Detecting Patterns of Anomalies

Kaustav Das

CMU-CS-09-xxx

March 2009

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Jeff Schneider (Chair)
Christos Faloutsos
Geoffrey Gordon
Daniel Neill
Gregory Cooper, University of Pittsburgh

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy.

This publication was supported in part by Grant Number 8-R01-HK000020-02 from CDC and by NSF under award IIS-0325581.

The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the sponsors.

Keywords: Machine Learning, Anomaly Detection, Bayesian Network, Biosurveillance

Abstract

An anomaly is an observation that does not conform to the expected *normal* behavior. With the ever increasing amount of data being collected universally, automatic surveillance systems are becoming more popular and are increasingly using data mining methods to detect patterns of anomalies. Detecting anomalies can provide useful and actionable information in a variety of real-world scenarios. For example, in disease monitoring, a timely detection of an epidemic can potentially save many lives.

The diverse nature of real-world datasets, and the difficulty of obtaining labeled training data make it challenging to develop a universal framework for anomaly detection. We focus on a key feature of most real world scenarios, that multiple anomalous records are usually generated by a common anomalous process. In this thesis we develop methods that utilize the self-similarity of these *groups* or *patterns* of anomalies to perform better detection. We also investigate new methods for detection of individual record anomalies, which we then incorporate into the group detection methods. A recurring feature of our methods is combinatorial search over some space (e.g. over all subsets of attributes, or over all subsets of records). We use a variety of computational speedup tricks and approximation techniques to make these methods scalable to large datasets. Since, most of our motivating problems involve datasets having categorical or symbolic values, we focus on categorical valued datasets. Apart from this, we make few assumptions about the data, and our methods are very general and applicable to a wide variety of domains.

Additionally, we investigate anomaly pattern detection in data structured by space and time. Our method generalizes the popular method of spatio-temporal scan statistics to learn and detect specific, time-varying spatial patterns in the data. Finally, we show an efficient and easily interpretable technique for anomaly detection in multivariate time series data. We evaluate our methods on a variety of real world data sets including both real and synthetic anomalies.

Acknowledgments

First I would like to thank my advisor Jeff Schneider, whose constant encouragement and valuable insights provided the perfect guidance that I needed through my graduate studies. I would also like to thank Daniel Neill for the stimulating discussions and comments that led to much of this work. I am thankful to Andrew Moore who advised me for a year, for his support and guidance.

I would like to thank my other committee members, Christos Faloutsos, Geoff Gordon and Greg Cooper for their valuable advice and comments. I would also like to thank all my colleagues in the Auton Laboratory and the Machine Learning Department for their support and friendship. A special thank goes to Diane Stidle, for her efficient management of the department and for patiently answering all my questions.

I am very grateful to all my friends in Pittsburgh for the wonderful time we spent together. They helped me make this city a home away from home.

Finally, I thank my parents and sister for their endless love and constant support. Without them I would not be where I am today.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Anomaly Detection

1.1.1 Motivation and Challenges

An anomaly is an observation or a pattern of observations that does not conform to the expected *normal*[1] behavior of the data. With the ever increasing amount of data being collected universally, it gets more important and challenging to spot unusual or unexpected observations. Such unexpected behavior might either be unwanted (e.g. in network intrusion detection, disease surveillance), requiring user intervention, or it might be interesting (e.g. in astronomy), leading to a better understanding of the system. The task of anomaly detection assumes an important role, since in most cases, the detection of anomalies results in actionable information, whereby we can either prevent or mitigate the effects of an undesired situation. For example, in biosurveillance, we want to detect causes such as epidemics or bioterrorist attacks which give rise to unusual patterns of emergency department records. Timely detection of such phenomena leading to appropriate action can save many lives.

[1]We use the word "normal" here in layman's terms, not as a reference to the normal distribution in statistics

Hence, automatic surveillance systems are becoming more popular and are increasingly using data mining methods to perform detection. The observation of industrial manufacturing processes is one traditional application of such systems. Time-series data from various sensors are monitored to detect out of control processes. Another common application is public health monitoring, where patient data from hospitals and sales data from pharmacies are monitored with the goal of detecting new disease outbreaks as early as possible [Wong et al., 2003, Neill et al., 2005c]. In recent times, searching for terrorist activity or attacks has attracted a lot of attention. Applications in that area include monitoring human health and behavioral data to detect a chemical or biological attack [Hogan et al., 2007], or searching for signs of radiation to detect development or deployment of nuclear devices [Theiler and Cai, 2003]. Other applications of anomaly detection include credit card fraud detection [Aleskerov et al., 1997], insider trading detection [Das et al., 2004b], image processing [Chen et al., 2005, Theiler and Cai, 2003, Augusteijn and Folkert, 2002] and traffic monitoring [Shekhar et al., 2001].

An important challenge in anomaly detection is the difficulty to obtain enough labeled data to characterize anomalies. Hence, in most cases we need to operate in an *unsupervised* setting, where only the normal behavior is characterized, and is used to detect deviations from it. In our data mining framework, it is usually assumed that we have a sufficiently large training dataset that contains no or very few anomalous cases. This dataset is assumed to define the normal behavior of the system. Along with this, we also need an anomalousness measure or score to compare new observations to the usual state. Given this scoring method, any observation that significantly deviates from the usual is flagged as an anomaly.

Another challenge in forming a universal framework for anomaly detection is that the definition of anomalies and that of normality is typically very domain specific. This has led to domain specific efforts in this area based on factors like the type of anomalies, the nature of the data, the availability of data labels and other constraints. In light of these factors, we present the relationship between the different anomaly detection techniques presented in this thesis.

1.1.2 Detection of Anomalous Groups and Patterns

The data is usually a collection of *records*, each of which is described by a set of *attributes* (or *features*). Broadly speaking, anomalies might be either *individual* anomalies (corresponding to a single record) or *collective* anomalies (corresponding to groups of records). For example, in credit card fraud detection, it is important to monitor each transaction to determine if they conform to the usual behavior of the customer. In this case we would like to use an anomaly detection method that tests each record individually and searches for single record anomalies. On the other hand, in disease surveillance, we wish to detect disease outbreaks which give rise to unusual patterns of emergency department patients or of medicine sales (consisting of multiple records).

For the case of individual anomaly detection, a standard approach is to create a model of normal data, and compare test records against it. A probabilistic approach builds a likelihood model from the training data. Records are tested for anomalies based on the complete record likelihood given the probability model. While this approach is good at finding outliers in the dataset, it often tends to detect records with attribute values that are rare, especially when the attributes have a high arity. Sometimes, in such cases, just detecting rare values of an attribute is not desired and such outliers are not considered as anomalies in that context. Hence in Chapter 2, we present an alternative definition of anomalies, and propose an approach of comparing attribute values against the marginal distribution of subsets of attributes. We show that this is a more meaningful way of detecting anomalies, and has a better performance for real world datasets.

In the case of collective anomalies, rather than finding individually anomalous records (which may be due to noise or errors in the data), we are more interested in detecting the emergence of new phenomena resulting in patterns of anomalous observations that cannot be explained by a previous model. In general, these activities give rise to multiple records in the dataset which are anomalous, but are similar to each other. Our goal here is to utilize the presence of such multiple cases to better detect collections (or *groups*) of anomalous records, as compared to searching for individual records.

In certain cases, the attributes can be divided into two distinct sets, the *contextual* and

the *behavioral* attributes [Song et al., 2007]. The contextual attributes specify the context (e.g. spatial information such as geographical coordinates or time) and the behavioral attributes then determine whether or not the records are anomalous within the given context. While monitoring emergency department cases, the zipcode and the date can be treated as contextual attributes, whereas the symptom of the patient is a behavioral attribute.

Most of the previous work that aims to detect groups of anomalies, assume some form of contextual information. In this case, the definition of a group relies on the similarity between records with respect to these contextual attributes. For example, in spatial scan, a group is defined as a set of geographically adjacent locations. And given a specific geographical region, the number of patients with a particular symptom (behavioral attribute) determine whether or not a disease outbreak has occurred in the region [Neill et al., 2005c, Kulldorff, 1997]. One of the main contributions of this thesis is to detect groups of anomalies in the more general case, where we do not restrict the contextual information to any particular subset. In this case, all the attributes take on the dual role of defining similarity between cases (the contextual information used to define groups), and indicating whether they display anomalous behavior (behavioral information). Alternatively, we can say that the contextual and behavioral attributes completely overlap, and are the same set. One central idea common to of most of our methods is that they usually aim to perform a combinatorial search over some space (e.g. over all subsets of attributes, or over all subsets of records), to overcome the lack of pre-defined contextual information. We use a variety of computational speedup tricks and approximation techniques to make these methods scalable to large datasets.

In this context of collective anomalies, we can broadly think of two possible scenarios. In the first scenario, an anomalous process generates records which are loosely similar to each other, based on one (or few) attribute value(s). Here, we assume that the underlying process is constrained such that it only has access to a fixed (but unknown) subset of the data. For example, in customs monitoring, a smuggler might be operating only from a fixed port of arrival, or might have access only to a particular shipping line. But within that subset, the smuggler will try to hide their activities by making them appear as random as possible. Similarly, in monitoring emergency department visits, a bioterrorist might

have access to only a particular geographical location, or to only a particular type of disease causing agent. Thus these activities give rise to multiple anomalous records which share common values in some (small) subset of their attributes. Since such patterns are loosely similar, we cannot use their similarity alone to detect them. Instead in this case, we assume that most of the records belonging to such a pattern stand out on their own as individual (or local) anomalies. In Chapter 3, we take a two step approach where we first use a "local anomaly detector" (such as the Conditional Method described in Chapter 2) with a low threshold setting, to flag individual anomalies. We then develop a rule based Anomaly Pattern Detector (APD) which detects anomalous patterns in subsets of the data and improves the detection performance (giving a lower false positives rate).

The second scenario is where all the records generated by an anomalous process are very self-similar. For example, in the case of network intrusion, the same task might be repeated a number of times to gain unauthorized access to a system. In health monitoring, a disease outbreak can lead to a large number of disease cases with almost identical features being reported. In this scenario, it is possible that the individual cases corresponding to an anomalous group might not stand out by themselves, but as a group they appear anomalous. Our method of Anomaly Group Detection (AGD) aims to address this problem, when there are a many self-similar anomalous cases, which might not be very anomalous on their own (Chapter 4).

We deal with anomaly detection in the presence of contextual information in Chapter 5 (TV-MBSS) and Chapter 6 (composite time series anomaly detection), focusing on the cases of spatio-temporal and purely temporal data from multiple time series respectively.

1.1.3 Other features of Anomaly Detection algorithms

Each attribute of a dataset can either be *real* or *categorical* valued. Comparison between attribute values in the case of real valued attributes is simple and straightforward. This has led to the development of a very diverse set of methods to model and analyze data having real valued features. Comparatively, it is less simple to model the relationship between attribute values in case of categorical (or symbolic) values. In this case, there is no inherent

ordering of the values, and their inter-relationships are often domain specific. Because of this added complexity, there are fewer methods that deal with categorical valued attributes. Note that any method that can deal with categorical valued attributes can usually be applied to real valued attributes by discretizing them (binning into a fixed set of buckets, based on quantiles). But the converse is not true in general. Hence the majority of the work in this thesis (Chapters 2, 3 and 4) focuses on datasets having categorical valued attributes. Any real valued attribute is assumed to be discretized using a suitable method. Chapter 5 deals with real valued attributes in the form of space-time coordinates as well as aggregated counts. Chapter 6 deals with time-series data which is also real valued.

Another important feature of an anomaly detection method is the availability (or unavailability) of data labels. In practice obtaining proper labels (whether each data record is normal or anomalous) is a difficult and time consuming task. Often the only reliable source is for an expert to hand label cases. On the other hand it is very easy to obtain data without any labels, since it is usually collected on a daily basis. There are three classes of methods - *supervised*, *semi-supervised* and *unsupervised*. Supervised methods assume that we have a fully labeled training dataset with both normal and anomalous class instances. Usually this is used to train an appropriate classification method, which is then used to classify new test records. Since it is usually not practical to assume a fully labeled training dataset, we mainly focus on the other two categories. In the semi-supervised approach, we usually have a few labeled cases, either from the normal or the anomalous class (or a few instances from both classes) that are hand labeled. The training data usually consists of these labeled instances along with a large number of unlabeled cases. The method described in Chapter 5 belongs to this category. The unsupervised setting assumes the absence of any labels in the training data. However, we typically assume that most of the cases in the training data are normal with the presence of very few anomalous cases. Thus a model of *normal behavior* can be learned from the training data. For the purpose of anomaly detection, unsupervised methods are very useful for two reasons. First, they do not rely on the availability of expensive and difficult to obtain data labels and second, they do not assume any specific characteristics of the anomalies. In many cases, it is important to detect unexpected or unexplained behavior which cannot be pre-specified. Since the unsupervised approach relies on detecting any observation that deviates from the normal

data cases, it is not restricted to any particular type of anomaly. Hence, we have mainly focused on unsupervised methods in this thesis, and the methods described in Chapters 2,3,4 and 6 belong to this category.

