[108] V. Chandola, A. Banerjee, and V. Kumar. Anomaly Detection for Discrete Sequences: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, 24(5): pp. 823–839, 2012.

[109] I. Chang, G. C. Tiao, and C. Chen. Estimation of Time Series Parameters in the Presence of Outliers. *Technometrics*, 30(2), pp. 193–204, 1988.

[110] C. Chen and L.-M. Liu. Joint Estimation of Model Parameters and Outlier Effects in Time Series. *Journal of the American Statistical Association*, 88(421), pp. 284–297, March 1993.

[111] Y. Chen, and L. Tu. Density-based Clustering for Real Time Stream Sata, *ACM KDD Conference*, 2007.

[112] D. Chen, C.-T. Lu, Y. Chen, and D. Kou. On Detecting Spatial Outliers, *Geoinformatica*, 12: pp. 455–475, 2008.

[113] T. Cheng and Z. Li. A Hybrid Approach to Detect Spatialtemporal Outliers. *International Conference on Geoinformatics*, 2004.

[114] T. Cheng and Z. Li. A Multiscale Approach for Spatio-temporal Outlier Detection, *Transactions in GIS*, 10(2), pp. 253–263, March 2006.

[115] H. Cheng, P.-N. Tan, C. Potter, and S. Klooster. Detection and Characterization of Anomalies in Multivariate Time Series, *SIAM Conference on Data Mining*, 2009.

[116] Y. Chi, X. Song, D. Zhou, K. Hino, and B. Tseng. Evolutionary Spectral Clustering by Incorporating Temporal Smoothness. *ACM KDD Conference*, 2007.

[117] A. Chiu, and A. Fu. Enhancements on Local Outlier Detection. *Database Engineering and Applications Symposium*, 2003.

[118] M. Chow, R. Sharpe, and J. Hung. On the Application and Design of Artificial Neural Networks for Motor Fault Detection. *IEEE Transactions on Industrial Electronics*, 40(2), 1993.

[119] C. Chow, sand D. Yeung. Parzen-Window Network Intrusion Detectors. *International Conference on Pattern Recognition*, 4, 2002.

[120] W. Cohen. Fast Effective Rule Induction. *ICML Conference*, 1995.

[121] D. Cohn, R. Atlas, and N. Ladner. Improving Generalization with Active Learning, *Machine Learning*, 15, pp. 201–221.

[122] R. Cooley, B. Mobashar, and J. Srivastava. Data Preparation for Mining World Wide Web Browsing Patterns, *Knowledge and Information Systems*, 1, pp, 5–32, 1999.

[123] D. Cook, and L. Holder. Graph-Based Data Mining. *IEEE Intelligent Systems*, 15(2), pp. 32–41, 2000.

[124] G. Cormack. Email Spam Filtering: A Systematic Review, *Foundations and Trends in Information Retrieval*, 1(4), pp. 335–455, 2007.

[125] C. Darwin. The Origin of the Species by Natural Selection, 1859. Manuscript now publicly hosted at: http://www.literature.org/authors/darwin-charles/the-origin-of-species/

[126] G. Das and H. Mannila. Context-based Similarity Measures for Categorical Databases. *PKDD Conference*, 2000.

[127] K. Das, J. Schneider, and D. Neill. Anomaly Pattern Detection in Categorical Data Sets, *ACM KDD Conference*, 2008.

[128] S. Das, B. Matthews, A. Srivastava, and N. Oza. Multiple kernel learning for heterogeneous anomaly detection: algorithm and aviation safety case study. *ACM KDD Conference*, 2010.

[129] D. Dasgupta and S. Forrest. Novelty Detection in Time Series using Ideas from Immunology, *International Conference on Intelligent Systems*, 1996.

[130] D. Dasgupta, and F. Nino. A comparison of negative and positive selection algorithms in novel pattern detection. *IEEE Conference on Systems, Man, and Cybernetics*, 1, pp. 125–130, 2000.

[131] T. Dasu, S. Krishnan, S. Venkatasubramanian, and K. Yi. An Information-Theoretic Approach to Detecting Change in Multidimensional Data Streams, *Symposium on the Interface of Computer Science, Statistics, and Applications*, 2006.

[132] N. Delannay, C. Archambeau, and M. Verleysen. Improving the Robustness to Outliers of Mixtures of Probabilistic PCAs. *PAKDD Conference*, 2008.

[133] S. T. Deerwester, S. T. Dumais, G. Furnas, and R. Harshman. Indexing by Latent Semantic Analysis, *JASIS*, 1990.

[134] K. A. De Jong. Analysis of the behaviour of a class of Genetic Adaptive Systems. *Ph.D. Dissertation, University of Michigan*, Ann Arbor, MI, 1975.

[135] A. P. Dempster, N. M. Laird and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm, *Journal of the Royal Statistical Society*, B, vol. 39(1), pp. 1–38, 1977.

[136] D. Denning. An Intrusion Detection Model. *IEEE Transactions of Software Engineering*, 13(2), pp. 222–232.

[137] R. Derrig. Insurance Fraud. *Journal of Risk and Insurance*, 69(3), pp. 271–287, 2002.

[138] M. Desforges, P. Jacob, and J. Cooper. Applications of Probability Density Estimation to the Detection of Abnormal Conditions in Engineering. *Proceedings of Institute of Mechanical Engineers*, Vol. 212, pp. 687–703, 1998.

[139] M. Deshpande and G. Karypis. Evaluation of Techniques for Classifying Biological Sequences. *PAKDD Conference*, 2002.

[140] M. Deshpande and G. Karypis. Selective Markov Models for Predicting Web Page Accesses, *ACM Transactions on Internet Technology*, 4(2), pp. 163–184, 2004.

[141] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh. Querying and Mining of Time Series Data: Experimental Comparison of Representations and Distance Neasures. *PVLDB*, 1(2), pp. 1542–1552, 2008.

[142] S. Donoho. Early Detection of Insider Trading in Option Markets. *ACM KDD Conference*, 2004.

[143] C. Drummond and R. Holte. C4.5, Class Imbalance, and Cost Sensitivity: Why Undersampling beats Oversampling. *ICML Workshop on Learning from Imbalanced Data Sets*, 2003.

[144] C. Drummond and R. Holte. Explicitly Representing Expected Cost: An Alternative to ROC representation. *ACM KDD Conference*, pp. 198–207, 2001.

[145] P. Domingos. MetaCost: A General Framework for Making Classifiers Cost-Sensitive, *ACM KDD Conference*, 1999.

[146] R. Duda, P. Hart, and D. Stork, Pattern Classification, *Wiley*, 2001.

[147] H. Dutta, C. Giannella, K. Borne, and H. Kargupta. Distributed top-k Outlier Detection in Astronomy Catalogs using the Demac System. *SDM Conference*, 2007.

[148] W. Eberle and L. B. Holder. Mining for Structural Anomalies in Graph-based Data. *DMIN*, 2007.

[149] F. Y. Edgeworth. On Discordant Observations. *Philosophical Magazine*, 23(5), pp. 364–375, 1887.

[150] M. Elahi, K. Li, W. Nisar, X. Lv, and H. Wang. Efficient Clustering-Based Outlier Detection Algorithm for Dynamic Data Stream. *FSKD Conference*, 2008.

[151] C. Elkan. The Foundations of Cost-Sensitive Learning, *IJCAI*, 2001.

[152] C. Elkan, and K. Noto. Learning Classifiers from only Positive and Unlabeled Data, *ACM KDD Conference*, 2008.

[153] D. Endler. Intrusion detection: Applying Machine Learning to Solaris Audit Data, *Annual Computer Security Applications Conference*, 1998.

[154] E. Eskin. Anomaly Detection over Noisy Data using Learned Probability Distributions, *ICML Conference*, 2000.

[155] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo. A Geometric Framework for Unsupervised Anomaly Detection, In *Applications of Data Mining in Computer Security*. Kluwer, 2002.

[156] E. Eskin, W. Lee, and S. Stolfo, Modeling System Call for Intrusion Detection using Dynamic Window Sizes, *DISCEX*, 2001.

[157] H. Fan, O. Zaiane, A. Foss. and J. Wu. A Nonparametric Outlier Detection for Efficiently Discovering top-n Outliers from Engineering Data. *PAKDD Conference*, 2006.

[158] W. Fan, S. Stolfo, J. Zhang, and P. Chan. AdaCost: Misclassification Cost Sensitive Boosting, *ICML Conference*, 1999.

[159] T. Fawcett. ROC Graphs: Notes and Practical Considerations for Researchers, *Technical Report HPL-2003-4*, Palo Alto, CA: HP Laboratories, 2003.

[160] T. Fawcett and F. Provost. Activity Monitoring: Noticing Interesting Changes in Behavior. *ACM KDD Conference*, 1999.

[161] D. Fetterly, M. Manasse, and M. Najork. Spam, Damn Spam, and Statistics: using Statistical Analysis to Locate Spam Web Pages, *WebDB*, 2004.

[162] A. Lung-Yut-Fong, C. Levy-Leduc, and O. Cappe. Distributed Detection/localization of Change-points in High-dimensional Network Traffic Data. *Corr*, abs/0909.5524, 2009.

[163] S. Forrest, C. Warrender, and B. Pearlmutter. Detecting Intrusions using System Calls: Alternate Data Models, *IEEE ISRSP*, 1999.

[164] S. Forrest, S. Hofmeyr, A. Somayaji, and T. A. Longstaff. A Sense of Self for Unix Processes, *ISRSP*, 1996.

[165] S. Forrest, P. D'Haeseleer, and P. Helman. An Immunological Approach to Change Detection: Algorithms, Analysis and Implications. *IEEE Symposium on Security and Privacy*, 1996.

[166] S. Forrest, F. Esponda, and P. Helman. A Formal Framework for Positive and Negative Detection Schemes. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, pp. 357–373, 2004.

[167] S. Forrest, A. Perelson, L. Allen, and R. Cherukuri. Self-Nonself Discrimination in a Computer. *IEEE Symposium on Security and Privacy*, 1994.

[168] A. Fox. Outliers in Time Series. *Journal of the Royal Statistical Society. Series B (Methodological)*, 34(3), pp. 350–363, 1972.

[169] A. Frank, and A. Asuncion. UCI Machine Learning Repository, Irvine, CA: University of California, School of Information and Computer Science, 2010. http://archive.ics.uci.edu/ml

[170] C. Franke and M. Gertz. ORDEN: Outlier Region Detection and Exploration in Sensor Networks. *ACM SIGMOD Conference*, 2009.

[171] A. Fu, O. Leung, E. Keogh, and J. Lin. Finding Time Series Discords based on Haar Transform. *Advanced Data Mining and Applications*, 2006.

[172] R. Fujumaki, T. Yairi, and K. Machida. An Approach to Spacecraft Anomaly Detection Problem using Kernel Feature Space. *ACM KDD Conference*, 2005.

[173] R. Fujamaki. Anomaly Detection Support Vector Machine and Its Application to Fault Diagnosis, *ICDM Conference*, 2008.

[174] P. Galeano, D. Pea, and R. S. Tsay. Outlier detection in Multivariate Time Series via Projection Pursuit.*Statistics and Econometrics Working Papers WS044221*, Universidad Carlos III, 2004.

[175] P. Gambaryan. A Mathematical Model of Taxonomy. *Izvest. Akad. Nauk Armen*, SSR, 17(12), pp. 47–53, 1964.

[176] V. Ganti, J. Gehrke, and R. Ramakrishnan. CACTUS: Clustering Categorical Data using Summaries. *ACM KDD Conference*, 1999.

[177] B. Gao, H.-Y. Ma, and Y.-H. Yang, HMMs (Hidden Markov Models) based on Anomaly Intrusion Detection Method, *International Conference on Machine Learning and Cybernetics*, 2002.

[178] H. Gao, X. Wang, J. Tang and H. Liu. Network Denoising in Social Media, *Technical Report, Arizona State University*, 2011.

[179] J. Gao and P.-N. Tan. Converting Outlier Scores from Outlier Detection Algorithms into Probability Estimates, *ICDM Conference*, 2006.

[180] J. Gao, F. Liang, W. Fan, C. Wang, Y. Sun, and J. Han. On Community Outliers and their Efficient Detection in Information Networks. *ACM KDD Conference*, pp. 813–822, 2010.

[181] Y. Ge, H. Xiong, Z.-H. Zhou, H. Ozdemir, J. Yu, and K. Lee. Top-Eye: Top-*k* Evolving Trajectory Outlier Detection. *CIKM Conference*, 2010.

[182] A. Ghosh, J. Wanken, and F. Charron. Detecting Anomalous and Unknown Intrusions against Programs, *Annual Computer Security Applications Conference*, 1998.

[183] A. Ghosh, A. Schwartzbard, and M. Schatz. Learning Program Behavior Profiles for Intrusion Detection, *USENIX Workshop on Intrusion Detection and Network Monitoring*, pp. 51–62, 1999.

[184] S. Ghosh and D. Reilly. Credit Card Fraud Detection with a Neural Network, *International Conference on System Sciences: Information Systems: Decision Support and Knowledge-Based Systems*, 3, pp. 621–630, 1994.

[185] A. Ghoting, S. Parthasarathy, and M. Otey. Fast Mining of Distance-based Outliers in High Dimensional Spaces. *SIAM Conference on Data Mining*, 2006.

[186] D. Gibson, J. Kleinberg, and P. Raghavan. Clustering categorical data: an approach based on dynamical systems. *The VLDB Journal*, 8(3), pp. 222–236, 2000.

[187] D. W. Goodall. A new similarity index based on probability. *Biometrics*, 22(4), pp. 882–907, 1966.

[188] F. Grubbs. Procedures for Detecting Outlying Observations in Samples, *Technometrics*, 11(1), pp. 1–21, 1969.

[189] S. Guha, R. Rastogi, and K. Shim. ROCK: A robust clustering algorithm for categorical attributes. *Information Systems*, 25(5), pp. 345–366, 2000.

[190] D. Gunopulos and G. Das. Time-series Similarity Measures, and Time Series Indexing, *ACM SIGMOD Conference*, 2001.

[191] S. Gunter, N. N. Schraudolph, and S. V. N. Vishwanathan. Fast Iterative Kernel Principal Component Analysis. *Journal of Machine Learning Research*, 8, pp 1893–1918, 2007.

[192] M. Gupta, C. Aggarwal, J. Han, and Y. Sun. Evolutionary Clustering and Analysis of Bibliographic Networks, *ASONAM Conference*, 2011.

[193] M. Gupta, C. Aggarwal, and J. Han. Finding Top-k Shortest Path Distance Changes in an Evolutionary Network, *SSDBM Conference*, 2011.

[194] M. Gupta, J. Gao, Y. Sun, and J. Han. Community Trend Outlier Detection Using Soft Temporal Pattern Mining. *ECML/PKDD Conference*, 2012.

[195] M. Gupta, J. Gao, Y. Sun, and J. Han. Integrating Community Matching and Outlier Detection for Mining Evolutionary Community Outliers. *KDD Conference*, 2012.

[196] D. Gusfield. Algorithms for Strings, Trees and Sequences, *Cambridge University Press*, 1997.

[197] D. Guthrie, L. Guthrie, and Y. Wilks. An Unsupervised Approach for the Detection of Outliers in Corpora, *LREC*, 2008.

[198] S. Guttormsson, R. Marks, M. El-Sharkawi, and I. Kerszenbaum. Elliptical Novelty Grouping for Online Short-turn Detection of Excited Running Rotors. *IEEE Transactions on Energy Conversion*, 14(1), pp. 16–22, 1999.

[199] R. Gwadera, M. Atallah, and W. Szpankowski. Markov Models for Identification of Significant Episodes, *SDM Conference*, 2005.

[200] R. Gwadera, M. Atallah, and W. Szpankowskii. Detection of Significant Sets of Episodes in Event Sequences, *IEEE ICDM Conference*, 2004.

[201] R. Gwadera, M. Atallah, and W. Szpankowski. Reliable Detection of Episodes in Event Sequences, *Knowledge and Information Systems*, 7(4), pp. 415–437, 2005.

[202] F. Hampel. A General Qualitative Definition of Robustness, *Annals of Mathematics and Statistics*, 43, pp. 1887–1896, 1971.

[203] J. Haslett, R. Brandley, P. Craig, A. Unwin, and G. Wills. Dynamic Graphics for Exploring Spatial Data With Application to Locating Global and Local Anomalies. *The American Statistician*, 45: pp. 234–242, 1991.

[204] V. Hautamaki, I. Karkkainen, and P. Franti. Outlier Detection using *k*-nearest neighbor graph. *International Conference on Pattern Recognition*, 2004.

[205] D. Hawkins. Identification of Outliers, *Chapman and Hall*, 1980.

[206] G. G. Hazel. Multivariate Gaussian MRF for Multispectral Scene Segmentation and Anomaly Detection, *GeoRS*, 38(3), pp. 1199–1211, 2000.

[207] J. He, and J. Carbonell. Nearest-Neighbor-Based Active Learning for Rare Category Detection. *CMU Computer Science Department*,

Paper 281, 2007.
http://repository.cmu.edu/compsci/281

[208] Z. He, S. Deng, and X. Xu. Outlier Detection Integrating Semantic Knowledge. *Web Age Information Management (WAIM)*, 2002.

[209] Z. He, X. Xu, J. Huang, and S. Deng. FP-Outlier: Frequent Pattern-based Outlier Detection, *COMSIS*, 2(1), 2005.

[210] Z. He, X. Xu, and S. Deng. Discovering Cluster-based Local Outliers, *Pattern Recognition Letters*, Vol 24(9–10), pp. 1641–1650, 2003.

[211] Z. He, X. Xu, and S. Deng. An Optimization Model for Outlier Detection in Categorical Data. *International Conference on Intelligent Computing*, 2005.

[212] Z. He, S. Deng, X. Xu, and J. Huang. A Fast Greedy Algorithm for Outlier Mining. *PAKDD Conference*, 2006.

[213] M. Henrion, D. Hand, A. Gandy, and D. Mortlock. CASOS: A Subspace Method for Anomaly Detection in High Dimensional Astronomical Databases. *Statistical Analysis and Data Mining*, 2012. Online first: http://onlinelibrary.wiley.com/doi/10.1002/sam.11167

[214] S. Hido, Y. Tsuboi, H. Kashima, M. Sugiyama, and T. Kanamori. Statistical Outlier Detection using Direct Density Ratio Estimation. *Knowledge and information Systems*, 26(2), pp. 309–336, 2011.