1.2 Contributions

The main contributions of this thesis can be summarized as follows:

- We propose several novel techniques of unsupervised anomaly detection in categorical valued datasets, that can handle attributes having a large number of possible values. The four different techniques proposed in this context are applicable over a range of anomaly generation scenarios.
- The Conditional Method and Marginal Method of anomaly detection described in Chapter 2 are applicable in detecting single record anomalies that stand out on their own. In this case, we show that comparing attribute values against the marginal distribution of all subsets of attributes gives better performance than calculating the complete likelihood of the record in real world anomaly detection scenarios.
- Under the condition that multiple anomalies are generated by a single process, all of which share some common attribute value(s), we propose the Anomaly Pattern Detection (APD) method in Chapter 3. This method detects collections of anomalies that are individually somewhat anomalous and are similar to the extent that they share some common attribute values. This method is a combination of the "local anomaly detection" methods and rule-based methods, and we demonstrate that it performs better at detection than either of these methods alone.
- We present another technique, for detecting anomalous groups of records in Chapter 4. This Anomalous Group Detection (AGD) method assumes that there are multiple similar anomalies generated by a common process. The anomalies may or may not stand out on their own as being anomalous. This method utilizes a likelihood ratio statistic which incorporates both the anomalousness and self-similarity of a group.

Table 1.1: Summary of Methods

Chapter No.	Method Name	Type of Anomalies Detected	Type of Dataset
2	Conditional Method	Individual record anomalies, ignoring rare values	Categorical valued
	Marginal Method	Individual record anomalies, including those due to rare values	Categorical valued
3	Anomaly Pattern Detection (APD)	Anomalous groups of records, with low self-similarity within the group, but high individual anomalousness scores	Categorical valued
4	Anomalous Group Detection (AGD)	Anomalous groups of records, with high self-similarity within the group, but low individual anomalousness scores	Categorical valued
5	Time Varying - Multivariate Bayesian Scan Statistics	Time varying spatio-temporal patterns	Real valued (Multivariate space-time data)
6	Arithmetic combinations of time series	Increases (or decreases) in time series counts	Real valued (Multivariate time series)

- In Chapter 5, we propose another technique of detecting collection of anomalies with specific temporal and spatial patterns. Our Time Varying Multivariate Bayesian Scan Statistic (TV-MBSS) extends the previously proposed method Multivariate Bayesian Scan Statistic (MBSS) to detect events having specific temporal patterns in multivariate datasets. Furthermore, we also model events that move over time and incorporate semi-supervised learning of the temporal patterns from data.

- Finally, we investigate an intuitive multivariate time-series anomaly detection method, which searches over simple arithmetic combinations of the different time-series. We show the effectiveness of this procedure over other traditional multivariate approaches.

1.3 Related Work

Anomaly detection has been applied in various domains and on different types of datasets. First, we present popular anomaly detection methods for temporal and spatial analysis. Next, we give an overview of existing methods for detecting anomalous records in large multivariate datasets. This technique has received a lot of attention in detecting intrusions in networks. Hence, we first give an overview of different approaches in this context. This is followed by a description of methods that apply to categorical datasets in an unsupervised setting.

1.3.1 Time Series Anomaly Detection

One of the most popular uses of automatic surveillance systems is in monitoring time series data to detect any abnormalities. The observation of industrial manufacturing systems is one traditional application of these systems. A more recent interest is in public health monitoring which has the goal of detecting new disease outbreaks as early as possible. A simple method to monitoring time series data is to place a restriction on the maximum and minimum tolerable values (e.g. three standard deviations below or above the expected

value), and to sound an alarm if the signal moves out of this envelope of acceptable behavior. A more sophisticated idea to detect shifts in the data is using a CUmulative SUM (CUSUM) method [Page, 1954, Montgomery, 1996]. As the name suggests, CUSUM maintains a cumulative sum of deviations from a reference value r. Let us consider a time series where at time i we have measurement X_i. The one-sided CUSUM calculation is as follows:

$$C_0 = 0$$

$$C_m = \max(0, X_m - (\mu_0 + K) + C_{m-1}) \tag{1.1}$$

μ_0 is the expected value. From the equations above, if the X_m values are close to the mean, then the C_m values will be some small value. However once a positive shift from the mean occurs, the C_m value will increase rapidly. K is known as the slack value or allowance. In equation 1.1, any values within K units of μ_0 will be effectively ignored. It also causes C_m to drift towards zero during normal behavior of the system. Alerts are raised whenever C_m exceeds a threshold decision interval H, and C_m is reset to zero.

We have investigated a POMDP based approach to optimal alarming in a decision theoretic framework in [Das et al., 2004a]. We propose a probabilistic model of the process being monitored and the detection algorithm observing it. Based on those models we can determine the correct belief state for the underlying process and the optimal decision when considering the costs of signaling an alarm and allowing an event to go undetected.

For a more realistic modeling, the data observed thus far can be used to predict future values. If there is a significant discrepancy between the predicted and observed values, it can be denoted as surprising. The most common technique is to model the time series as an Auto-Regressive Moving Average (ARMA) or Auto-Regressive Integrated Moving Average (ARIMA) process [Box et al., 1994]. A summary of this and other time series monitoring techniques that are commonly used in biosurveillance can be found in [Moore et al., 2002].

Another area of interest is the case of categorical and discrete time series variables. Researchers have used subsequence matching algorithms [Keogh et al., 2002b, Patel et al., 2002] to detect anomalies in such cases. Yang *et al.* [Yang et al., 2001] use an information

gain metric to locate surprising periodic patterns. Keogh *et al.* develop an algorithm TARZAN, that can efficiently compute the expected frequency of a pattern using suffix trees [Keogh et al., 2002a]. They use this idea to detect surprising patterns in the dataset. There has also been some work on novelty detection in time series using neural networks [Whitehead and Hoyt, 1993, Borisyuk et al., 2000].

In §6.2 we further discuss other multivariate methods that can be used for time-series anomaly detection.

1.3.2 Spatial Anomaly Detection

Along with temporal analysis, spatial analysis of data is another important and much applied area of research. In particular, the detection of spatial clusters or 'bumps' in spatial data has numerous applications in epidemiology, bio-surveillance, astronomy and other fields. Among the wide range of methods proposed to test for spatial clustering, the spatial scan statistic is a common approach.

Consider the plot in Figure 1.1. Each point shows the home location of a patient arriving in the emergency department[2]. The crosses mark points with a particular symptom of interest such as respiratory problems. We are interested in determining whether there is some region within this data (such as the circle shown in the plot) that has a higher incidence rate of the symptom of interest. This is a typical spatial scan statistic application. Studies of this sort are common in the field of public health and are used to determine whether environmental factors are causing higher disease rates in certain areas. In our case, we are interested in early detection of a bio-terrorist attack, which under several delivery mechanisms including airborne anthrax release, may be clustered spatially.

The goal is to detect whether some region has a higher incidence rate. Theoretically we can imagine a region to have any arbitrary shape and spread. However in a realistic scenario, we are interested in a regions that are geographically compact, i.e. include a set of locations that are all near to each other. In practice different regular shapes such as

[2]This data comes from emergency departments in the Pittsburgh area. The data has been anonymized and the locations have significant noise added for further privacy protection.

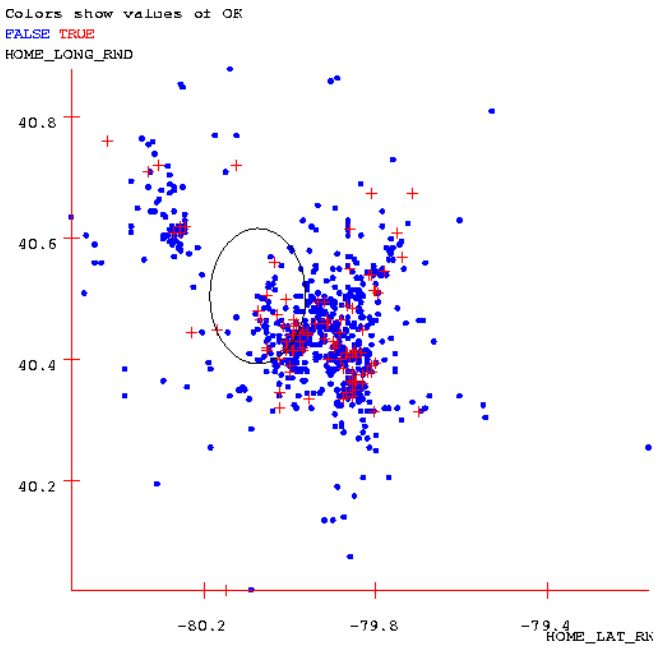


Figure 1.1: Sample scan statistic application.

circles, squares or rectangles have been used to define regions. The original formulation [Kulldorff, 1997] scans all possible circular regions to find the region with maximum discrepancy. The algorithm for computing the scan statistic is as follows (adapted from [Glaz and Balakrishnan, 1999, Kulldorff, 1997, Neill et al., 2005c]):

Given a region S (a set of locations), a score $F(S)$ is defined that indicates the degree of discrepancy for that region. The most common statistical framework for the spatial scan is a frequentist, hypothesis testing approach. In this approach, $F(S)$ is defined as a likelihood ratio score:

$$F(S) = \frac{\Pr(Data|H_1(S))}{\Pr(Data|H_0)} \tag{1.2}$$

Here H_0 denotes the null hypothesis that there are no clusters and $H_1(S)$ denotes the alternative hypothesis assuming a cluster in region S.

In general, the hypotheses might have some parameter space, and in the maximum likelihood framework, the estimates of the parameters that maximize the likelihood of the

data are used Neill et al. [2005c]:

$$F(S) = \frac{\max_{\theta_1(S) \in \Theta_1(S)} \Pr(Data|H_1(S), \theta_1(S))}{\max_{\theta_0 \in \Theta_0} \Pr(Data|H_0, \theta_0)} \tag{1.3}$$

where, $\Theta_1(S)$ denote the parameter space for the alternate hypothesis $H_1(S)$ and Θ_0 denote the parameter space for the null hypothesis H_0. In the simple case of assuming that the marks in Figure 1.1 are Bernoulli random variables, the likelihood of the data given the null hypothesis is as follows:

$$\Pr(Data|H_0) = \left(\frac{N^+}{N}\right)^{N^+} * \left(\frac{N^-}{N}\right)^{N^-} \tag{1.4}$$

where N is the total number of data points and N^+ and N^- are the number of positive and negative instances respectively.

For the alternate hypothesis, the likelihood of the data using the maximum likelihood estimates of the distribution parameters is:

$$\begin{aligned} \Pr(Data|H_1(S)) \quad = \quad & \left(\frac{N^+(S)}{N(S)}\right)^{N^+(S)} * \left(\frac{N^-(S)}{N(S)}\right)^{N^-(S)} * \\ & \left(\frac{N^+(\overline{S})}{N(\overline{S})}\right)^{N^+(\overline{S})} * \left(\frac{N^-(\overline{S})}{N(\overline{S})}\right)^{N^-(\overline{S})} \end{aligned} \tag{1.5}$$

where the S and $\overline{S}$ in parentheses indicate the respective counts of points inside and outside the region S.

We then compute the score $F(S)$ over all possible regions and we report the score $F(S^*)$ and the region S^* that yielded this maximum score. This approach of searching over all possible regions is computationally very expensive. Various approaches have been suggested to make it tractable. These include grid based multi-resolution branch and bound search [Neill and Moore, 2004, Neill et al., 2005c], approximation algorithms [Agrawal and Srikant, 1994] and greedy strategies [Friedman and Fisher, 1999]. We can also calculate the p-value of each detected region by randomization testing. A set of N_{random} replica data sets under the null hypothesis (no clustering) is created by randomly shuffling the data

labels. The spatial scan algorithm is run on all the replicas, and the maximum region score is recorded for each run. The p-value of a region under the original scan is computed by comparing it to the distribution of maximum region scores under the null hypothesis.

Alternatively it is possible to use a Bayesian approach where the test statistic $F(S)$ is defined as the posterior probability of a cluster in the region, marginalized over the model parameters:

$$F(S) = \text{Pr}(Data \mid H_1(S)) = \frac{\text{Pr}(Data|H_1(S))\text{Pr}(H_1(S))}{\text{Pr}(Data|H_0)\text{Pr}(H_0) + \sum_{S'} \text{Pr}(Data|H_1(S'))\text{Pr}(H_1(S'))} \tag{1.6}$$

1.3.3 Network Intrusion Detection

Anomaly detection applied to network intrusion detection has been an active area of research since it was proposed in [Denning, 1980]. Traditional anomaly detection approaches builds models of normal data and detect deviations from the normal model in observed data. A survey of these techniques is given in [Warrender et al., 1999]. One approach is to use sequence analysis to determine anomalies. A method of modeling normal sequences using look ahead pairs and contiguous sequences is presented in [Hofmeyr et al., 1998], and a statistical method to determine frequent sequences in intrusion data is presented in [Helman and Bhangoo, 1997]. [Lee and Stolfo, 1998] uses a decision tree model over normal data and [Ghosh and Schwartzbard, 1999] uses neural network to obtain the model. [Eskin, 2000] uses a probability distribution model from the training data to determine anomalous data. They use a mixture model to explain the presence of anomalies. A clustering based approach to detect anomalies in the dataset is used in [Leung and Leckie, 2005] and [Eskin et al., 2002]. One-class SVMs [Li et al., 2003, Heller et al., 2003] and Genetic Algorithms [Shon et al., 2005] have also been used to classify anomalies in this context.

1.3.4 Association Rule Based Approaches

The task of association rule mining has received considerable attention especially, in the case of market basket analysis [Agrawal et al., 1993]. An association rule is an expression of the form $X \Rightarrow Y$, where X and Y are sets of items. Given a database of records (or transactions) D, where each record $T \in D$ is a set of items, $X \Rightarrow Y$ expresses that whenever a record T contains X, then T probably also contains Y. The confidence of the rule is the probability $p(Y|X)$. The support of the rule is the number of training cases where both X and Y are present. Typically we search for rules with both high confidence and high support. Instead of sets of items, X and Y can also be considered to be the events that an attribute of T takes some particular values.