[215] A. Hinneburg, C. Aggarwal, and D. Keim. What is the nearest neighbor in high-dimensional spaces?, *VLDB Conference*, 2000.

[216] H. O. Hirschfeld. A connection between correlation and contingency, *Proc. Cambridge Philosophical Society*, 31, pp. 520–524, 1935.

[217] S.-S. Ho. A Martingale Framework for Concept Change Detection in Time-Varying Data Streams, *ICML Conference*, 2005.

[218] V. Hodge and J. Austin. A Survey of Outlier Detection Methodologies, *Artifical Intelligence Review*, 22(2), pp. 85–126, 2004.

[219] J. Hodges. Efficiency in normal samples and tolerence of extreme values for some estimates of location, *Fifth Berkeley Symposium on Mathematics, Statistics and Probability*, 1, pp. 163–168, 1967.

[220] H. Hoffmann. Kernel PCA for Novelty Detection, *Pattern Recognition*, 40(3), pp. 863–874, 2007.

[221] T. Hofmann. Probabilistic Latent Semantic Indexing. *ACM SIGIR Conference*, 1999.

[222] S. Hofmeyr, S. Forrest, and A. Somayaji. Intrusion Detection using Sequences of System Calls, *Journal of Computer Security*, 6(3), pp. 151–180, 1998.

[223] J. H. Holland. Adaptation in Natural and Artificial Systems. *University of Michigan Press*, Ann Arbor, MI, 1975.

[224] G. Hollier, and J. Austin. Novelty Detection for Strain-gauge Degradation using Maximally Correlated Components. *European Symposium on Artificial Neural Networks*, 2002.

[225] J. Hollmen, and V. Tresp. Call-based Fraud Detection in Mobile Communication Networks using a Hierarchical Regime-switching Model. *NIPS Conference*, pp. 889–895, 1998.

[226] P. Horn, L. Feng, Y. Li, and A. J. Pesce. Effect of Outliers and Nonhealthy Individuals on Reference Interval Estimation. *Clinical Chemistry*, 47(12), pp. 2137–2145, 2001.

[227] L. Huang, M. I. Jordan, A. Joseph, M. Garofalakis, and N. Taft. In-network PCA and anomaly detection, *NIPS Conference*, 2006.

[228] J. W. Hunt and T. G. Szymanski. A Fast Algorithm for Computing Longest Common Subsequences, *Communications of the ACM*, 20(5), pp. 350–353, 1977.

[229] T. Ide, and H. Kashima. Eigenspace-based Anomaly Detection in Computer Systems. *ACM KDD Conference*, 2004.

[230] P. Indyk, R. Motwani, P. Raghavan, and S. Vempala. Locality-preserving Hashing in Multidimensional Spaces. *ACM STOC Conference*, 1997.

[231] D. Jackson, and Y. Chen. Robust Principal Component Analysis and Outlier Detection with Ecological Data, *Environmentrics*, 15, pp. 129–139, 2004.

[232] A. Jain, and R. Dubes. Algorithms for Clustering Data, *Prentice Hall*, 1988.

[233] H. Jagadish, N. Koudas, and S. Muthukrishnan. Mining Deviants in a Time-Series Database, *VLDB Conference*, 1999.

[234] V. Janeja and V. Atluri. Random Walks to Identify Anomalous Free-form Spatial Scan Windows. *IEEE Transactions on Knowledge and Data Engineering*, 20(10), 2008.

[235] P. Janeja and V. Atluri. Spatial Outlier Detection in Heterogeneous Neighborhoods. *Intelligent Data Analysis*, 13(1), 2008.

[236] H. Javitz, and A. Valdez. The SRI IDES Statistical Anomaly Detector. *IEEE Symposium on Security and Privacy*, 1991.

[237] Y. Jeong, M. Jeong, and O. Omitaomu, Weighted Dynamic Time Warping for Time Series Classification, *Pattern Recognition*, 44, pp. 2231–2240, 2010.

[238] B. Jiang, and J. Pei. Outlier Detection on Uncertain Data: Objects, Instances, and Inferences, *ICDE Conference*, 2011.

[239] M. F. Jiang, S. S. Tseng, and C. M. Su. Two-phase Clustering Process for Outliers Detection. *Pattern Recognition Letters*, 22, 6–7, pp. 691–700, 2001.

[240] R. Jiang, H. Fei, and J. Huan. Anomaly Localization for Network Data Streams with Graph Joint Sparse PCA. *ACM KDD Conference*, 2011.

[241] W. Jin, A. Tung, and J. Han. Mining Top-n Local Outliers in Large Databases. *ACM KDD Conference*, 2001.

[242] W. Jin, A. Tung, J. Han, and W. Wang. Ranking outliers using symmetric neighborhood relationship. *PAKDD Conference*, 2006.

[243] T. Johnson, I. Kwok, and R. Ng. Fast Computation of 2-dimensional Depth Contours. *ACM KDD Conference*, 1998.

[244] I. Jolliffe. Principal Component Analysis, *Springer*, 2002.

[245] M. Joshi, R. Agarwal, and V. Kumar. Mining Needles in a Haystack: Classifying Rare Classes via Two-Phase Rule Induction, *ACM SIGMOD Conference*, 2001.

[246] M. Joshi, V. Kumar, and R. Agarwal. Evaluating Boosting Algorithms to Classify Rare Classes: Comparison and Improvements. *ICDM Conference*, pp. 257–264, 2001.

[247] M. Joshi, and R. Agarwal. PNRule: A Framework for Learning Classifier Models in Data Mining (A Case Study in Network Intrusion Detection), *SDM Conference*, 2001.

[248] M. Joshi, R. Agarwal, and V. Kumar. Predicting Rare Classes: Can Boosting Make Any Weak Learner Strong? *ACM KDD Conference*, 2002.

[249] M. Joshi. On Evaluating Performance of Classifiers on Rare Classes, *ICDM Conference*, 2003.

[250] P. Juszczak and R. P. W. Duin. Uncertainty Sampling Methods for One-class Classifiers. *ICML Workshop on Learning from Imbalanced Data Sets*, 2003.

[251] J. Kang, S. Shekhar, C. Wennen, and P. Novak. Discovering Flow Anomalies: A SWEET Approach. *ICDM Conference*, 2008.

[252] G. Karakoulas and J. Shawe-Taylor. Optimising Classifiers for Imbalanced Training Sets, *NIPS*, 1998.

[253] D. R. Karger. Random sampling in cut, flow, and network design problems, *STOC*, pp. 648–657, 1994.

[254] S. Kasiviswanathan, P. Melville, and A. Banerjee. Emerging Topic Detection using Dictionary Learning, *CIKM Conference*, 2011.

[255] L. Kaufman, and P. Rousseeauw. Finding Groups in Data: An Introduction to Cluster Analysis, *Wiley-Interscience*, 1990.

[256] F. Keller, E. Muller, K. Bohm. HiCS: High-Contrast Subspaces for Density-based Outlier Ranking, *IEEE ICDE Conference*, 2012.

[257] E. Keogh, S. Lonardi, and B. Y.-C. Chiu. Finding Surprising Patterns in a Time Series Database in Linear Time and Space. *ACM KDD Conference*, 2002.

[258] E. Keogh, J. Lin, and A. Fu. HOT SAX: Finding the Most Unusual Time Series Subsequence: Algorithms and Applications, *ICDM Conference*, 2005.

[259] E. Keogh, S. Lonardi, and C. Ratanamahatana. Towards Parameter-Free Data Mining. *ACM KDD Conference*, 2004.

[260] D. Kifer, S. Ben-David, and J. Gehrke. Detecting Change in Data Streams, *VLDB Conference*, 2004.

[261] E. Knorr, and R. Ng. Algorithms for Mining Distance-based Outliers in Large Datasets. *VLDB Conference*, 1998.

[262] E. Knorr, and R. Ng. Finding Intensional Knowledge of Distance-Based Outliers. *VLDB Conference*, 1999.

[263] E. M. Knorr, R. T. Ng, and V. Tucakov. Distance-based Outliers: Algorithms and Applications, *VLDB Journal*, 8(3), pp. 237–253, February 2000.

[264] J. Koh, M.-L. Lee, W. Hsu, and W. Ang. Correlation-based Attribute Outlier Detection in XML. *ICDE Conference*, 2008.

[265] G. Kollios, D. Gunopulos, N. Koudas, and S. Berchtold. Efficient Biased Sampling for Approximate Clustering and Outlier Detection in Large Data Sets, *IEEE Transactions on Knowledge and Data Engineering*, 15(5), pp. 1170–1187, 2003.

[266] M. Kontaki, A. Gounaris, A. Papadopoulos, K. Tsichlas, and Y. Manolopoulos. Continuous Monitoring of Distance-based Outliers over Data Streams, *ICDE Conference*, 2011.

[267] K. Kontonasios and T. Bie. An Information-Theoretic Approach to Finding Noisy Tiles in Binary Databases, *SIAM Conference on Data Mining*, 2003.

[268] Y. Kou, C. T. Lu, and D. Chen. Spatial Weighted Outlier Detection, *SDM Conference*, 2006.

[269] H.-P. Kriegel, M. Schubert, and A. Zimek. Angle-based Outlier Detection in High-Dimensional Data, *ACM KDD Conference*, 2008.