Association rule mining is commonly used in the analysis of market-basket data, where the target of mining is not predetermined. Chan *et al.* [Chan et al., 2006] developed a rule learning method, LERAD, to detect anomalies. They consider rules of the form $X \Rightarrow Y$, where X and Y are mutually exclusive subsets of attributes taking on particular values. They seek combinations of X and Y with large values of $P(Y|X)$. The anomaly score of a record depends on $P(\neg Y|X)$, where Y, though expected, is not observed when X is observed.

Balderas *et al.* [Balderas et al., 2005] mine hidden association rules, or rules that are not common, but have high confidence. Such rules are assumed to represent the rare anomaly class.

WSARE developed by Wong *et al.* [Wong et al., 2002] also uses rules to identify anomalies. But in this case, all possible one and two component rules are evaluated by comparing the events on the current day against events in the past. It is described in more detail in §3.2.2.

1.3.5 Likelihood Based Approaches

In these approaches, 'normal' data is modeled as a probability distribution. Any test record that has an unusually low likelihood based on the probability model is flagged as anoma-

lous. For multivariate categorical data, dependency trees and Bayesian networks are common representations of a probability density model. Dependency trees have been used to detect anomalies in [Pelleg, 2004]. We choose the Bayesian network as the standard model against which we compare most of our algorithms. Hence, we give an overview of this method next.

Anomaly Detection Using Bayesian Network

The Bayesian networks are popular representations of probability models over attributes for categorical data because of its parsimonious use of parameters, and efficient learning and inference techniques. Bayes Net have been used for detecting anomalies in network intrusion detection [Bronstein et al., 2001, Ye and Xu, 2000], detecting malicious emails [Dong-Her et al., 2004] and disease outbreak detection [Wong et al., 2003].

[Cooper et al., 2006] and [Cooper et al., 2004] use Bayesian network models to detect anomalies in Emergency Department data. The doctoral thesis [Jiang, 2008] investigates the use of spatio temporal information within the Bayesian network framework.

A typical anomaly detection approach is to learn the structure and parameters of a Bayesian network using the training data, compute the likelihood of each record in the test dataset given the Bayesian network model, and report test records with unusually low likelihoods as potential anomalies. Any good structure and parameter learning algorithm is appropriate to learn the model. For our experiments, we used the optimal reinsertion algorithm [Moore and Wong, 2003] to learn the structure, and then did a maximum likelihood estimation of the network parameters. Once the model is built, to test any record we find its complete record likelihood given the probability model. Test records that have unusually low likelihood are then flagged as anomalies. In general, the log-likelihood value can be used as the anomalousness score of each record.

1.4 Datasets Used

In this section we describe the various real-world datasets that we have used in this thesis to evaluate our anomaly detection method. All the datasets correspond to records of observations of some system. Most of these datasets have categorical (or symbolic) valued attribute types. This motivates our initial focus on categorical valued datasets. Also, none of these have known labels of real life anomalies. Some form of simulation is used in each case to generate anomalies that would be relevant to a domain user, and inject them into the dataset, and these simulated anomalies are used for evaluation and comparison of methods.

1.4.1 PIERS Dataset

Attribute Name	Arity
Country	22
Foreign Port	42
US Port	16
Shipping Line	4
Shipper Name	4218
Importer Name	6412
Commodity Description	1649
Size (discretized)	5
Weight (discretized)	5
Value (discretized)	5

Table 1.2: Features in PIERS Dataset

Our first dataset consists of records describing containers imported into the country. This data was obtained from the Port Import Export Reporting Service (PIERS). Each record consists of 10 attributes, describing the container, its contents, and its transport

as outlined in Table 1.2. Most of the attributes are categorical, such as the country of origin, the departing and arriving ports and shipping line. There are three real valued attributes, the size, weight and value of the container. We have categorized these to five discrete levels. Our work is motivated by the need to detect unusual shipments among all imports into the country. Specifically, we are interested in detection of illegal activities like smuggling. However, we do not have any labels in the data, i.e. there are no known cases of smuggling or unusual activity. Hence, in all our evaluations, we generate some form of synthetic anomalies, as described in each chapter.

1.4.2 KDD Cup 99 Network Connections Dataset

The network connection records dataset from KDD Cup 1999 [KDDCup, 1999] contains a wide variety of intrusions simulated in a military network environment. Each record is a vector of extracted feature values from a connection record obtained from the raw network data. The extracted features include the basic features of an individual TCP connection such as its duration, protocol type, number of bytes transferred etc. Other features of an individual connection are obtained using some domain knowledge, and includes the number of file creation operations, number of failed login attempts, whether root shell was obtained, and others. Finally there are a number of features computed using a two second time window. These includes the number of connections to the same host as the current connection, the number of connections to the same service, etc. In total there are 41 features, most of them taking continuous values. The continuous features are discretized to 5 levels.

There are a total of 24 types of attack. Some of the attacks such as denial of service or probing attacks are much easier to detect than other attacks. We have selected the most common kinds of attacks for our evaluations: apache2, guess password, mailbomb, neptune, smurf, snmpguess snmpgetattack and warezmaster. In this dataset, the attacks are labeled, but we do not use a supervised approach. Instead, we train on data with no attacks present and then test our ability to recognize these attacks as anomalies in a test set.

1.4.3 Sales of Over the Counter (OTC) medicines data

This data consists of Over the Counter medicine sales in US pharmacies for a period of 2 years (2004-2006). Each sale has the following information:

1. Date of Sale
2. Zipcode specifying the location of sale
3. Category: There are five categories of sales: Baby/Child Electrolytes, Cough/Cold, Internal Analgesics, Stomach Remedies and Thermometers.
4. Promotion: Whether the sale was part of a promotion offer.
5. Count: The number of sales matching the above attributes.

We used this information to compute the aggregate sales for each category in each zipcode on each day, and our goal is to detect spatial regions (clusters of zipcodes) with anomalous values of recent counts. These clusters may represent disease outbreaks or other unusual patterns in the dataset, such as inclement weather. Since the data has no labeled disease events, we will test using surrogate events such as hurricanes, for which we can easily obtain our own labels.

1.4.4 Emergency Department Dataset

This dataset consists of Emergency Department records from hospitals around the country. The data spans 400 days.

It has the following attributes:

1. Admit Date: Date on which the patient was admitted.
2. Prodrome: The main category of the patient's complaint upon arrival at the emergency department. It can have 7 possible values.

3. Age Decile: 1, 2, . . . 9.

4. Gender

5. Zipcode: This specifies the location where the patient was admitted.

6. Count: The number of cases matching the above attributes.

While there are no known real disease outbreaks in this data, we have simulated cases of airborne anthrax release produced by a state-of-the-art simulator [Hogan et al., 2007], and measured each method's timeliness and accuracy of detection as a function of the false positive rate.

Chapter 2

Detecting Anomalous Records in Categorical Datasets

We consider the problem of detecting anomalies in high arity categorical datasets. Quite often we have access to unlabelled data which consists mostly of normal records, along with a small percentage of anomalous records. We are interested in the problem of *unsupervised anomaly detection*, where we use the unlabelled data for training, and detect test records that do not follow the definition of normality. In this chapter, we focus on the problem of detecting single record anomalies, testing for each record independently.

A standard approach is to create a model of normal data, and compare test records against it. A probabilistic approach builds a likelihood model from the training data. Records are tested for anomalousness based on the complete record likelihood given the probability model. For categorical attributes, Bayesian networks give a standard representation of the likelihood. While this approach is good at finding outliers in the dataset, it often tends to detect records with attribute values that are rare. Sometimes, just detecting rare values of an attribute is not desired and such outliers are not considered as anomalies in that context. We present an alternative definition of anomalies, and propose an approach of comparing against marginal distributions of attribute subsets. We show that this is a more meaningful way of detecting anomalies, and has a better performance on

detecting semi-synthetic anomalies injected into real world datasets.

We also compare our methods against a association rule based method LERAD [Chan et al., 2006]. It considers rules of the form $X \Rightarrow Y$, where X and Y are mutually exclusive subsets of attributes taking on particular values. They seek combinations of X and Y with large values of $P(Y|X)$. The anomaly score of a record depends on $P(\neg Y|X)$, where Y, though expected, is not observed when X is observed. The main disadvantage of this method is that it learns a very small subset of all the possible rules. Various other rule based methods used to detect anomalies are discussed in §1.3.4.

First, we present the problem statement, along with our algorithms for anomaly detection. We show ways of speeding up the computation and making it more memory efficient. This is followed by the experimental setup, where we describe the datasets used, and the evaluation procedure. The results of our algorithms on the datasets are presented next. We conclude with a discussion of possible extensions of the current work. Parts of this chapter have been adapted from our paper in KDD 2007 [Das and Schneider, 2007].

2.1 Approach

Suppose we are given a set of records comprised of several attributes. The data contains both normal and anomalous records. However, we do not have any labeling of the data. The problem is to identify the anomalous records among them. First we need to define *normality* with respect to the given data. Here, we make an assumption that in the training data a majority of records are normal and there are only a few anomalous records. This means we can build a model of all the data with minimal harm caused by the anomalous records. We discuss several ways of approaching this problem in the following sections.

Training

1. Construct a *conditional AD Tree* over the training dataset (§2.1.3).
2. Determine the dependence between all attribute sets up to size *k* by computing the mutual information between them (§2.1.3).
3. Construct the cache for denominator counts (§2.1.4).

Testing: Scoring a test record t

1. For each *mutually exclusive* and *dependent* pair of attribute sets A and B (§2.1.3):
   (a) Compute $r(a_t, b_t)$ (§2.1.3).
2. Compute the overall *conditional* score of the record t from all the r-values calculated above (§2.1.3).

Figure 2.1: Conditional Anomaly Test Algorithm.

2.1.1 Baseline Approach of using Bayesian Networks

We have used the method of using Bayesian Networks (described in §1.3.5) to detect individual record anomalies as the baseline method. We learn a Bayesian network probability model from the training dataset, and to test any record we compute its complete record likelihood given the Bayes net. The record log-likelihood is used as the anomalousness score. Given a threshold, test records that have a lower log-likelihood value are then flagged as anomalies.

2.1.2 Conditional and Marginal Methods

Figures 2.1 and 2.2 give an overview of the two proposed algorithms used to test for anomalous records. We will explain the steps in detail in the following sections.

Training

1. Construct a *marginal AD Tree* over the dataset (§2.1.5).
2. Compute the marginal count histograms over the training data (§2.1.5).

Testing: Scoring a test record t

1. For each attribute set A with up to k attributes:
   (a) Compute $qval(a_t)$ (§2.1.5).
2. Compute the overall *marginal* score of the record t as the minimum q-value calculated above (§2.1.5).

Figure 2.2: Marginal Anomaly Test Algorithm.

Our current work is motivated by the need to detect unusual shipments among all imports into the country. Each record corresponds to a container that is being imported. It has attributes describing the container, its contents, and its transport as outlined in Table 1.4.1.

2.1.3 Conditional Probability Tests

We will motivate our method based on the disadvantage of using the likelihood based approach to detect anomalies in this context. Consider the attribute *ShipperName*, which has a very high arity of more than 4000. In this case, as in many real world problems, the distribution of values of high arity attributes is very skewed. Some of the values are common, while a large number of them are very rare. When we construct a probability distribution of the data, these rare attribute values contribute to a skewed distribution. If a record has *ShipperName* as one of the rare values, then the record's likelihood is dominated by this term. This means that rare values will cause these records to look very unusual. But often, an attribute having a rare value might not be useful information. In our data,

more than 20% of the instances contain a value of *ShipperName* that occurs only once in the training data. Another disadvantage of using the likelihood based method is relates to the fact that we learn a fixed probability structure during the training phase. Since most structure learning algorithms use approximate methods, there can be mistakes in the learned structure. The testing phase depends only on the learned probability model, and any model learning mistake is going to persist throughout the testing phase. To overcome this, our proposed method does not learn an overall probability model, but instead directly uses the counts from the training data during the testing phase.

Consider a particular test record t and the attributes *ShipperName* and *Country*. We define $P(SN_t, C_t)$ = *P(Shipper- Name* =SN_t, *Country*=C_t*)*, where SN_t and C_t are the *ShipperName* and *Country* of the test record t respectively. In general, let A be a set of attributes. Define $P(a_t)$ = $P(A = a_t)$, where a_t is the corresponding set of values of A in the test record t.

We are interested in detecting unusual combinations of attribute values. For example, say $ShipperName = SN_1$ always occurs with *Country*=C_1 and never with *Country*=C_2. Then a record t having *ShipperName*=SN_1 and *Country*=C_2 is considered unusual or anomalous. This corresponds to the probability $P(SN_1, C_2)$. But we have to be careful in interpreting this. Consider a situation where *Country*=C_2 occurs very rarely in the data. In this case, the fact that *ShipperName*=SN_1 has never occurred with *Country*=C_2 can be explained by the rarity of seeing records from *Country*=C_2. It might not mean that for shipments coming from *Country*=C_2, it is unusual to see *ShipperName*=SN_1. Here, we do not have enough data to support the hypothesis that this is really anomalous. To take care of this fact, we can normalize the joint probability of these attributes with the marginal probability $P(Country_t)$. Now, if $P(Country_t)$ has a low value, the ratio $\frac{P(ShipperName_t, Country_t)}{P(Country_t)}$ will no longer be small. But, the same argument applies to the attribute *ShipperName*, and hence, we also normalize with respect to $P(ShipperName_t)$. The quantity we now consider is the ratio $\frac{P(ShipperName_t, Country_t)}{P(ShipperName_t)P(Country_t)}$.