[270] H.-P. Kriegel, P. Kroger, and A. Zimek. Outlier Detection Techniques, *Conference Tutorial at SIAM Data Mining Conference*, 2010. Tutorial Slides at: http://www.siam.org/meetings/sdm10/tutorial3.pdf

[271] H.-P. Kriegel, P. Kroger, E. Schubert, and A. Zimek. Interpreting and Unifying Outlier Scores. *SDM Conference*, pp. 13–24, 2011.

[272] H.-P. Kriegel, P. Kroger, E. Schubert, and A. Zimek. A General Framework for Increasing the Robustness of PCA-Based Correlation Clustering Algorithms. *SSDBM Conference*, 2008.

[273] H.-P. Kriegel, P. Kroger, E. Schubert, and A. Zimek. Outlier Detection in Axis-Parallel Subspaces of High Dimensional Data. *PAKDD Conference*, 2009.

[274] H.-P. Kriegel, P. Kroger, E. Schubert, and A. Zimek. Outlier Detection in Arbitrarily Oriented Subspaces, *ICDM Conference*, 2012.

[275] C. Kruegel, and G. Vigna. Anomaly-detection of Web-based Attacks, *CCS*, 2005.

[276] C. Kruegel, D. Mutz, W. Robertson, and F. Valeur. Bayesian Event Classification for Intrusion Detection. *Computer Security Applications Conference*, 2003.

[277] C. Kruegel, T. Toth, and E. Kirda. Service Specific Anomaly Detection for Network Intrusion Detection. *ACM symposium on Applied computing*, 2002.

[278] M. Kubat and S. Matwin. Addressing the Curse of Imbalanced Training Sets: One Sided Selection. *ICML Conference*, 1997.

[279] L. Kuncheva. Change Detection in Streaming Multivariate Data using Likelihood Detectors, *IEEE Transactions on Knowledge and Data Engineering*, Preprint, PP(99), 2011.

[280] A. Lakhina, M. Crovella, and C. Diot. Mining Anomalies using Traffic Feature Distributions, *ACM SIGCOMM Conference*, pp. 217–228, 2005.

[281] A. Lakhina, M. Crovella, and C. Diot. Diagnosing network-wide traffic anomalies. *ACM SIGCOMM Conference*, pp. 219–230, 2004.

[282] G. Lanckriet, L. Ghaoui, and M. Jordan. Robust Novelty Detection with Single Class MPM, *NIPS*, 2002.

[283] J. Lafferty, A. McCallum, and F. Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data, *ICML Conference*, 2001.

[284] T. Lane and C. Brodley. Temporal Sequence Learning and Data Reduction for Anomaly Detection, *ACM Transactions on Information and Security*, 2(3), pp. 295–331, 1999.

[285] T. Lane and C. Brodley. An Application of Machine Learning to Anomaly Detection, *NIST-NCSC National Information Systems Security Conference*, 1997.

[286] T. Lane, and C. Brodley. Sequence matching and learning in anomaly detection for computer security. *AI Approaches to Fraud Detection and Risk Management*, pp. 43–49, 1997.

[287] R. Lasaponara. On the use of Principal Component Analysis (PCA) for Evaluating Interannual Vegetation Anomalies from SPOT/VEGETATION NDVI Temporal Series. *Ecological Modeling*, 194(4), pp. 429–434, 2006.

[288] J. Laurikkala, M. Juholal, and E. Kentala. Informal Identification of Outliers in Medical Data, *Fifth International Workshop on Intelligent Data Analysis in Medicine and Pharmacology*, pp. 20–24, 2000.

[289] A. Lazarevic, and V. Kumar. Feature Bagging for Outlier Detection, *ACM KDD Conference*, 2005.

[290] A. Lazarevic, L. Ertoz, V. Kumar, A. Ozgur, and J. Srivastava. A Comparative Study of Anomaly Detection Schemes in Network Intrusion Detection. *SIAM Conference on Data Mining*, 2003.

[291] S. Q. Le and T. B. Ho. An Association-based Dissimilarity Measure for Categorical Data. *Pattern Recognition Letters*, 26(16), pp. 2549–2557, 2005.

[292] J.-G. Lee, J. Han, and X. Li. Trajectory Outlier Detection: A Partition-and-detect Framework, *ICDE Conference*, 2008.

[293] W. Lee and B. Liu. Learning with Positive and Unlabeled Examples using Weighted Logistic Regression. *ICML Conference*, 2003.

[294] W. Lee, S. Stolfo, and P. Chan. Learning Patterns from Unix Execution Traces for Intrusion Detection, *AAAI workshop on AI methods in Fraud and Risk Management*, 1997.

[295] W. Lee, S. Stolfo, and K. Mok. Adaptive Intrusion Detection: A Data Mining Approach, *Artificial Intelligence Review*, 14(6), pp. 533–567, 2000.

[296] W. Lee, and S. Stolfo. Data Mining Approaches for Intrusion Setection. *Proceedings of the 7th USENIX Security Symposium*, 1998.

[297] W. Lee, and D. Xiang. Information Theoretic Measures for Anomaly Detection, *IEEE Symposium on Security and Privacy*, 2001.

[298] N. Lesh, M. J. Zaki, and M. Ogihara. Mining Features for Sequence Classification, *ACM KDD Conference*, 1999.

[299] C. Leslie, E. Eskin, and W. Noble. The Spectrum Kernel: A String Kernel for SVM Protein Classification, *Pacific Symposium on Biocomputing*, pp. 566–575, 2002.

[300] X. Li, J. Han, S. Kim, and H. Gonzalez. ROAM: Rule and Motif-based Anomaly Detection in Massive Moving Object Data Sets, *SDM Conference*, 2007.

[301] X. Li, B. Liu, and S. Ng. Negative Training Data can be Harmful to Text Classification, *EMNLP*, 2010.

[302] X. Li, Z. Li, J. Han, and J.-G. Lee. Temporal Outlier Detection in Vehicle Traffic Data. *ICDE Conference*, 2009.

[303] D. Lin. An Information-theoretic Definition of Similarity. *ICML Conference*, pp. 296–304, 1998.

[304] J. Lin, E. Keogh, A. Fu, and H. V. Herle. Approximations to Magic: Finding Unusual Medical Time Series, *Mining Medical Data (CBMS)*, 2005.

[305] J. Lin, E. Keogh, S. Lonardi, and B. Y.-C. Chiu. A Symbolic Representation of Time Series, with Implications for Streaming Algorithms. *DMKD Workshop*, 2003.

[306] B. Liu, W. S. Lee, P. Yu, and X. Li. Partially Supervised Text Classification, *ICML Conference*, 2002.

[307] B. Liu, Y. Dai, X. Li, W. S. Lee, P. Yu. Building Text Classifiers Using Positive and Unlabeled Examples. *ICDM Conference*, 2003.

[308] G. Liu, T. McDaniel, S. Falkow, and S. Karlin, Sequence Anomalies in the cag7 Gene of the Helicobacter Pylori Pathogenicity Island, *National Academy of Sciences of the United States of America*, 96(12), pp. 7011–7016, 1999.

[309] L. Liu, and X. Fern. Constructing Training Sets for Outlier Detection, *SDM Conference*, 2012.

[310] F. T. Liu, K. M. Ting, and Z.-H. Zhou. Isolation Forest. *ICDM Conference*, 2008.

[311] S. Lin, and D. Brown. An Outlier-based Data Association Method for Linking Criminal Incidents. *SIAM Conference On Data Mining*, 2003.

[312] X.-Y. Liu, J. Wu, and Z.-H. Zhou. Exploratory Undersampling for Class-Imbalance Learning. *IEEE Transactions on Systems, Man and Cybernetics– Part B, Cybernetics*, 39(2), pp. 539–550, April 2009.

[313] X. Liu, X. Wu, H. Wang, R. Zhang, J. Bailey, and K. Ramamohanarao. Mining Distribution Change in Stock Order Data Streams, *ICDE Conference*, 2010.

[314] Z. Liu, W. Shi, D. Li, and Q. Qin. Partially Supervised Classification – based on Weighted Unlabeled Samples Support Vector Machine. *ADMA*, 2005.

[315] X. Liu, P. Zhang, and D. Zeng. Sequence Matching for Suspicious activity Detection in Anti-money Laundering. *Lecture Notes in Computer Science*, Vol. 5075, pp. 50–61, 2008.

[316] S. Loncarin. A Survey of Shape Analysis Techniques. *Pattern Recognition*, 31(5), pp. 983–1001, 1998.

[317] C.-T. Lu, D. Chen, and Y. Kou. Algorithms for Spatial Outlier Detection, *ICDM Conference*, 2003.

[318] J. Ma and S. Perkins. Online Novelty Detection on Temporal Sequences, *ACM KDD Conference*, 2003.

[319] J. Ma, L. Saul, S. Savage, and G. Volker. Learning to Detect Malicious URLs, *ACM Transactions on Intelligent Systems and Technology*, 2(3), Article 30, April 2011.

[320] M. Mahoney, and P. Chan. Learning Nonstationary Models of Normal Network Traffic for Detecting Novel Attacks, *ACM KDD Conference*, 2002.

[321] M. Mahoney, and P. Chan. Learning Rules for Anomaly Detection of Hostile Network Traffic, *ICDM Conference*, 2003.

[322] F. Malliaros, V. Megalooikonomou, and C. Faloutsos. Fast Robustness Estimation in Large Social Graphs: Communities and Anomaly Detection. *SDM Conference*, 2012.