In general, we consider the ratio $r(a_t, b_t) = \frac{P(a_t, b_t)}{P(a_t)P(b_t)}$ for attributes *A* and *B*. An unusually low value of this ratio suggests a strong negative dependence between the occurrences of a_t and b_t in the training data. When we observe them together in the test record t, we

can reasonably say that it is anomalous. This also ensures we have seen enough cases of a_t and b_t in the training data to support the hypothesis of negative dependence. We quantify this notion of *minimum support* in §2.1.4.

To generalize this idea to more than two attributes, we can consider attribute sets instead of single attributes. For example, we can consider whether the combination of attribute set *A* = {*ShipperName, Weight*} and the attribute set *B* = {*Country, Commodity*} is unusual. The ratio that we consider here is:

$$r(a_t, b_t) = \frac{P(a_t, b_t)}{P(a_t)P(b_t)}$$
$$= \frac{P(ShipperName_t, Weight_t, Country_t, Commodity_t)}{P(ShipperName_t, Weight_t)P(Country_t, Commodity_t)}$$

Similarly, we can compare any two subsets of attributes, the only constraint being that there should be no common attribute among them. Let us call this ratio the r-value of the record t for the attribute sets A and B. Considering all possible subsets would require computation time exponential in the number of attributes. Therefore, we only consider subsets up to size k. Also, we want to avoid comparing attribute sets that are completely independent. We compute the mutual information $\mu(A, B)$ between two attribute sets A and B, and calculate $r(a_t, b_t)$ only if the mutual information is greater than a threshold. We define A and B to be *dependent* if,

$$\mu(A, B) \geq \beta_\mu \tag{2.1}$$

where, β_μ is a threshold parameter, set to a low value of 0.1 (empirically) in our experiments.

Thus, for a given record, we consider all pairs of *dependent* and *mutually exclusive* subsets having up to k attributes, and calculate the corresponding r-values.

A ratio of the form r = $\frac{P(A,B)}{P(A)P(B)}$ has been proposed as a measure of *suspicious coincidence* by Barlow [Barlow, 1989]. It states that two candidate fragments A and B should be combined into a composite object AB if the probability of their joint appearance $P(A, B)$ is much higher than the probability expected in case of statistical independence $P(A)P(B)$. It has also been used to investigate unsupervised learning of complex visual

stimuli by human subjects [Edelman et al., 2002]. In association rule mining, this quantity is known as *interest* [Brin et al., 1997] or *lift* [Sheikh et al., 2004]. In most of these cases, large values of r are interesting as it signifies a suspicious coincidence of the events co-occurring. We are interested in exactly the opposite situation, where low r values signify that the events do *not* co-occur naturally. If they are observed together, then we treat it as an anomaly.

Partitioning the training data

A further generalization is to use a ratio of the form: $rval(a_t, b_t|c_t) = \frac{P(a_t,b_t|c_t)}{P(a_t|c_t)P(b_t|c_t)}$, where A, B and C are mutually exclusive subsets of attributes with at most k elements. This ratio is similar to the previous formula, but here we consider the probabilities conditioned on a set of attributes. It is equivalent to partitioning the training data and considering only a subset to estimate the probabilities, consisting only of records that match the test record t in a subset of attributes, C.

Combining evidence across different attribute sets

One disadvantage of our method is that it considers only a subset of attributes at a time. The final score of a record is the minimum score obtained over all such subsets. But, the score reflects the behavior of only a particular subset of size up to $2k$, ignoring the values of other attributes. Here, we make an assumption that maximum of $2k$ attribute values indicate anomalous behavior. In many practical problems this assumption is reasonable.

But, as shown in the results using artificial anomalies, when the number of anomalous attributes is larger than $2k$, comparing against a joint distribution might give more accurate results.

To solve this problem, we can combine the evidence across different attribute sets. We use the following heuristic to score the record t:

1. Order the r-values in ascending order. Consider only the ordered values r_1 to r_q which are less than a threshold α (described in the next section).

2. Initialize: Score = 1, and $U = \phi$.

3. For i = 1 to q

   (a) If there is any common attribute between the attributes defining r_i and U, then skip to the next value of i.

   (b) Else, $Score = Score * r_i$, and include the attributes defining r_i in U.

This heuristic computes the product of the selected r-values corresponding to mutually exclusive sets of attributes. The intuition is that if the attribute subsets were not only disjoint, but also independent, then this would be the r-value for the larger combined set of attributes.

$$\begin{aligned} r(a_t, b_t) \times r(c_t, d_t) &= \frac{P(a_t, b_t)P(c_t, d_t)}{P(a_t)P(b_t)P(c_t)P(d_t)} \\ &= \frac{P(a_t, b_t, c_t, d_t)}{P(a_t, c_t)P(b_t, d_t)} \\ &= r([a_t, c_t], [b_t, d_t]) \end{aligned} \tag{2.2}$$

Here, we assume $(A \perp C)$ and $(B \perp D)$. In general, this assumption does not hold, but the heuristic gives a reasonable strategy to combine evidence from multiple r-values.

User specified pruning of the search space

In many applications we can use domain information to restrict our search space. For example, consider the attributes *Country* and *City*. Given the value of *City*, the value of *Country* is fixed. We do not need to test if there is a rare combination of these two attributes. In general, if there is a hierarchical structure of the attributes, we do not want to compare between the higher and lower level attributes. One exception is the case of searching for data entry errors, which is another potential application of our algorithm.

A user may simply be uninterested in some combinations of attributes. For example, a medical diagnosis tool may not care about an anomalous combination of patient demographic features. It may only be interested in anomalous sets of symptoms or symptoms in combination with demographics.

In either case, our algorithms can easily ignore special combinations of attributes. This improves computational speed by reducing the search space, and will produce results that are more meaningful to the end user.

Estimating the probability values

For calculating the r-value $r(a_t, b_t)$ of a test record t, we need to estimate the marginal probability values from the training data. The MLE estimate is $P(a_t) = \frac{C(a_t)}{N}$, where $C(a_t)$ is the count of training cases where $A = a_t$. N is the total number of training records. A problem with this estimator is that when $C(a_t, b_t) = 0$, then $r(a_t, b_t) = 0$. Regardless of the threshold α, all such cases will be flagged as anomalies.

To avoid this problem, we calculate the expected value of $p_A = P(a_t)$ with a Bayesian prior. Given the record t, each attribute behaves as binary. The attribute set A can have two possible values a_t and 'not a_t'.

$$P(Data|p_A) = Binomial(N, p_A) \tag{2.3}$$

$$P(p_A|Data) = \frac{P(Data|p_A) * P(p_A)}{P(Data)} \tag{2.4}$$

$$P(p_A|Data) \sim p_A^{C(a_t)}(1 - p_A)^{N - C(a_t)} \tag{2.5}$$

$$P(p_A|Data) \sim Beta(C(a_t) + 1, N - C(a_t) + 1) \tag{2.6}$$

Here we assume an uniform prior over p_A. Hence $E[p_A] = \frac{C(a_t)+1}{N+2}$.

Bound on the counts

From eqn. 2.6 above, we can calculate:

$r(a_t, b_t) = \frac{P(a_t,b_t)}{P(a_t)P(b_t)} = \frac{C(a_t,b_t)+1}{N+2} \times \frac{N+2}{C(a_t)+1} \times \frac{N+2}{C(b_t)+1}$.

To compute this ratio we need the counts $C(a_t)$, $C(b_t)$ and $C(a_t, b_t)$. We use a caching technique to cache these counts as described in §2.1.4. To make this caching tractable, we compute a lower bound for $C(a_t)$ and $C(b_t)$.

The record t is interesting when $r(a_t, b_t) \leq \alpha$.

$$
\begin{aligned}
&\Longrightarrow \frac{C(a_t, b_t) + 1}{N + 2} \times \frac{N + 2}{C(a_t) + 1} \times \frac{N + 2}{C(b_t) + 1} \leq \alpha \\
&\Longrightarrow \frac{C(a_t, b_t) + 1}{N + 2} \times \frac{N + 2}{C(a_t) + 1} < \alpha \\
&\quad \text{[because, } (N + 2) > (C(b_t) + 1)\text{]} \\
&\Longrightarrow \frac{C(a_t, b_t) + 1}{C(a_t) + 1} < \alpha \\
&\Longrightarrow \frac{1}{C(a_t) + 1} < \alpha \\
&\quad \text{[because, } C(a_t, b_t) \geq 0\text{]} \\
&\Longrightarrow C(a_t) > \frac{1}{\alpha} - 1
\end{aligned} \tag{2.7}
$$

Similarly, $C(b_t) > \frac{1}{\alpha} - 1$. Hence, we need to consider only the cases where $C(a_t)$ and $C(b_t)$ are greater than this bound.

Using AD Trees for computing counts

The required counts are conjunctive counting queries on the dataset, and can be efficiently queried using an AD Tree [Moore and Lee, 1998]. The AD Tree building algorithm scans the dataset once, and precomputes information needed to answer every possible query in time independent of the number of records. The parameter *leaflist size* can be adjusted to obtain a tradeoff between the memory used and the query response time. Note that for our algorithm, we will never need an AD Tree of depth greater than $2k$.

2.1.4 Computational Speedup

Reducing arity

The memory required to build an AD Tree significantly depends on the arity of the attributes. We use the result from eqn. 2.7 to reduce the arity of each attribute. Consider an attribute value l_t of attribute L in test record t. Let A and B be two attribute sets, such that $L \in A$ (or equivalently it could belong to B), and we want to calculate the value of $r(a_t, b_t)$. The r-value will be of interest only when $C(a_t) > \frac{1}{\alpha} - 1$ and $C(b_t) > \frac{1}{\alpha} - 1$. Since $L \in A$, $C(l_t) \geq C(a_t)$. This implies $C(l_t) > \frac{1}{\alpha} - 1$. So we can ignore all values l_i of L where $C(l_i) < \frac{1}{\alpha} - 1$. All such values are called rare values of attribute L. All other values are called common values of attribute L. Any r-value that includes the attribute L corresponding to a rare value, will always be greater than α. So, we can replace all rare values by a generic rare value. While computing the r-value of attribute sets A and B we skip the computation if either a_t or b_t contains any rare value. We can ignore missing values originally present in the dataset in a similar fashion. This scheme of keeping only the common values significantly reduces the arity of each attribute and drastically reduces the memory required to build the AD Tree. This also ensures that if any ratio $r(a_t, b_t)$ is anomalous, then there is a *minimum support* of $\frac{1}{\alpha}$ training cases corresponding to the attribute values a_t and b_t.

Caching values

Even though the AD Tree structure retrieves the counts quite efficiently, it has some overhead because it tries to store the results for all possible queries, whereas, we are interested only in some special cases as described below. We can improve the query response time by building an additional cache that is more specialized for the task. We build an AD Tree as the base query module. We then build a more specialized cache as described below, by obtaining the relevant counts from the AD Tree. This caching scheme gives 1.5 to 2 times speedup in computation.

Caching the Denominator values: Let there be M attributes in the dataset, numbered from 1 to M. There are $S = \binom{M}{1} + \binom{M}{2} + ... + \binom{M}{k}$ attribute combinations, considering up to k attributes in each combination. We call these S composite attributes. We create a tree data structure where each node represents a composite attribute, i.e., a set of attributes. The root node represents the null set. It has M children, each representing the unary set of the corresponding attribute. Let q be the highest attribute number in the set represented by node n. Then n has M-q children, child i corresponding to the union of the set represented by n, and attribute number q+i. We limit the depth of the tree to k. The complete tree has S+1 nodes, corresponding to each composite attribute and the null set.

Now, for each composite attribute, we find the common values (§2.1.4) present in the dataset. We store the count of the number of occurrences for each common value of each composite attribute in the corresponding node. As noted above, the counts $C(a_t)$ and $C(b_t)$ are needed only when they are greater than $\frac{1}{\alpha} - 1$ (i.e., when they are common). Hence all the counts that we need to compute the denominator of any r-value, are precomputed in our cache. It takes O(k) time to retrieve any count stored in the cache.

Caching the Numerator values: Unlike the denominator counts, the numerator counts can correspond to rare value combinations (i.e., $C(a_t, b_t)$ can be as small as zero). It becomes infeasible to store counts for all possible combinations of values for all attributes (as a caching scheme, it is actually equivalent to the full blown AD Tree, which does the job more efficiently). However, given a test record t, it is possible to cache the corresponding counts for all attribute combinations, as each combination now represents a fixed set of values. We see that we can reuse the computation of probability values $P(a_t, b_t)$. For example, we compute $P(Country_t, Shipper_t, ForeignPort_t, Weight_t)$ when *A*={*Country, Shipper*} and *B*={*Foreign Port, Weight*}. We have the same value for *A*={*Country, Shipper, Foreign Port*} and *B*={*Weight*}. Therefore, each time before computing the value of $P(a_t, b_t)$, we first check if it has been already calculated. If not, we compute its value, obtaining relevant counts from the AD Tree. We then cache this value in our tree cache structure for future use. This reduces the number of (relatively) expensive AD Tree queries.

Note that the cached values are useful only for a particular test record. For a new test record we clear the cache and start over.

2.1.5 Marginal Probability Tests

While computing the r-value, we normalize with respect to the marginal probabilities. This means that an unusually low marginal probability value will not be detected by this method. That is fine because we want to detect unusual pairings of sets of attributes, rather than just detecting a rare combination. But in some cases, detecting rare combinations might also be useful.

We define $qval(a_t)$, the q-value of an attribute set A for the test record t as the sum of $P(A = a_t)$ and all values of $P(A)$ that are smaller or equal to $P(A = a_t)$. Here a_t is the corresponding set of values of the attributes in A in the test record t.

$$qval(a_t) = \sum_{x \in X} P(x) \text{ where, } X \equiv \{x : P(x) \leq P(a_t)\} \tag{2.8}$$

This is parallel to the standard definition of p-value for continuous variables, which sums over values that are more extreme than the current value. In our definition for the case of categorical attributes, *more extreme* corresponds to values that have a probability less than the current value[1].

The q-value of an attribute gives an indication of rarity of its occurrence. An attribute set A is considered anomalous in record t if $qval(a_t) \leq \alpha_m$, where α_m is a predetermined threshold. The advantage of using this measure is seen when there are a lot of rare values of some attribute. For example, in the container shipment data, the attribute $ShipperName$ has a very high arity, and a lot of the values are rare. But, in this situation, the $qval$ of a rare value is computed by aggregating over all similar rare values, and will not have a small value. This avoids the problem of detecting each occurrence of a rare value as an anomaly.

[1]We have heard equally vigorous arguments that this is exactly a p-value and that it is not. We refrain from taking a position in this debate

Implementation

Computing the $qval(a_t)$ of an attribute set A in test record t is somewhat more complicated than calculating the r-value. To calculate $qval(a_t)$, we not only need to know $C(a_t)$, but also the counts for all other possible values a_i of A such that, $C(a_i) \leq C(a_t)$. When dealing with composite attributes, the number of possible values it can have becomes exponentially large. Even if all the counts are cached, going through each of them for every test becomes prohibitive.

Instead, for every composite attribute A, we store the histogram h of the number of times different values occur in the training dataset. For example we precompute the fact that A has $h(1)$ values occurring only once, $h(2)$ values occurring twice and in general, $h(i)$ values occurring i times. When testing attribute set A in record t, we compute $C(a_t)$, and compare that to the precomputed histogram. We compute the quantity $C_{rarer} = \sum_{i \leq C(a_t)} i * h(i)$. Normalizing with respect to the number of data-points N, gives the desired $qval(a_t)$. We still need to get the count $C(a_t)$, and unlike the conditional method, we are especially interested in rare values. Hence, we cannot reuse the AD Tree constructed for the conditional method. We construct another AD Tree without any reduction of arity from the original dataset. We call this the marginal AD Tree. We use a bigger leaf-list size to keep the size of the tree manageable [Moore and Lee, 1998].

Note that all the information in the conditional AD Tree is also contained in the marginal AD Tree. But, we still maintain the conditional AD Tree separately as it is faster to query from the smaller tree for the conditional method.

2.2 Experimental Setup

2.2.1 Datasets

PIERS Dataset

This dataset (described in §1.4.1) consists of records of containers imported into US. Since there were no labeled anomalies in the original data, we create synthetic anomalies by randomly flipping attribute values. We first partition the dataset into training and testing sets. We randomly choose 10% of the data as a test set, and the remaining 90% is the training set. The dataset used for generating these results has 100,000 records so the training set has 90,000 records and the test set has 10,000. We modify a random 10% (i.e. 1000) of the test set records to be anomalies. For each record that is modified, a random set of up to l attributes is chosen. The values for these attributes are reassigned by drawing from the corresponding attribute marginal distribution. Higher the value of l, greater the degree of anomaly.

Apart from randomly flipping attribute values, we use another method to create anomalies in the test data. The training data is from the month of June 2002. We randomly pick 1000 records from a different month (June 2003), and replace 1000 randomly chosen records in the test set. We deliberately do not include records from June 2003 that have attribute values not present in the training data. Otherwise, detecting those anomalies is a trivial task.

KDD Cup 99 Network Connections Dataset

We have used a network connection records dataset from KDD Cup 1999 [KDDCup, 1999], which contained a wide variety of intrusions simulated in a military network environment. In total there are 41 features, most of them taking continuous values. The continuous features were discretized to 5 levels ((described in more detail in §1.4.2)).

The goal of the KDD dataset was to produce a good training set for learning methods that use labeled data. Hence, the proportion of attack instances to normal ones is very

large. To create more realistic data, we have reduced the number of attack records to about 10% of the test dataset. There are a total of 24 types of attack. Some of the attacks which are Denial of Service or probing attacks are much easier to detect than other attacks. We have selected four kinds of attacks - mailbomb, guess password, warezmaster and apache2. Correspondingly, we created four test sets containing 10% records of the particular attack type, and 90% normal records. We used other normal records for training our model.

2.2.2 Training

We build our model, which includes the conditional AD Tree, the marginal AD Tree, the mutual information matrix, cache for the denominator counts §2.1.4 and the marginal count histograms using the training data. Building these comprise the training phase.

2.2.3 Testing

For each test record t, we consider every possible pair of composite attributes, that are *mutually exclusive* and *dependent* (see eqn.2.1). For each such pair, A and B, we compute $r(a_t, b_t)$. The minimum r-value is assigned as the score of the record t. In some cases we have used the combining evidence heuristic (§2.1.3) to assign score to a record. For the KDD dataset, we have also considered the partitioning method described in §2.1.3. Here we consider all possible *mutually exclusive* subsets A, B and C to compute the ratio $rval(a_t, b_t|c_t)$.

2.2.4 Evaluation

We evaluate our methods against a likelihood based approach using a Bayesian network representation and association rule based learner LERAD [Chan et al., 2006]. The conditional and marginal models are evaluated separately. For the conditional and marginal methods, we vary the value of α between 0.001 to 0.02 to generate points on the curve. For the Bayesian network method, we vary the likelihood threshold. In our plots, the

x-axis represents the detection rate, i.e., the proportion of total true anomalies that are detected. The y-axis gives the corresponding precision of detection, i.e., the ratio of number of true positives to the total number of predicted positives. A higher curve denotes better performance.

2.3 Results

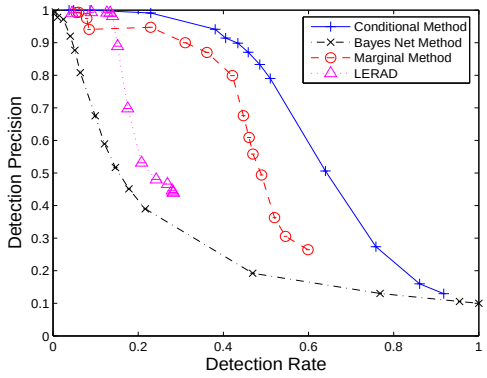
2.3.1 PIERS Dataset

In Figure 2.3 we show the comparison our methods (conditional and marginal) against the Bayesian network likelihood method and LERAD [Chan et al., 2006] on the PIERS dataset. The data points correspond to particular threshold parameter values. The points denote the average performance over 20 randomly generated test sets for each algorithm. The 95% confidence error bars are much smaller than the marker sizes. Hence any difference that appears in the plots is statistically significant.

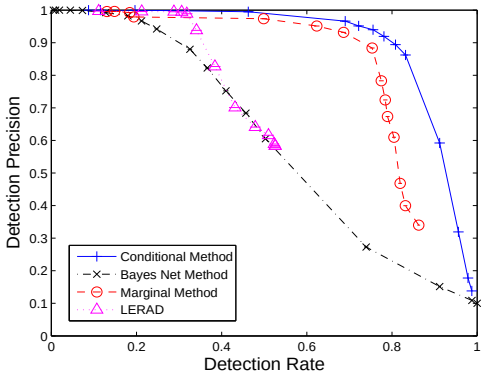
In Figure 2.3(a) we see the performance of the methods when $l = 1$, i.e., the anomalies are generated by flipping just one attribute value. For the conditional method, we set $k = 3$ for all the experiments. This means we consider up to three attributes in each composite attribute. We see that the conditional method performs best, followed by the marginal method. Both these methods outperform the Bayes net and LERAD significantly.

Figures 2.3(b) and 2.3(c) shows the performance when $l = 3$ and $l = 7$ respectively. Our methods outperform the Bayes net method and LERAD. As mentioned previously, we take $k = 3$ for the conditional method. This means that we consider up to six attributes while computing a r-value. Even though the Bayes net models the likelihood of all the attributes combined together, the conditional and marginal methods still perform better.

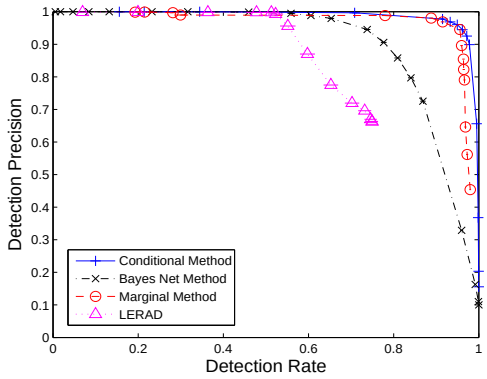
Figure 2.3(d) shows the performance when the anomalies are actually records inserted from a different month. We see that the marginal method performs the best, followed by the conditional method. The Bayes net method and LERAD perform very poorly in comparison. The superlative performance of the marginal method can be explained by



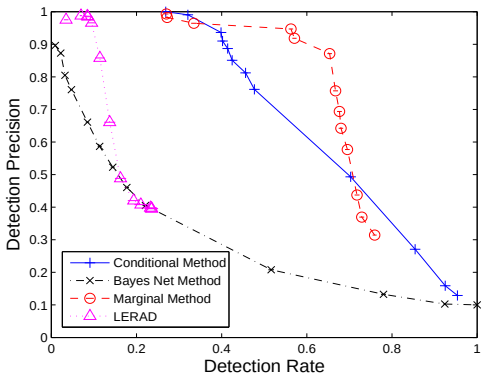
(a) Algorithm performances for $l = 1$



(b) Algorithm performances for $l = 3$

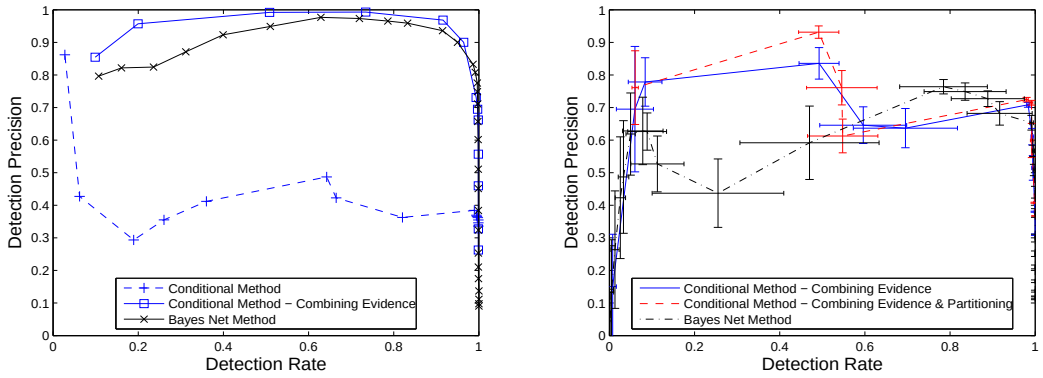


(c) Algorithm performances for $l = 7$



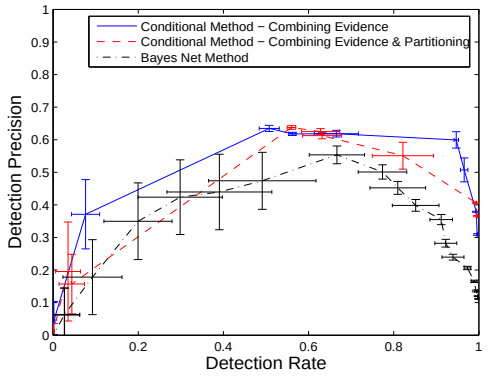
(d) Algorithm performances for inserted records from different month

Figure 2.3: Comparison of algorithm performances for the Piers dataset. The x axis is the fraction of the true anomalies found by the algorithm. The y axis is the fraction of predicted anomalies that were true anomalies. The curves are created by varying the threshold parameter α. Curves that are higher and farther to the right are better.

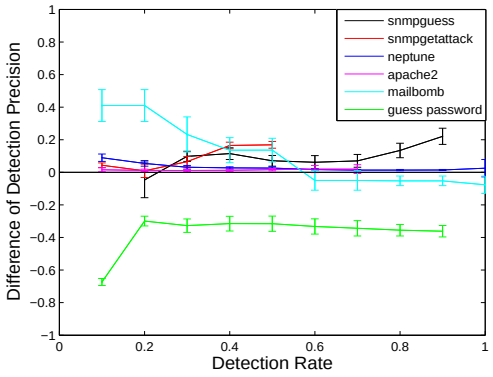


(a) Apache2

(b) Mailbomb



(c) Snmpguess



(d) Comparison for all attack types

Figure 2.4: Performance over the Network Connections KDD Cup 99 dataset

the fact that records from the other month have combinations of attribute values that are not present in the training set. The conditional method ignores these values, while the marginal method takes advantage of this fact.

Dataset	Training Size	Test Size	Number of Attributes	Training Time (secs)	Testing Time (secs)	Memory (MB)
Piers	500,000	10,000	10	6.9	4.7	4.5
KDD Cup 99	500,000	10,000	41	297	1.6	152

Table 2.1: Time and Space requirement for Bayes Network Method

Dataset	Training Size	Test Size	k	Training Time (secs)	Testing Time (secs)	Memory (MB)	Marginal Memory (MB)
Piers	500,000	10,000	1	7.6	16.8	337	334
			2	7.8	133	338	340
			3	9.3	790	341	489
KDD Cup 99	500,000	10,000	1	10.2	15	323	222
			2	44	7145	332	2618

Table 2.2: Time and Space requirement for Conditional and Marginal Methods

2.3.2 KDD Cup 99 Network Connections Dataset

On the network connections dataset, we see that some attack types are easier to detect than others. Figure 2.4 shows the performance comparison of the different methods for some of the attack types. As number of attributes is quite large, we have used up to $k = 2$ attribute combinations. This means that up to four attribute values are considered at a time. For the conditional method, we have used the heuristic to combine evidence (§2.1.3) from different attribute sets. Here, we have also compared the performance of the partitioning method §2.1.3.