[323] L. M. Manevitz and M. Yousef. One-class SVMs for Document Classification, *Journal of Machine Learning Research*, 2: pp, 139–154, 2001.

[324] C. Marceau. Characterizing the Behavior of a Program using Multiple-length n-grams, *Workshop on New Security Paradigms*, pp. 101–110, 2000.

[325] M. Markou and S. Singh. Novelty detection: A Review, Part 1: Statistical Approaches, *Signal Processing*, 83(12), pp. 2481–2497, 2003.

[326] M. Markou and S. Singh. Novelty Detection: A Review, Part 2: Neural Network-based Approaches, *Signal Processing*, 83(12), pp. 2481–2497, 2003.

[327] M. Masud, Q. Chen, L. Khan, C. Aggarwal, J. Gao, J, Han, and B. Thuraisingham. Addressing Concept-Evolution in Concept-Drifting Data Streams. *ICDM Conference*, 2010.

[328] M. Masud, T. Al-Khateeb, L. Khan, C. Aggarwal, J. Gao, J. Han, and B. Thuraisingham. Detecting Recurring and Novel Classes in Concept-Drifting Data Streams. *ICDM Conference*, 2011.

[329] M. Masud, Q. Chen, L. Khan, C. Aggarwal, J. Gao, J. Han, A. Srivastava, and N. Oza. Classification and Adaptive Novel Class Detection of Feature-Evolving Data Streams, *IEEE Transactions on Knowledge and Data Engineering*, to appear, Online verion appeared on May 22, 2012.
http://doi.ieeecomputersociety.org/10.1109/TKDE.2012.
109.

[330] C. McDiarmid. On the Method of Bounded Differences. In *Surveys in Combinatorics*, pp. 148–188, *Cambridge University Press*, Cambridge, 1989.

[331] P. Melville and R. Mooney. Diverse Ensembles for Active Learning, *ICML Conference*, 2004.

[332] C. Michael and A. Ghosh. Two State-based Approaches to Program-based Anomaly Detection, *Computer Security Applications Conference*, pp. 21, 2000.

[333] B. Miller, N. Bliss, and P. Wolfe. Subgraph Detection using Eigenvector L1-Norms. *NIPS Conference*, 2010.

[334] B. Miller, M. Beard, and N. Bliss. Eigenspace Analysis for Threat Detection in Social Networks. *International Conference on Information Fusion*, 2011.

[335] D. Mladenic and M. Grobelnik. Feature Selection for Unbalanced Class Distribution and Naive Bayes. *ICML Conference*, 1999.

[336] M. Mongiovi, P. Bogdanov, R. Ranca, A. Singh, E. Papalexakis, and C. Faloutsos. SIGSPOT: Mining Significant Anomalous Regions from Time-evolving Networks. *ACM SIGMOD Conference*, 2012.

[337] E. Muller, M. Schiffer, and T. Seidl. Statistical Selection of Relevant Subspace Projections for Outlier Ranking. *ICDE Conference*, pp, 434–445, 2011.

[338] E. Muller, M. Schiffer, P. Gerwert, M. Hannen, T. Jansen, and T. Seidl, SOREX: Subspace Outlier Ranking Exploration Toolkit, *Joint ECML PKDD Conference*, 2010.

[339] E. Muller, M. Schiffer, and T. Seidl. Adaptive Outlierness for Subspace Outlier Ranking, *CIKM Conference*, 2010.

[340] E. Muller, F. Keller, S. Blanc, and K. Bohm. OutRules: A Framework for Outlier Descriptions in Multiple Context Spaces. *ECML/PKDD Conference*, 2012.

[341] E. Muller, I. Assent, P. Iglesias, Y. Mulle, K. Bohm. Outlier Analysis via Subspace Analysis in Multiple Views of the Data, *ICDM Conference*, 2012.

[342] R. Motwani, and P. Raghavan. Randomized Algorithms, *Cambridge University Press*, 1995.

[343] A. Mueen, E. Keogh, and N. Young. Logical-Shapelets: An Expressive Primitive for Time Series Classification, *ACM KDD Conference*, 2011.

[344] A. Naftel and S. Khalid. Classifying Spatiotemporal Object Trajectories using Unsupervised Learning in the Coefficient Feature Space. *Multimedia Systems*, 12(3), pp. 227–238, 2006.

[345] K. Narita, and H. Kitagawa. Outlier Detection for Transaction Databases using Association Rules, *WAIM*, 2008.

[346] H. Nguyen, V. Gopalkrishnan, and I. Assent, An Unbiased Distance-based Outlier Detection Approach for High Dimensional Data, *DASFAA*, 2011.

[347] V. Niennattrakul, E. Keogh, and C. Ratanamahatana. Data Editing Techniques to Allow the Applicability of Distance-based Outlier Detection in Streams, *ICDM Conference*, 2010.

[348] H. Ning, W. Xu, Y. Chi, Y. Gong, and T. Huang. Incremental Spectral Clustering With Application to Monitoring of Evolving Blog Communities. *SDM Conference*, 2007.

[349] C. Noble, and D. Cook. Graph-based Anomaly Detection, *ACM KDD Conference*, 2003.

[350] P. Olmo Vaz de Melo, L. Akoglu, C. Faloutsos, and A. Loureiro. Surprising Patterns for the Call Duration Distribution of Mobile Phone Users, *ECML/PKDD Conference*, 2010.

[351] M. Otey, S. Parthasarathy, A. Ghoting, G. Li, S. Narravula, and D. Panda. Towards NIC-based Intrusion Detection. *ACM KDD Conference*, 2003.

[352] M. Otey, S. Parthasarathy, and A. Ghoting. Fast Distributed Outlier Detection in Mixed Attribute Data Sets, *Data Mining and Knowledge Discovery*, 12(2–3), pp. 203–228, 2006.

[353] C. R. Palmer and C. Faloutsos. Electricity based External Similarity of Categorical Attributes. *PAKDD Conference*, 2003.

[354] Y. Panatier. Variowin. Software For Spatial Data Analysis in 2D. *New York: Springer-Verlag*, 1996.

[355] C. Papadimitriou, P. Raghavan, H. Tamakai, and S. Vempala. Latent Semantic Indexing: A Probabilistic Analysis, *ACM PODS Conference*, 1998.

[356] S. Papadimitriou, H. Kitagawa, P. Gibbons, and C. Faloutsos. LOCI: Fast Outlier Detection using the Local Correlation Integral. *ICDE Conference*, 2003.

[357] S. Papadimitriou, J. Sun, and C. Faloutsos. SPIRIT: Streaming pattern discovery in multiple time-series. *VLDB Conference*, 2005.

[358] L. Parra, G. Deco, and S. Andmiesbach. Statistical Independence and Novelty Detection with Information Preserving Nonlinear Maps. *Neural Computation*, 8(2), pp. 260–269, 1996.

[359] Y. Pei, O. Zaiane, and Y. Gao. An Efficient Reference-based Approach to Outlier Detection in Large Datasets. *ICDM Conference*, 2006.

[360] D. Pelleg, and A. Moore. Active Learning for Anomaly and Rare Category Detection, *NIPS Conference*, 2004.

[361] Z. Peng, and F. Chu. Review Application of the Wavelet Transform in Machine Condition Monitoring and Fault Diagnostics, *Mechanical Systems and Signal Processing*, 18(2), pp. 199–221, March 2004.

[362] S. Petrovic, M. Osborne, and V. Lavrenko. Streaming First Story Detection with Application to Twitter. *Proceedings of the ACL Conference*, pp. 181–189, 2010.

[363] N. Pham, and R. Pagh. A Near-linear Time Approximation Algorithm for Angle-based Outlier Detection in High-dimensional Data, *ACM KDD Conference*, 2012.

[364] J. Pickands. Statistical Inference using Extreme Order Statistics. *The Annals of Statistics*, 3(1), pp. 119–131, 1975.

[365] B. Pincombe. Anomaly Detection in Time Series of Graphs using ARMA Processes. *ASOR Bulletin*, 24(4): 2–10, 2005.

[366] M. Pinsky. Introduction to Fourier Analysis and Wavelets, *American Mathematical Society*, 2009.

[367] C. Phua, V. Lee, K. Smith, and R. Gayler. A Comprehensive Survey of Data Mining-based Fraud Detection Research. http://arxiv.org/abs/1009.6119.

[368] C. Phua, D. Alahakoon, and V. Lee. Minority Report in Fraud Detection: Classification of Skewed Data, *ACM SIGKDD Explorations Newsletter*, 6(1), pp. 50–59, 2004.

[369] D. Pokrajac, A. Lazerevic, and L. Latecki. Incremental Local Outlier Detection for Data Streams, *CIDM Conference*, 2007.

[370] C. Potter, P. N. Tan, M. Steinbach, S. Klooster, V. Kumar, R. Myneni, and V. Genovese. Major Disturbance Events in Terrestrial Ecosystems detected using Global Satellite Data Sets. *Global Change Biology*, pp. 1005–1021, 2003.

[371] A. Pires, and C. Santos-Pereira. Using Clustering and Robust Estimators to Detect Outliers in Multivariate Data. *International Conference on Robust Statistics*, 2005.

[372] L. Portnoy, E. Eskin, and S. Stolfo. Intrusion Detection with Unlabeled Data using Clustering. *ACM Workshop on Data Mining Applied to Security*, 2001.

[373] B. Prakash, N. Valler, D. Andersen, M. Faloutsos, and C. Faloutsos. BGP-lens: Patterns and Anomalies in Internet Routing Updates. *ACM KDD Conference*, 2009.

[374] M. Prastawa, E. Bullitt, S. Ho, and G. Gerig. A Brain Tumor Segmentation Framework based on Outlier Detection, *Medical Image Analysis*, 8, pp. 275–283, 2004.

[375] C. Priebe, J. Conroy, D. Marchette, and Y. Park. Scan Statistics on Enron Graphs, *Computational and Mathematical Organizational Theory*, 11(3), pp. 229–247, 2005.

[376] F. Provost, and T. Fawcett. Analysis and Visualization of Classifier Performance: Comparison under Imprecise Class and Cost Distributions, *ACM KDD Conference*, 1997.

[377] F. Provost, T. Fawcett, and R. Kohavi. The Case against Accuracy Estimation while Comparing Induction Algorithms, *ICML Conference*, 1998.

[378] G. Qi, C. Aggarwal, and T. Huang. On Clustering Heterogeneous Social Media Objects with Outlier Links, *WSDM Conference*, 2012.

[379] L. R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, *Proceedings of the IEEE*, 77(2), pp. 257–285, Feb. 1989.

[380] M. Radovanovic, A. Nanopoulos, and M. Ivanovic. On the Existence of Obstinate Results in Vector Space Models, *ACM SIGIR Conference*, 2010.

[381] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient Algorithms for Mining Outliers from Large Data Sets. *ACM SIGMOD Conference*, pp. 427–438, 2000.

[382] B. Raskutti and A. Kowalczyk. Extreme Rebalancing for SVMS: A Case Study. *SIGKDD Explorations*, 6(1): pp. 60–69, 2004.

[383] S. Roberts. Novelty Detection using Extreme Value Statistics, *IEEE Proceedings on Vision, Image and Signal Processing*, 146(3). pp. 124–129, 1999.

[384] S. Roberts. Extreme Value Statistics for Novelty Detection in Biomedical Signal Processing. *International Conference on Advances in Medical Signal and Information Processing*. pp. 166–172, 2002.

[385] J. Rogan, J. Miller, D. Stow, J. Franklin, L. Levien, and C. Fischer. Land-Cover Change Monitoring with Classification Trees Using Landsat TM and Ancillary Data. *Photogrammetric Engineering and Remote Sensing*, 69(7), pp. 793–804, 2003.

[386] D. Ron, Y. Singer, and N. Tishby. The Power of Amnesia: Learning Probabilistic Automata with Variable Memory Length, *Machine Learning*, 25(2–3) pp. 117–149, 1996.

[387] P. Rousseeuw and A. Leroy. Robust Regression and Outlier Detection. *Wiley*, 2003.

[388] I. Ruts, and P. Rousseeuw, Computing Depth Contours of Bivariate Point Clouds. *Computational Statistics and Data Analysis*, 23, pp. 153–168, 1996.

[389] M. Sahami, S. Dumais, D. Heckerman, E. Horvitz. A Bayesian approach to filtering junk e-mail. *AAAI Workshop on Learning for Text Categorization. Tech. Rep. WS-98-05.* http://robotics.stanford.edu/users/sahami/papers.html

[390] R. K. Sahoo, A. J. Oliner, I. Rish, M. Gupta, J. E. Moreira, S. Ma, R. Vilalta, and A. Sivasubramaniam. Critical Event Prediction for Proactive Management in Large-scale Computer Clusters. *ACM KDD Conference*, 2003.

[391] G. Salton, and M. J. McGill. Introduction to Modern Information Retrieval, *McGraw Hill*, 1986.

[392] S. Sarawagi, R. Agrawal, and N. Megiddo. Discovery-driven Exploration of OLAP Data Cubes. *EDBT Conference*, 1998.

[393] G. Scarth, M. McIntyre, B. Wowk, and R. Somorjai. Detection of Novelty in Functional Images using Fuzzy Clustering. *Meeting of International Society for Magnetic Resonance in Medicine*, 1995.

[394] R. Schapire and Y. Singer. Improved Boosting Algorithms using Confidence-rated Predictions. *Annual Conference on Computational Learning Theory*, 1998.

[395] E. Schubert, R. Wojdanowski, A. Zimek, and H.-P. Kriegel. On Evaluation of Outlier Rankings and Outlier Scores, *SDM Conference*, 2012.

[396] B. Scholkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7), pp. 1443–1472, 2001.

[397] B. Scholkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt. Support-vector Method for Novelty Detection, *NIPS Conference*, 2000.

[398] R. Schoen, T. Habetler, F. Kamran, and R. Bartfield. Motor Bearing Damage Detection using Stator Current Monitoring. *IEEE Transactions on Industry Applications*, 31(6), pp. 1275–1279, 1995.

[399] K. Sequeira, and M. Zaki. ADMIT: Anomaly-based Data Mining for Intrusions, *ACM KDD Conference*, 2002.

[400] H. Seung, M. Opper, and H. Sompolinsky. Query by Committee. *ACM Workshop on Computational Learning Theory*, 1992.

[401] S. Shekhar, C. T. Lu, and P. Zhang. Detecting Graph-based Spatial Outliers: Algorithms and Applications, *ACM KDD Conference*, 2001.

[402] S.Shekhar, C. T. Lu, and P. Zhang. A Unified Approach to Detecting Spatial Outliers, *Geoinformatica*, 7(2), pp. 139–166, 2003.

[403] S. Shekhar and S. Chawla. A Tour of Spatial Databases. *Prentice Hall*, 2002.

[404] S. Shekhar, C. T. Lu, and P. Zhang. Detecting Graph-based Spatial Outliers, *Intelligent Data Analysis*, 6, pp. 451–468, 2002.

[405] P. Showbridge, M. Kraetzl, and D. Ray. Detection of Abnormal Change in Dynamic Networks. *Proceedings of the Intl. Conf. on Information, Decision and Control*, pp. 557–562, 1999.

[406] M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, and L. Chang. A Novel Anomaly Detection Scheme based on Principal Component Classifier, *ICDM Conference*, 2003.

[407] A. Siebes, J. Vreeken, and M. van Leeuwen. Itemsets than Compress, *SIAM Conference on Data Mining*, 2006.

[408] J. Silva, and R. Willett. Detection of Anomalous Meetings in a Social Network, *SocialCom*, 2008.

[409] B. W. Silverman. Density Estimation for Statistics and Data Analysis. *Chapman and Hall*, 1986.

[410] K. Smets and J. Vreeken. The Odd One Out: Identifying an Characterising Anomalies, *SIAM Conference on Data Mining*, 2011.

[411] E. S. Smirnov. On exact methods in systematics. *Systematic Zoology*, 17(1), pp. 1–13, 1968.

[412] R. Smith, A. Bivens, M. Embrechts, C. Palagiri, and B. Szymanski. Clustering Approaches for Anomaly Based Intrusion Detection. *Intelligent Engineering Systems through Artificial Neural Networks*, 2002.

[413] P. Smyth. Clustering Sequences with Hidden Markov Models, *Neural Information Processing*, 1997.

[414] P. Smyth. Markov Monitoring with Unknown States. *IEEE Journal on Selected Areas in Communications*, 12(9), pp. 1600-1612, 1994.

[415] H. Solberg, and A. Lahti. Detection of Outliers in Reference Distributions: Performance of Horn's Algorithm, *Clinical Chemistry*, 51(12), pp. 2326–2332, 2005.

[416] X. Song, M. Wu, C. Jermaine, and S. Ranka. Conditional Anomaly Detection, *IEEE Transaction on Knowledge and Data Engineering*, 19(5), pp. 631–645, 2007.

[417] X. Song, M. Wu, C. Jermaine, and S. Ranka. Statistical Change Detection for Multidimensional Data, *ACM KDD Conference*, 2007.

[418] C. Spence, L. Parra, and P. Sajda. Detection, Synthesis and Compression in Mammographic Image Analysis with a Hierarchical Image Probability Model. *IEEE Workshop on Mathematical Methods in Biomedical Image Analysis*, 2001.

[419] A. Srivastava. Discovering System Health Anomalies using Data Mining Techniques, *Joint Army Navy NASA Airforce Conference on Propulsion*, 2005.

[420] A. Srivastava. Enabling the Discovery of Recurring Anomalies in Aerospace Problem Reports using High-dimensional Clustering Techniques. *Aerospace Conference*, 2006.

[421] A. Srivastava, and B. Zane-Ulman. Discovering Recurring Anomalies in Text Reports regarding Complex Space Systems. *Aerospace Conference*, 2005.

[422] A. Srivastava, A. Kundu, S. Sural, and A. Majumdar. Credit card fraud detection using Hidden Markov Model. *IEEE Transactions on Dependable and Secure Computing*, 5(1), pp. 37–48, 2008.

[423] P. Srivastava, D. Desai, S. Nandi, and A. Lynn. HMM-ModE-Improved Classification using Profile Hidden Markov Models by Optimizing the Discrimination Threshold and Modifying Emission Probabilities with Negative Training Sequences, *BMC Bioinformatics*, 8 (104), 2007.