The marginal method performs very poorly in this case and starts with a large number of false positives even at the lowest sensitivity level. Since this dataset has a very large

number of attributes, there is a high chance that even for normal records, there is a value of an attribute combination that is not present in the training data. This leads to flagging of a large number of records as maximally anomalous. Hence, we haven't shown the marginal algorithm curve for the plots as it performs very poorly.

We have evaluated the performance of each algorithm over 20 randomly chosen test sets of size 10,000 each. We show the average performance for each attack type. For attack types *mailbomb* and *snmpguess* we also show the 95% confidence error bars.

For attack type *apache2* in Figure 2.4(a), the original conditional method performs worse than the Bayesian network likelihood approach. But using the combining evidence heuristic results in a much better accuracy. Here, the conditional method is able to detect almost all the attacks with a very high precision rate.

For attack types *mailbomb* and *snmpguess*, the conditional method performs slightly better than the Bayes net method. Using the partitioning of training data in the conditional method results in similar or better performance to the basic method. Here we see that the error bars are quite large. Figure 2.4(d) gives a better comparison of performance between the methods. This plots the difference of detection precision between the conditional method and the Bayes net method. A positive difference means that the conditional method has higher precision. We see that for five of the attack types considered, the difference is mostly above zero. But, for the attack type *guess password* the Bayes net method performs significantly better. Here, the error bars represent 95% confidence intervals.

2.4 Conclusions

We have proposed two methods of anomaly detection in high arity categorical datasets: the Conditional method and the Marginal method. We show that performing a combinatorial search over all possible subsets of attributes (up to a certain size) gives a better performing and more meaningful anomaly detection method.

The current work focuses on finding single records that are anomalous. Sometimes in real world applications we are more interested in detecting groups of unusual records

that deviate from the norm, rather than detecting the records separately. For example, in astronomical datasets, we might be more interested in an unusual phenomenon if it keeps repeating at some interval. Just observing one such instance may not be significant, as it could be attributed to some measurement error. In biosurveillance, we might be interested in the emergence of a new disease by detecting a group of unusual but similar cases. It is specially relevant in network security monitoring, as we can detect a new pattern of user behavior from a group of records. This can signal possible malicious behavior. In the following chapters we investigate different techniques to detect such groups of anomalous records.

Chapter 3

Anomaly Pattern Detection in Categorical Datasets

3.1 Introduction

So far we looked at individual record anomalies. In this chapter, rather than finding individually anomalous records (which may be due to noise), we consider multiple anomalies following a pattern, and propose a new method for detecting such patterns of anomalies in categorical datasets. We assume that anomalies are generated by some underlying process which affects only a particular subset of the data. For example, in customs monitoring, a smuggler might be operating only from a fixed port of arrival, or might have access only to a particular shipping line. But within that subset, the smuggler will try to hide their activities by making them appear as random as possible. Similarly, in monitoring emergency department visits, a bioterrorist might have access to only a particular geographical location, or to only a particular type of disease causing agent. Thus these activities give rise to multiple anomalous records which share common values in some subset of their attributes. In this work, we develop a new detection method that can efficiently and accurately detect such patterns.

While *local anomaly detection* methods (such as in Chapter 2) can be used to detect in-

dividually anomalous records, they cannot take advantage of the fact that there are multiple anomalies from the same source which have some similarity between them. Nevertheless, these methods can be incorporated into our proposed *anomaly pattern detector*, which uses the presence of many similar anomalous records (generated by a common process) to improve the detection performance. Our method consists of two steps: we first use a *local anomaly detector* to identify individual records with anomalous attribute values, and then detect patterns where the number of anomalous records is higher than expected. Here we assume that most of the anomalous records sufficiently stand out from the normal records on their own, and can be detected by the local anomaly detector. Given the set of anomalies flagged by the local anomaly detector, we search over all subsets of the data defined by any set of fixed values of a subset of the attributes, in order to detect self-similar patterns of anomalies. We wish to detect any such subset of the test data which displays a significant increase in anomalous activity as compared to the normal behavior of the system (as indicated by the training data). We perform significance testing to determine if the number of anomalies in any subset of the test data is significantly higher than expected, and propose an efficient algorithm to perform this test over all such subsets of the data. We show that this algorithm is able to accurately detect anomalous patterns in real-world hospital, container shipping and network intrusion data.

"What's Strange About Recent Events" (WSARE) [Wong et al., 2002] is a method designed to detect clusters of anomalies in the data. WSARE operates under a different set of assumptions than our proposed method: it tries to detect anomalies evidenced by differences in the relative counts of records matching particular rules for the current and historical datasets. This is not sufficient for our purposes, since in our case, the presence of anomalies need not necessarily increase the total counts in certain subsets of the data. Rather, we use the detection capability of a feature-based local anomaly detector, and search for patterns by incorporating the output of such a detector. Here, we are interested in detecting increased incidence counts of *anomalous records* (records with unexpected attribute values, as determined by the local anomaly detector) as compared to the total number of records in a subset of the data. The detection of such patterns with many anomalies matching certain rules indicates the presence of anomalous processes.

To formalize our problem, we assume that we have a sufficiently large *training dataset* which defines the normal behavior of the system. We typically have unlabeled training data, in which we assume that no anomalies are present, but our methods can tolerate the presence of a small percentage of anomalies in the training set. Our goal is to detect the presence of patterns of anomalies in an unlabeled *test dataset*, where each pattern corresponds to a fixed set of attribute value(s). There might be single or multiple such anomalous patterns present, possibly generated by several distinct causes. We want to detect the anomalous records generated by such patterns, while minimizing the false positive rate and avoiding detection of irrelevant anomalies due to noise. Much of this chapter has been adapted from our paper in KDD 2008 [Das et al., 2008].

3.2 Anomaly Pattern Detection

Our proposed method can be thought of as generalizing two lines of previous research: the use of standard anomaly detection methods to detect individually anomalous records, and the use of WSARE 2.0 [Wong et al., 2002] to detect anomalous clusters of counts in categorical data. We generalize the former method by integrating information from *patterns* of potentially anomalous records. We extend WSARE by using the information from a local anomaly detector and determining if any subset of the data has more anomalous records than expected. This is distinct from the original formulation of WSARE, which detects subsets with more total records than expected and does not consider whether each individual record is anomalous.

3.2.1 Local Anomaly Detection

In this work, we use two local anomaly detection methods to score the records individually. Our method of pattern detection uses the output of either of these algorithms to search for patterns. We briefly describe both these methods of local anomaly detection.

Bayesian Network Anomaly Detection We use the method described in §1.3.5. We learn the structure and parameters of a Bayes Net using the training data, compute the likelihood of each record in the test dataset given the Bayes Net model, and report test records with unusually low likelihoods as potential anomalies.

Conditional Anomaly Detection As described in Chapter 2, in the Conditional Method a score is then assigned to the test record t based on all r-values corresponding to all possible pairs of attribute sets. The score is defined as the maximum value of the product of r-values over all possible partitions of the attributes for record t. In our experiments we use the parameter values $k = 2$ and $\alpha = 0.02$ for the conditional method in most cases. Here, k is the maximum set size of A or B. α is the threshold for the r-values to be significant. For the KDD Cup 99 dataset (§3.3.3), we use $k = 1$ since it has a larger number of attributes.

3.2.2 WSARE

The WSARE 2.0 method [Wong et al., 2002] searches over all possible rules in the dataset. Each rule R can be written as $R : A = a_j$, where A is a subset of attributes and a_j is an assignment of attribute values. WSARE considers rules with one component (e.g. $Country = Japan$) or two components (e.g. $Country = Japan$ AND $Shipper = ShipCo$). It determines whether the count of cases that match the rule in the test dataset is significantly different from the expected count determined by the training dataset. The statistical significance of each rule is determined by using a Fisher's exact test on the two by two table (Table 3.1), where $C(R)_{test}$ and $C(R)_{train}$ represent the numbers of test records and training records corresponding to rule R, and C_{test} and C_{train} denote the total numbers of test and training records respectively.

To account for multiple hypothesis testing, these p-values are adjusted using a randomization test. In a later version of the algorithm (WSARE-3) [Wong et al., 2003], the authors consider determining the baseline using a Bayesian Network rather than directly using the counts from the training dataset. We use the algorithm WSARE-2 with up to two

	Test	Train
Match R	$C(R)_{test}$	$C(R)_{train}$
Do not match R	$C_{test} - C(R)_{test}$	$C_{train} - C(R)_{train}$

Table 3.1: 2×2 Contingency Table for WSARE

component rules for comparing against our methods.

To understand the key difference between our current problem and that considered in WSARE, let us first look at what we mean by an *anomalous pattern*. Here, there are two factors to consider. The first factor is that each individual record is individually anomalous with respect to some normal behavior. The second factor is the *pattern* formed by these anomalies (defined by some constraint of similarity between them) which signifies that the records are generated by the same underlying anomalous process. WSARE does not take the anomalousness of each individual record's attribute values into account, but instead counts the number of records corresponding to a given rule and reports rules for which these counts are anomalous. In our current work, an anomalous process can generate a *pattern* of anomalous records that are similar with respect to a particular subset of the attributes, but which are anomalous due to unusual values in any (potentially different) set of attributes. This definition of a pattern is particularly useful when we have an adversarial process creating the anomalies. The adversary might try to make the generated records look as random as possible, but might be restricted to a particular set of fixed values of some of the attributes. For example, in customs monitoring, a smuggler wants to smuggle goods using a variety of methods to avoid detection, but they might have access to only a particular port or shipping line. In such a case, detecting increased incidence of suspicious activity corresponding to that subset of the data can alert us to the illegal activity.

3.2.3 Algorithm

To detect the presence of anomalies in this scenario, we first make use of a local anomaly detector that can detect individual anomalies in a dataset. Any such detector may detect

many false positives. In order to successfully determine if a subset of the test data has a higher than expected concentration of true anomalies, we compare it to the corresponding subset in the training data. If the number of positives in the subset of the test data is significantly larger than what is expected from the training data, it signals the presence of true positives clustered in that subset. The outline of our anomaly pattern detection algorithm is given in Figure 3.1.

While searching for patterns of anomalies, we retain the concept of anomalousness of individual records. In **Step 1** of our algorithm we score all the records of both the test and training dataset using one of the local anomaly detection algorithms described in §3.2.1. Our anomaly detector requires baseline or training examples which correspond to the normal behavior of the system. While scoring the test records, the training dataset is used as the baseline. To score the training records, we use a leave-one-out approach, where the entire training data excluding the current record is used as the baseline. We then set a score threshold (**Step 2**), and all records that are more anomalous than the threshold are flagged as anomalies. The threshold score is set such that a fixed proportion of the records in the training dataset ($PositiveRate$) are marked as positives or anomalies. For example, when $PositiveRate = 0.1$, the threshold is chosen so that 10% of the records are flagged as positives in this step. We use these "most anomalous records" to detect patterns in the data. We would like to set the value of $PositiveRate$ such that most of the true anomalies in the test data are captured within the top $PositiveRate$ proportion of anomalous records. In the case of the training dataset (which is assumed to contain no true anomalies), the flagged anomalies can be thought of as the false positives reported by the local anomaly detector. We wish to compare this false positive rate to the number of anomalies detected in subsets of the test data to determine the presence of patterns of true anomalies.

In **Step 3** we search over all possible rules of the form $R : A = a_j$. Here A denotes any subset of attributes of size up to k and a_j is the jth value combination of A. For example, if $A = \{Country, Shipper\}$, a_j can correspond to any fixed combination of Country and Shipper Name. Each rule R defines subsets of the test and training datasets respectively, corresponding to the records that match the rule. For each rule R, we determine

the number of records in the corresponding test and training subsets of the data ($C(R)_{test}$ and $C(R)_{train}$) and the count of positives detected by the local anomaly detector in those subsets ($C(R)^+_{test}$ and $C(R)^+_{train}$).

Our null hypothesis is that the proportion of detected positives by the local anomaly detector will be the same in the test and training datasets. When true positives are present in the test dataset, the null hypothesis may be rejected, since we would expect to see a higher proportion of detected positives in the affected subset of the data. To test these hypotheses we use a one-sided Fisher's Exact Test [Good, 2000] (using Stirling's approximation to calculate the factorials) on the 2×2 table (Table 3.2). We use a one-sided test since our alternate hypothesis is that $C(R)^+_{test}$ is *higher* than expected. This gives us a p-value for each such rule tested.

	Test	Train
Positives	$C(R)^+_{test}$	$C(R)^+_{train}$
Negatives	$C(R)_{test} - C(R)^+_{test}$	$C(R)_{train} - C(R)^+_{train}$

Table 3.2: 2×2 Contingency Table for Anomaly Pattern Detection

Since we are searching over all possible anomalous patterns rather than considering isolated anomalies, we are performing multiple hypothesis tests, increasing the expected number of false positives proportional to the number of tests performed. To compensate for multiple testing, we use the False Discovery Rate (FDR) method [Benjamini and Hochberg, 1995]. It is used to find a critical value for the hypothesis tests such that the expected proportion of false positives is below α. In our experiments we use FDR with $\alpha = 0.9$. We use a high value of α because we want to compare the different methods over a wide range of recall values. Using a lower value of α will give us fewer false positives, but at the cost of a lower recall rate. In real-world applications, we can use an appropriate value of α based on our desired false discovery rate.