[424] M. Stephens. Use of the Kolmogorov-Smirnov, Cramer-von Mises and Related Statistics without Extensive Tables, *Journal of the Royal Statistical Society*. Series B, pp. 115–122, 1970.

[425] S. Stolfo, D. Fan, W. Lee, A.L. Prodromidis, and P. Chan. Credit Card Fraud Detection Using Meta-Learning: Issues and Initial Results, *AAAI Workshop AI Methods in Fraud and Risk Management*, pp. 83–90, 1997.

[426] S. Stolfo, D. Fan, W. Lee, A. Prodromidis, and P. Chan. Cost-Based Modeling for Fraud and Intrusion Detection: Results from the JAM Project, *DARPA Information Survivability Conf. and Exposition*, 2, pp. 130–144, 2000.

[427] S. Subramaniam, T. Palpanas, D. Papadopoulos, V. Kalogeraki, and D. Gunopulos. Online Outlier Detection in Sensor Data using Non-parametric Models. *VLDB Conference*, 2006.

[428] H. Sun, Y. Bao., F. Zhao, G. Yu, and D. Wang. CD-Trees: An Efficient Index Structure for Outlier Detection. *Web-Age Information Management (WAIM)*, pp. 600–609, 2004.

[429] J. Sun, S. Papadimitriou, P. Yu, and C. Faloutsos. Graphscope: Parameter-free Mining of Large Time-Evolving Graphs, *ACM KDD Conference*, 2007.

[430] J. Sun, D. Tao, and C. Faloutsos. Beyond Streams and Graphs: Dynamic Tensor Analysis, *ACM KDD Conference*, 2006.

[431] J. Sun, H. Qu, D. Chakrabarti, and C. Faloutsos. Neighborhood Formation and Anomaly Detection in Bipartite Graphs. *ICDM Conference*, 2005.

[432] J. Sun, Y. Xie, H. Zhang, and C. Faloutsos. Less is More: Compact Matrix Representation of Large Sparse Graphs. *SIAM Conference on Data Mining*, 2007.

[433] P. Sun, and S. Chawla. On Local Spatial Outliers, *IEEE ICDM Conference*, 2004.

[434] P. Sun, S. Chawla, and B. Arunasalam. Mining for Outliers in Sequential Databases, *SIAM International Conference on Data Mining*, 2006.

[435] Y. Sun, Y. Yu, and J. Han. Ranking-based Clustering of Heterogeneous Information Networks with Star Network Schema. *ACM KDD Conference*, 2009.

[436] C. Surace, and K. Worden. A Novelty Detection Method to Diagnose Damage in Structures: An Application to an Offshore Platform. *International Conference of Offshore and Polar Engineering*, 4, pp. 64–70, 1998.

[437] C. Surace, K. Worden, and G. Tomlinson. A Novelty Detection Approach to Diagnose Damage in a Cracked Beam. *Proceedings of SPIE*, Vol. 3089, pp. 947–953, 1997.

[438] E. Suzuki, T. Watanabe, H. Yokoi, and K. Takabayashi. Detecting Interesting Exceptions from Medical Test Data with Visual Summarization. *International Conference on Data Mining*, pp. 315–322, 2003.

[439] T. Takahashi, R. Tomioka, and K. Yamanishi. Discovering Emerging Topics in Social Streams via Link Anomaly Detection. *ICDM Conference*, 2011.

[440] P.-N. Tan and V. Kumar. Discovery of Web Robot Sessions based on their Navigational Patterns, *Data Mining and Knowledge Discovery*, 6(1), pp. 9–35, 2002.

[441] Y. Tao, X. Xiao, and S. Zhou. Mining Distance-based Outliers from Large Databases in any Metric Space. *ACM KDD Conference*, 2006.

[442] J. Tang, Z. Chen, A. W.-C. Fu, D. W. Cheung. Enhancing Effectiveness of Outlier Detections for Low Density Patterns. *PAKDD Conference*, 2002.

[443] Y. Tang, Y.-Q. Zhang, N. V. Chawla, and S. Krasser. SVMs Modeling for Highly Imbalanced Classification, *IEEE Transactions on Systems, Man and Cybernetics- Part B: Cybernetics*, 39(1), pp. 281–288, 2009.

[444] M. Taniguchi, M. Haft, J. Hollmen, and V. Tresp. Fraud Detection in Communications Networks using Neural and Probabilistic Methods. *IEEE International Conference in Acoustics, Speech and Signal Processing*, 2, pp. 1241–1444, 1998.

[445] D. Tax. One Class Classification: Concept-learning in the Absence of Counter-examples, *Doctoral Dissertation, University of Delft*, Netherlands, 2001.

[446] L. Tarassenko. Novelty detection for the Identification of Masses in Mammograms. *IEEE International Conference on Artificial Neural Networks*, 4, pp. 442–447, 1995.

[447] P. Thompson, D. MacDonald, M. Mega, C. Holmes, A. Evans, and A. Toga. Detection and Mapping of Abnormal Brain Structure with a Probabilistic Atlas of Cortical Surfaces. *Journal of Computer Assisted Tomography*, 21(4), pp. 567–581, 1997.

[448] M. Thottan, and C. Ji. Anomaly Detection in IP Networks. *IEEE Transactions on Signal Processing*, 51(8), pp. 2191–2204, 2003.

[449] S. Tian, S. Mu, and C. Yin. Sequence-similarity Kernels for SVMs to Detect Anomalies in System Calls, *Neurocomputing*, 70(4–6), pp. 859–866, 2007.

[450] K. M. Ting. An Instance-weighting Method to Induce Cost-sensitive Trees. *IEEE Transaction on Knowledge and Data Engineering*, 14: pp. 659–665, 2002.

[451] M. E. Tipping, and C. M. Bishop. Probabilistic Principal Component Analysis. *Journal of the Royal Statistical Society*, B 61, pp. 611–622, 1999.

[452] H. Tong, and C.-Y. Lin. Non-Negative Residual Matrix Factorization with Application to Graph Anomaly Detection. *SDM Conference*, 2011.

[453] V. Venkatasubramanian, R. Rengaswamy, K. Yin, and S. N. Kavuri. A Review of Process Fault Detection and Diagnosis Part I: Quantitative Model-based Methods. *Computers and Chemical Engineering*, 27(3), pp. 293–311, 2003.

[454] J. S. Vitter. Random sampling with a reservoir, *ACM Trans. Math. Softw.*, vol. 11(1), pp. 37–57, 1985.

[455] W. Tobler. Cellular geography. In *Philosophy in Geography, Dordrecht Reidel Publishing Company*, pp. 379–386, 1979.

[456] S. Viaene, R. Derrig, B. Baesens, and G. Dedene. A Comparison of State-of-the-Art Classification Techniques for Expert Automobile Insurance Claim Fraud Detection, *Journal of Risk and Insurance*, 69(3), pp. 373–421, 2002.

[457] N. Wale, X. Ning, and G. Karypis. Trends in Chemical Data Mining, *Managing and Mining Graph Data, Springer*, 2010.

[458] X. Wang, C. Zhai, X. Hu and R. Sproat. Mining Correlated Bursty Topic Patterns from Coordinated Text Streams. *ACM KDD Conference*, 2007.

[459] B. Wang, G. Xiao, H. Yu, and X. Yang, Distance-based Outlier Detection on Uncertain Data, *International Conference on Computer and Information Technology*, 2009.

[460] B. Wang, X. Yang, G. Wang, and G. Yu. Outlier Detection over Sliding Windows for Probabilistic Data Streams, *Journal of Computer Science and Technology*, 25(3), pp. 389–400, 2010.

[461] L. Wei, W. Qian, A. Zhou, and W. Jin. HOT: Hypergraph-based Outlier Test for Categorical Data. *PAKDD Conference*, 2007.

[462] L. Wei and E. Keogh. Semi-supervised Time Series Classification, *ACM KDD Conference*, 2006.

[463] G. Weiss and F. Provost. Learning When Training Data are Costly: The Effect of Class Distribution on Tree Induction, *Journal of Artificial Intelligence Reserach*, 19: pp. 315–354, 2003.

[464] G. Williams, R. Baxter, H. He, S. Hawkings, and L. Gu. A Comparative Study of RNN for Outlier Detection in Data Mining. *IEEE ICDM Conference*, 2002.

[465] W.-K. Wong, A. Moore, G. Cooper, and M. Wagner. Rule-based Anomaly Pattern Detection for Detecting Disease Outbreaks, *National Conference on Artificial Intelligence*, 2002.

[466] K. van Leemput, F. Maes, D. Vandermeulen, A. Colchester, and P. Suetens. Automated Segmentation of Multiple Sclerosis Lesions by Model Outlier Detection, *IEEE Transactions on Medical Imaging*, vol. 20, pp. 677–688, August 2001.

[467] T. De Vries, S. Chawla, and M. Houle. Finding Local Anomalies in Very High Dimensional Space, *ICDM Conference*, 2010.

[468] M. Wang, C. Zhang, and J. Yu. Native API-based Windows Anomaly Intrusion Detection Method using SVM, *International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*, 2006.

[469] L. Wei, E. Keogh, and X. Xi. SAXually Exaplicit Images: Finding Unusual Shapes, *ICDM Conference*, 2006.

[470] G. Wu and E. Y. Chang. Class-boundary Alignment for Imbalanced Dataset Learning. *Proceedings of the ICML Workshop on Learning from Imbalanced Data Sets*, 2003.

[471] M. Wu, and C. Jermaine. Outlier Detection by Sampling with Accuracy Guarantees. *ACM KDD Conference*, 2006.

[472] E. Wu, W. Liu, and S. Chawla. Spatio-temporal Outlier Detection in Precipitation Data, *Knowledge Discovery from Sensor Data, Springer, LNCS 5840*, 2008.

[473] M. Wu, X. Song, C. Jermaine, S. Ranka, and J. Gums. A LRT Framework for Fast Spatial Anomaly Detection. *ACM KDD Conference*, 2009.

[474] X. Xi, E. Keogh, C. Shelton, L.Wei, and C. Ratanamahatana. Fast Time Series Classification using Numerosity Reduction, *ICML Conference*, 2006.

[475] Z. Xing, J. Pei, and E. Keogh. A Brief Survey on Sequence Classification, *ACM SIGKDD Explorations*, 12(1), 2010.

[476] L. Xiong, X. Chen, and J. Schneider. Direct Robust Matrix Factorization for Anomaly Detection. *ICDM Conference*, 2011.

[477] L. Xiong, B. Poczos, J. Schneider, A. Connolly, and J. VanderPlas. Hierarchical Probabilistic Models for Group Anomaly Detection, *Artificial Intelligenece and Statitistics*, 2011.

[478] K. Yaminshi, J. Takeuchi, and G. Williams. Online Unsupervised Outlier Detection using Finite Mixtures with Discounted Learning Algorithms, *ACM KDD Conference*, 2000.

[479] K. Yaminshi, and J. Takeuchi. A Unified Framework for Detecting Outliers and Change Points from Time Series Data, *ACM KDD Conference*, 2002.

[480] R. Yan, Y. Liu, R. Jin, and A. Hauptmann. On Predicting Rare Classes with SVM Ensembles in Scene Classification. *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2003.

[481] J. Yang, and W. Wang. CLUSEQ: Efficient and Effective Sequence Clustering, *ICDE Conference*, 2003.

[482] P. Yang, and Q. Zhu. Finding Key Outlying Subspaces for Outlier Detection, *Knowledge-based Systems*, 24(2), pp. 269–274, 2011.

[483] X. Yang, L. Latecki, and D. Pokrajac. Outlier Detection with Globally Optimal Exemplar-based GMM. *SDM Conference*, 2009.

[484] Y. Yang, J. Carbonell, R. Brown, T. Pierce, B. T. Archibald, and X. Liu. Learning Approaches for Detecting and Tracking News Events. *IEEE Intelligent Systems*, 14(4):32–43, 1999.

[485] Y. Yang, T. Pierce, and J. Carbonell. A Study on Retrospective and On-line Event Detection. *ACM SIGIR Conference*, 1998.

[486] Y.Yang, J. Zhang, J. Carbonell, and C. Jin. Topic-conditioned Novelty Detection. *ACM KDD Conference*, 2002.

[487] D. Yankov, E. Keogh, and U. Rebbapragada. Disk-aware Discord Discovery: Finding Unusual Time Series in Terabyte Sized Data Sets, *ICDM Conference*, 2007.

[488] N. Ye. A Markov Chain Model of Temporal Behavior for Anomaly Detection, *IEEE Information Assurance Workshop*, 2004.

[489] N. Ye, and Q. Chen. An Anomaly Detection Technique based on a Chi-square Statistic for Detecting Intrusions into Information Systems. *Quality and Reliability Engineering International*, 17, pp. 105–112, 2001.

[490] L. Ye and E. Keogh. Time Series Shapelets: a New Primitive for Data Mining. *ACM KDD Conference*, 2009.

[491] B.-K. Yi, N. D. Sidiropoulos, T. Johnson, H. Jagadish, C. Faloutsos, and A. Biliris. Online Data Mining for Co-evolving Time Sequences. *ICDE Conference*, 2000.

[492] D. Yu, G. Sheikholeslami, and A. Zhang. Findout: Finding Outliers in Very Large Datasets. *Knowledge And Information Systems*, 4(4), pp. 387–412, 2002.

[493] H. Yu, J. Han, and K. C.-C. Chang. PEBL: Web Page Classification without Negative Examples. *IEEE Transactions on Knowledge and Data Engineering*, 16(1), pp. 70–81, 2004.

[494] J. X. Yu, W. Qian, H. Lu, and A. Zhou. Finding Centric Local Outliers in Categorical/Numeric Spaces. *Knowledge and Information Systems*, 9(3), pp. 309–338, 2006.

[495] B. Zadrozny, and C. Elkan. Transforming Classifier Scores into Accurate Multiclass Probability Estimates. *ACM KDD Conference*, 2002.

[496] B. Zadrozny, J. Langford, and N. Abe. Cost-Sensitive Learning by Cost-Proportionate Example Weighting, *ICDM Conference*, 2003.

[497] B. Zadrozny, and C. Elkan. Learning and Making Decisions when Costs and Probabilities are Unknown, *KDD Conference*, 2001.

[498] J. Zhang, Q. Gao, and H. Wang. SPOT: A System for Detecting Projected Outliers from High-Dimensional Data Stream, *ICDE Conference*, 2008.

[499] J. Zhang, M. Lou, T. W. Ling and H. Wang. HOS-Miner: A System for Detecting Outlying Subspaces of High-dimensional Data. *VLDB Conference*, 2004.

[500] J. Zhang, Q. Gao and H. Wang. A Novel Method for Detecting Outlying Subspaces in High-dimensional Databases Using Genetic Algorithm. *ICDM Conference*, 2006.

[501] J. Zhang and H. Wang. Detecting Outlying Subspaces for High-Dimensional Data: the New Task, Algorithms and Performance. *Knowledge and Information Systems*, 10(3), pp. 333–355, 2006.

[502] Y. Zhang, P. Meratnia, and P. Havinga. Outlier Detection for Wireless Sensor Networks: A Survey. *IEEE Communications Surveys and Tutorials*, 12(2), 2010.

[503] J. Zhang, Z. Ghahramani, and Y. Yang. A Probabilistic Model for Online Document Clustering with Application to Novelty Detection. *NIPS*, 2005.

[504] D. Zhang, and G. Lu. Review of Shape Representation and Description Techniques. *Pattern Recognition*, 37(1), pp. 1–19, 2004.

[505] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An Efficient Data Clustering Method for Very Large Databases. *ACM SIGMOD Conference*, 1996.

[506] J. Zhang and I. Mani. KNN Approach to Unbalanced Data Distributions: A Case Study Involving Information Extraction. *Proceedings of the ICML Workshop on Learning from Imbalanced Datasets*, 2003.

[507] D. Zhang and W. S. Lee. A Simple Probabilistic Approach to Learning from Positive and Unlabeled Examples. *Annual UK Workshop on Computational Intelligence*, pp. 83–87, 2005.

[508] X. Zhang, P. Fan, and Z. Zhu. A New Anomaly Detection Method based on Hierarchical HMM, *International Conference on Parallel and Distrbuted Computing, Applications, and Technologies*, 2003.

[509] Y. Zhang, J. Hong, and L. Cranor. CANTINA: A Content-based Approach to Detecting Phishing Web Sites. *WWW Conference*, 2007.

[510] J. Zhao, C.-T. Lu, and Y. Kou. Detecting Region Outliers in Meteorological Data. *ACM GIS Conference*, 2003.

[511] Z. Zheng, X. Wu, and R. Srihari. Feature Selection for Text Categorization on Imbalanced Data. *SIGKDD Explorations*, 6(1), pp. 80–89, 2004.

[512] C. Zhu, H. Kitagawa, S. Papadimitriou, and C. Faloutsos. OBE: Outlier by Example, *PAKDD Conference*, 2004.

[513] C. Zhu, H. Kitagawa, and C. Faloutsos. Example-based Robust Outlier Detection in High Dimensional Data Sets, *ICDM Conference*, 2005.

[514] A. Zimek, A. Schubert, and H.-P. Kriegel. A Survey on Unsupervised Outlier Detection in High-dimensional Numerical Data, *Journal on Statistical Analysis and Data Mining*, Preprint available at the online Wiley library: http://onlinelibrary.wiley.com/doi/10.1002/sam.11161/abstract, 2012.

[515] http://www.itl.nist.gov/iad/mig/tests/tdt/tasks/fsd.html

[516] D. D. Lewis. Reuters-21578 Data Set.
http://www.daviddlewis.com/resources/test-collections/reuters21578.

[517] http://kdd.ics.uci.edu/databases/20newsgroups

[518] http://www.informatik.uni-trier.de/~ley/db/

[519] http://www.kdnuggets.com/software/deviation.html

[520] http://www.kdnuggets.com/software/index.html

[521] http://www.cs.waikato.ac.nz/ml/weka/

[522] http://www.cs.ucr.edu/~eamonn/time_series_data/

[523] http://www.cs.ucr.edu/~eamonn/SAX.htm

[524] http://www-935.ibm.com/services/nz/en/it-services/ibm-proventia-network-anomaly-detection-system-ads.html

[525] http://www.ibm.com/software/analytics/spss

[526] http://www-01.ibm.com/software/analytics/spss/products/statistics/

[527] http://www.sas.com/

[528] http://www.sas.com/software/security-intelligence/index.html

[529] http://www.oracle.com/technetwork/database/options/advanced-analytics/odm/index.html

[530] http://www.wizsoft.com

Index