Step 4 of our algorithm outputs the most anomalous patterns found in Step 3. Additionally, for comparison to the baseline method of the local anomaly detection that does not consider patterns, we assign an anomalousness score for each individual record R in

the test data. The score of R is set equal to the score assigned by the local anomaly detector if it belongs to one of the detected patterns. The significant patterns may cover only a small subset of the true anomalies present, giving a low recall rate. To compare the algorithms over the entire range of recall values, we append the rest of the records to our list of detected anomalies. To score these records we adjust the local anomaly detector score such that they are less important than the records belonging to a pattern, but retain the original ordering from the local anomaly detector.

3.2.4 Computational Speedup

Since we consider all possible attribute sets up to a size k, and all possible value combinations corresponding to these sets, the total number of possible rules is $O(n^k a^k)$, where n is the total number of attributes, and a is the maximum arity. We can have a large number of such rules for large values of n or a. k is usually set to 2 or 3 in our experiments. To be able to efficiently search over all the rules, we employ several computational speedup techniques as described below:

Using AD Trees for Computing Counts

The required counts ($C(R)^+_{test}$, $C(R)^+_{train}$, $C(R)_{test}$, and $C(R)_{train}$) are conjunctive counting queries on the dataset, and can be efficiently queried using an AD Tree [Moore and Lee, 1998]. The AD Tree building algorithm scans the dataset once, and precomputes information needed to answer every possible query in time independent of the number of records. The parameter *leaflist size* can be adjusted to obtain a tradeoff between the memory used and the query response time. We build two separate AD-Trees for the training and test datasets respectively. We append an extra Boolean attribute to each record indicating whether it has been flagged by the local anomaly detector as a positive. This attribute is used to retrieve the counts for the positive cases.

Ignoring Rare Values

We can treat rare values in a way similar to what was done in Chapter 2. For computational efficiency we can set a lower bound min_size, on the size of the test subset ($C(R)_{test}$) corresponding to a rule R. This means we are only interested in patterns of anomalies that affect subsets of the data larger than min_size. If $C(R)_{test} < min\_size$, we then ignore the rule R. Predefining the value of min_size can save us computational time and memory, especially if some of the attributes have high arity. Consider the jth value x_j of the attribute X. If x_j occurs less than min_size times in the test dataset, then it is easy to see that any rule R containing x_j will be ignored. We call such values of the attributes which occur less than min_size times in the test dataset as *rare* values and all other values are as *common* values. We can replace all the rare values of each attribute by a generic rare value. While considering the possible rules we ignore this generic rare value for each attribute. This scheme of keeping only the common values reduces the arity of each attribute and significantly reduces the memory required to build the AD Tree. This also reduces the total number of rules that we need to consider and hence gives us a computational time saving as well. In our experiments, we have set $min\_size = 10$.

Pruning the Search Space

Since anomalies are usually rare, we use another simple trick to speed up computation. If a rule R corresponding to some set of attribute values has no anomalies in the test data ($C(R)^+_{test} = 0$), then all rules R' which contain the same set of attribute values (along with some other attribute values) will also have $C(R')^+_{test} = 0$. Hence, once we find a rule R that does not correspond to any anomalies in the test dataset, we can prune away all the rules that are an extension of R.

3.3 Datasets

We evaluate the methods on the three datasets described below.

3.3.1 PIERS Dataset

Our first dataset consists of records describing containers imported into the country from various ports in Asia as described in §1.4.1.

Since there were no labeled anomalies in the original data, we create synthetic anomalies by randomly altering attribute values for a subset of the data. We first partition the dataset into training and testing sets. We randomly choose 10,000 records from the data as a test set, and then choose 100,000 of the remaining records to form the training set. We modify a random $NumAnom$ records of the test set records to be anomalous patterns, as described below.

Our goal is to identify patterns or groups of anomalies in the data. A pattern is defined as a set of anomalous records which belong to a particular subset of the data, characterized by one or more fixed values of the attribute(s). To create such patterns in the dataset, we adopt the following procedure:

CreatePattern($Data_{test}, NumAnom, MinSetSize, PatternRate$)

1. Initialize $NumGenerated = 0$.
2. Select a rule $R : A = a$ where A is a set of up to k attributes, and a is any combination of values of those attributes, uniformly at random.
3. Select the set of records $Data(R)_{test}$ that match the rule R in $Data_{test}$.
4. If $Size(Data(R)_{test}) < MinSetSize$, goto Step 2 and reselect a rule R.
5. Choose a random $PatternRate$ fraction of records from $Data(R)_{test}$. For each record T which is selected (and as long as $NumGenerated < NumAnom$):
   (a) Choose an attribute X_{rand} uniformly at random.
   (b) Draw a random value val_x of attribute X_{rand} from the marginal distribution of values of X in $Data_{train}$.
   (c) Replace the value of X_{rand} in T by val_x

(d) Update $NumGenerated = NumGenerated + 1$.

6. If $NumGenerated < NumAnom$ then goto step 2 else stop.

This algorithm creates anomalies in particular subsets of the data corresponding to randomly chosen rules. We have a restriction on the minimum size of the subset of data since very small patterns are almost indistinguishable from randomly chosen individual records. We set $MinSetSize = 200$ for all our experiments. Once we choose a suitable rule R, we affect a fixed fraction ($PatternRate$) of them to be anomalous. A high value of $PatternRate$ would mean that a large fraction of records corresponding to the rule R are anomalous and make such patterns easier to detect by the pattern detector. Each record in the pattern is anomalous in the sense that it has an attribute value changed randomly. This breaks the relationship of that attribute with the rest of the attributes. Our goal here is to use the similarity pattern in these anomalies to improve the performance of our detection algorithm.

Anomalies are injected into this dataset using the method described above. We consider one possible real world scenario where we might see such anomalous patterns. A smuggler can try to smuggle in goods using various means, but might have access to only a particular US port of arrival. Hence even if he tries to avoid detection by hiding the smuggled containers randomly, the fact that an unusual number of suspicious cases are seen at a particular port gives a strong indication of illegal activity.

3.3.2 Emergency Department Dataset

This real-world dataset contains records of patients visiting emergency departments (ED) from hospitals around Allegheny county in the year 2004 as described in §1.4.4. Each record consists of six categorical attributes: the hospital id, prodrome, age decile, home zip code and the chief complaint class. The dataset is injected with simulated ED cases resembling an anthrax release. The simulated cases of anthrax were produced by a state of art simulator [Hogan et al., 2007] that implements a realistic simulation model of the effects of an airborne anthrax release on the number and spatial distribution of respiratory

ED cases. We treat the first two days when the attack symptoms begin to appear as the test data, thus evaluating our ability to detect anthrax attacks within two days of the appearance of symptoms. We train our model on the previous 90 days' data.

3.3.3 KDD Cup 1999 Network Intrusion Detection Dataset

We have also evaluated APD on the KDD Cup 1999 data [KDDCup, 1999], which contained a wide variety of intrusions simulated in a military network environment (described in §1.4.2). The goal of the KDD dataset was to produce a good training set for learning methods that use labeled data. Hence, in this case we have labeled anomalies (network attacks) and the proportion of attack instances to normal ones is very large. To create more realistic data, we have reduced the number of attack records to 1% of the test dataset. We have run our algorithms on the 6 most common types of attacks - apache2, guess password, mailbomb, neptune, smurf and snmpguess. Correspondingly, we created six different test sets containing 1% records of the particular attack type, and 99% normal records. We use rest of the normal records for training our model.

3.4 Evaluation and Results

We compare the performance of our Anomaly Pattern Detection (APD) method to the baseline method of just using the local anomaly detector. We compare the performance of both the baseline methods described in section 3.2.3, choosing the better one to use as the base method for pattern detection.

We note that our anomalous pattern detection algorithm is similar to running WSARE on a dataset where each record is augmented by a binary indicator attribute L, denoting the output of the local anomaly detector. But it differs from this augmented version of WSARE (WSARE-AUG) in the following ways:

1. WSARE-AUG searches over all possible rules including ones which are not related to the anomaly feature. The rules we consider always include the feature L.

2. We perform a one-sided significance test since we are interested only in increases in the proportion of anomalies.

3. Our search over rules is different from WSARE-AUG. We search only over rules of the form $\{L = 1\}|R$ ($\{L = 1\}$ conditioned on R), where the rule R can contain up to k components. WSARE-AUG chooses the best one component rule C_0 and then finds the best two component rule $\{C_0, C_1\}$ where the rules $C_0 \mid C_1$ and $C_1 \mid C_0$ are both determined to be significant.

We also compare the anomaly detection performance of our method of pattern detection to both WSARE and WSARE-AUG.

The procedure for generating the test and train data and injecting anomalous patterns is randomly repeated 50 to 100 times for each dataset. We run each algorithm on these datasets in order to obtain 95% confidence intervals on the performance measure. The evaluation criteria we use is the ability of each algorithm to identify each individual anomaly correctly. We plot the detection precision, i.e. the ratio of number of true positives to the total number of predicted positives, against the detection rate, i.e. the proportion of total true anomalies that are detected. A point on the plot is obtained by setting a particular threshold score $Score_T$ to flag anomalies. Any record having a score greater than $Score_T$ is flagged as an anomaly. The corresponding precision and detection rate are then calculated. By varying $Score_T$ we obtain the plot for the entire range of detection rates. This threshold is varied independently for each of the methods. Here, a higher curve denotes better performance, since it corresponds to a higher detection precision for a given detection rate.

Figures 3.2(a) and 3.3(a) gives the performance plots using the PIERS dataset and anomaly patterns generated using $NumAnom = 100$ and $PatternRate = 0.1$. This gives a nominal detection precision of 0.01 if we randomly select records. The parameter values used in the Anomaly Pattern Detection algorithm are: $PositiveRate = 0.1$, $k = 2$ and $\alpha = 0.9$. All the plots also show the 95% confidence intervals for the performances.

Figure 3.2(a) compares the performance of the two baseline methods and WSARE on this dataset. We see that the conditional method performs best. The Bayesian Network

method performs quite poorly in this case. Since our method of pattern detection relies on the output of a baseline local detection method, we choose the better performing method for our experiments. Note that the detection precision of WSARE is almost the same as the *chance* precision. This shows that WSARE is unable to detect the kind of anomalies that we consider here. This is not surprising since we do not increase (or decrease) the count of any particular subset of the data, which is what WSARE attempts to detect.

We ran WSARE-AUG (§3.2.3) on this dataset, augmenting each record with the output from the local anomaly detector. In all cases, the most interesting rule detected by WSARE-AUG is that there is a larger proportion of anomalies in the entire test dataset as compared to the training dataset. Also, no other rules were reported containing the component $L = 1$, where L is the augmented anomaly attribute. This gives a degenerate result that all the anomalies detected by the local anomaly generator are actually anomalies. So, in effect we do not get any improvement in performance using WSARE-AUG over the baseline methods. This same effect is seen when WSARE-AUG is run on the other datasets.

Figure 3.3(a) compares the performance of our proposed anomaly pattern detector (APD) with the baseline method of conditional anomaly detection (§3.2.1) on the PIERS data, with anomaly patterns generated using $PatternRate = 0.1$. In this case the pattern detection algorithm uses the conditional method as its local anomaly detector. Figure 3.3(b) shows the performances when $PatternRate = 0.2$. We see that in both these cases the pattern detection method performs significantly better (with a significance level of $\alpha = 0.05$) than the baseline. For the higher value of $PatternRate$ we see a greater improvement in performance as expected. We also evaluated the performance with the parameter $PositiveRate$ varying between 0.05 and 0.3. The detection performance does not vary much with different values of the parameter. In general, the value of this parameter can be set based on our estimation of the proportion of anomalies that might be present in the dataset.

Our goal in this work is to use the patterns formed by the anomalies to detect them more effectively (with fewer false positives). However to give a better understanding of how well our algorithm can correctly identify the rules that generated the anomaly clusters

Table 3.3: Normalized area under the curves for KDD Cup 99 Dataset comparing Baseline and APD, with 95% CI

Attack Type	Baseline	APD
apache2	0.9636 ± 0.0057	0.9668 ± 0.0053
guess_passwd	0.7316 ± 0.0133	0.7792 ± 0.0145
mailbomb	0.1782 ± 0.0104	0.2243 ± 0.014
neptune	0.9938 ± 0.003	0.9938 ± 0.003
smurf	0.6758 ± 0.0125	0.7662 ± 0.0131
snmpguess	0.9616 ± 0.0059	0.9773 ± 0.0045

in the data, we perform an alternate evaluation. Since our datasets either have a large number of attributes, or the attributes have very high arity, the number of possible rules is very large. Also, due to the strong dependence between different variables, multiple rules can correspond to very similar subsets of the data. Hence instead of trying to retrieve the exact rules, we measure the similarity between the rules detected by APD and those which were used to generate the anomalies as described in §3.3.1. We use an intuitive similarity index to calculate the overlap between these two sets of rules. Let d_1 and d_2 denote the subsets of the data that matches the two sets of rules. Then the Jaccard index [Jaccard, 1912] is defined as $\frac{Size(d_1 \cap d_2)}{Size(d_1 \cup d_2)}$. A higher value of this index denotes a greater degree of similarity between the rule sets. For the experiment corresponding to figure 3.3(a) the average Jaccard index of APD is 0.27. We can compare this with the average Jaccard index of 0.15 for the null rule that matches all records in the test set. We achieve an improvement by a factor of about 2 in this case.

Figure 3.2(b) shows the comparison of APD with the baseline methods and WSARE on the emergency department dataset. Note that the WSARE algorithm was originally developed to detect anomalies in this context. However, in Wong et. al. [Wong et al., 2002] the evaluation criteria used was to detect the presence of increased counts of patients rather than to identify the particular patients showing anomalous behavior. We see that the baseline method of using Bayes Net and WSARE perform very similarly. The conditional

method performs better than both these methods in the recall range [0,0.5]. The conditional method does not assign a score to every record, but only scores the records that it flags as anomalies. Hence, it does not extend beyond recall rate 0.5 as the remaining anomalies are not detected by the method. We see that APD gives a significant improvement in performance within the same range. The curve for APD also includes the rest of the records (ones not flagged by the conditional method) appended in some random order. This causes the curve to extend beyond recall rate 0.5, but decreases the precision rate below the other methods in that range.

Figures 3.4(a) and 3.4(b) gives the comparison of APD with the conditional method for attack types guess password and smurf in the KDD Cup 99 dataset. We have summarized the results for the 6 attack types in table 3.3. It gives the normalized area under the curves for the baseline conditional method and APD for the recall range [0.1,0.9]. We see that APD gives a significant improvement in the detection precision for the attack types guess password, mailbomb, smurf and snmpguess. The remaining two attack types apache2 and neptune are very easy to detect by the conditional method and APD does not give a significant increase in precision.

3.5 Conclusions

We propose a new method to search for patterns of anomalies in large multidimensional categorical datasets. Our method utilizes the output from a local anomaly detector to locate subsets of the data that might be affected. We consider two such local anomaly detectors, the Bayesian Network likelihood method, and conditional anomaly detection method. We also note the similarity and differences of our proposed method of anomaly pattern detection (APD) to a rule based anomaly detector WSARE. We evaluate the performances of these algorithms on three real world datasets with synthetic and real anomalies. We show that APD performs significantly better at detecting anomalies over the other methods.

We also note that the pattern search in APD is orthogonal to the local anomaly detection method. We can use any such local anomaly detector which is more appropriate for

a given domain. Finally, while we believe that the chosen BARD outbreak simulation is a highly realistic model of anthrax release, evaluating our methods on real, known disease outbreaks can provide more robust evidence of the usefulness of our method.

Figure 3.1: Anomaly Pattern Detection (APD) Algorithm
Input Datasets: *test dataset* and *training dataset*
Parameters: $PositiveRate$, k, α

1. Use any local anomaly detector to score all the records in *test dataset* and *training dataset*.

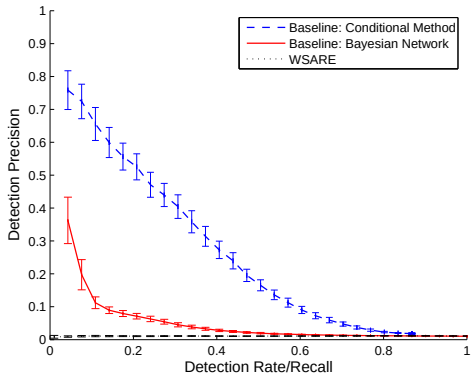
2. Fix an anomaly score threshold using the parameter $PositiveRate$. Label all records in the test and training datasets which are more anomalous than the threshold to be anomalies.

3. For each possible rule $R : A = a_j$, where a_j is any value combination of any subset of attributes A containing up to k attributes:

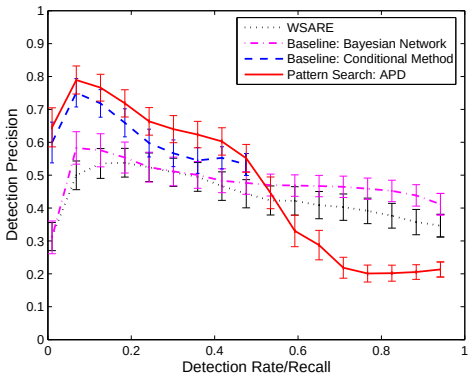
   (a) Compute the counts in the 2×2 contingency table shown in Table 3.2. These correspond to the number of records matching the rule R and the number of positives detected in them for both the training and test datasets.

   (b) Use Fisher's exact test to determine the p-value of the alternate hypothesis that the count $C(R)^{+}_{test}$ (number of detected positives in the test dataset that match the rule R) is higher than what is expected under the independence assumption (null hypothesis).

4. Output all *patterns* that have significantly higher test case anomalies. Use FDR method (with parameter α) to determine the significant patterns.

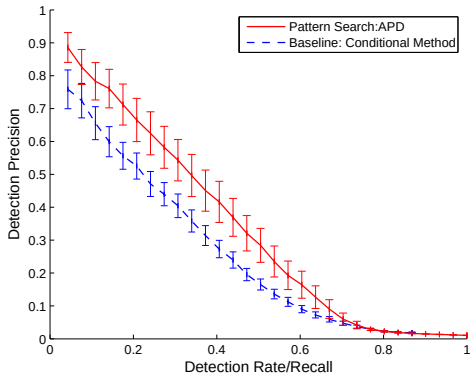


(a) PIERS dataset: Baseline methods for $PatternRate = 0.1$

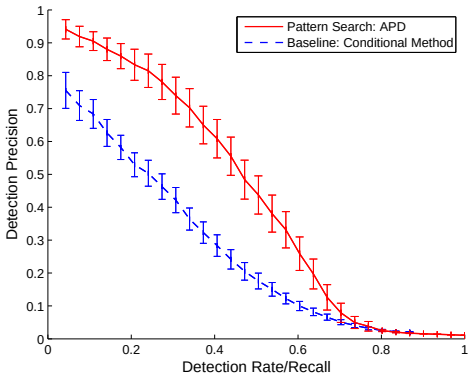


(b) ED dataset: Baseline Methods and APD

Figure 3.2: Detection precision vs. recall curves for PIERS and ED datasets

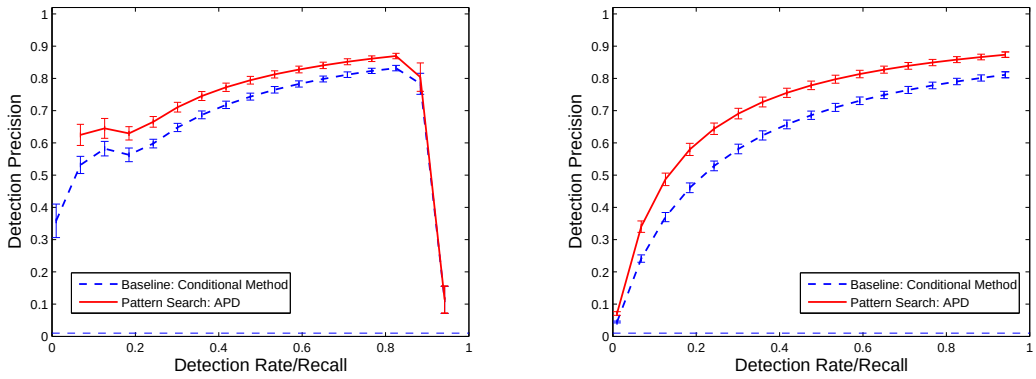


(a) $PatternRate = 0.1$



(b) $PatternRate = 0.2$

Figure 3.3: PIERS dataset: Performance comparison between pattern detection and baseline, with 95% confidence intervals



(a) guess password (b) smurf

Figure 3.4: KDD Cup 99: Performance comparison between pattern detection and baseline

Chapter 4

Detecting Anomalous Groups in Categorical Datasets

4.1 Introduction

In this chapter, we consider another scenario for detecting groups of anomalies in categorical datasets. In Chapter 3, we considered the case when multiple anomalies generated by a common process are loosely self-similar based on one (or a few) attribute values. We also assumed that the individual anomalous records sufficiently stood out from the rest on their own. In contrast, in certain situations, anomalous groups of highly self similar records might be generated by a common process. For example, in the case of network intrusion, the same task might be repeated a number of times to gain unauthorized access to a system. In health monitoring, a disease outbreak can lead to a large number of disease cases with almost identical features being reported. In these cases, the individual records belonging to the group might not appear anomalous by itself, but as a group they stand out from the rest. We propose a novel technique of Anomalous Group Detection (AGD) to detect such groups of anomalous records in categorical valued datasets.

Our approach is a generalization of the spatial scan statistic, a commonly used method for detecting clusters of increased counts in spatial data. We extend this framework to

non-spatial datasets with discrete valued attributes, where the degree of anomalousness of each record depends on its attribute values and we wish to find self-similar groups of anomalous records. We model the relationship between the attributes using a probabilistic model (e.g. Bayesian network), define a likelihood ratio statistic in terms of the pseudo-likelihoods for the null and alternative hypotheses, and maximize this statistic over all subsets of records. Since an exhaustive search over all such groups is computationally infeasible, we propose an efficient (but approximate) search heuristic. We show that this algorithm is able to accurately detect anomalous groups in real-world hospital, container shipping and network connections data.

To formalize our problem, assume we have a sufficiently large *training dataset* which defines the normal behavior of the system. We typically have unlabeled training data, in which we assume that no anomalies are present, but our methods can tolerate the presence of a small percentage of anomalies in the training set. Our goal is to detect the presence of groups of anomalies in an unlabeled *test dataset*. There might be single or multiple anomalous groups present, possibly generated by several distinct causes. We want to detect the anomalous groups of records, while minimizing the false positive rate. This chapter has been adapted from our paper submitted in KDD 2009 [Das et al., 2009].

4.2 Related work

Our proposed method can be thought of as generalizing two lines of previous research: the use of Bayesian networks and other probabilistic models to detect individually anomalous records in data, and the use of spatial scan statistics to detect clusters in spatial data. We extend the former method by integrating information from *groups* of anomalous records, and generalize the latter method from a simple univariate model (Poisson-distributed and spatially labeled counts) to multivariate datasets.

The Bayesian Network anomaly detection method (§1.3.5) is used as a baseline algorithm in our empirical studies. We also compare the performance of our proposed anomalous group detection method to another individual record anomaly detector, the Condi-

tional Method, described in Chapter 2.

One of the most important statistical tools for cluster detection is the *spatial scan statistic* [Kulldorff and Nagarwalla, 1995, Kulldorff, 1997, Neill and Moore, 2005]. This method searches over a given set of spatial regions, finding those regions which maximize a likelihood ratio statistic and thus are most likely to be generated under the alternative hypothesis of clustering rather than the null hypothesis of no clustering. Kulldorff's framework assumes that the count of data points in a region S is Poisson distributed with some unknown rate of incidence q. Then the goal of the scan statistic is to find regions where the incidence rate is significantly higher inside the region than outside. The statistic used for this is the likelihood ratio $F(S) = \frac{P(Data \mid H_1(S))}{P(Data \mid H_0)}$, where the null hypothesis H_0 assumes no clusters, and the alternative hypothesis $H_1(S)$ assumes a cluster in region S. Under H_0, we assume a uniform incidence rate q_{all}, while under $H_1(S)$ we assume that the incidence rate is higher inside region S than outside (i.e. $q_{in} > q_{out}$). The spatial scan in described in more detail in §1.3.2.

For the spatial scan, each data point consists of a set of real-valued location attributes, which can be mapped to a point in a Euclidean space, as well as a real-valued count. The search regions are defined in terms of the location attributes, while the likelihood ratio statistic is a function of the aggregate counts inside and outside a region. The spatial scan searches over subsets of the data which are geographically contiguous. For computational efficiency, further size and shape restrictions may be imposed on the set of search regions [Kulldorff, 1997].

Rule-based algorithms have been proposed to detect groups of records. They find anomalous patterns by searching over *rules* of the form "$A_1 = v_1$ and $A_2 = v_2$" (e.g. Gender = Male and Symptom = Cough), where each rule defines a subset of records with the given attribute values. Anomaly Pattern Detection (APD) as described in Chapter 3, begins with an individual anomaly detector and then uses the rule learning method to find groups of records that have an abnormally high proportion of individual anomalies. What's Strange About Recent Events (WSARE) [Wong et al., 2003] compares the actual and expected numbers of records fitting a rule using Fisher's Exact Test, and finds rules (subsets of records) with a higher or lower number of records than expected. We compare to both of

these methods in our empirical studies.

The patterns detected by APD and WSARE are constrained to match a particular rule, and therefore are not flexible enough to include arbitrary subsets of the records. Another limitation of APD is that it can detect anomalous patterns only when the individual records forming the pattern are anomalous enough to be detected by the individual anomaly detector. We propose a method that can overcome the above limitations, finding arbitrary subsets of records that may not be individually anomalous but are anomalous when considered together.

4.3 Anomalous group detection

We would like to generalize the methodology of spatial scan statistics to find anomalous groups in arbitrary, non-spatial datasets with discrete valued attributes. This problem differs from spatial cluster detection in several respects. First, we do not have a defined set of location attributes, and thus we can no longer predefine a set of search regions based on geographical attributes such as size, shape, or contiguity. While we could conceivably define a distance metric between records with categorical attributes, we do not have a direct embedding of the data points in Euclidean space or a notion of adjacency between different attribute values. Nevertheless, we want to formulate a measure of how well the data points fit as a *group* based on the similarity between them. We must then search over subsets of the data in order to find the most anomalous groups.

The second key difference is in the way we define the anomalousness of a data point or a group of points. Scan statistics are usually applied to detect over-densities of records in a given space. They assign the same level of interest or importance to each record, and aggregate individual records to counts to determine the anomalousness of a cluster. In our case, each record has many discrete-valued attributes rather than a single real-valued count, and can have an inherent degree of anomalousness depending on its features. Most records are generated from the "normal" (or usual) distribution of data and hence are not interesting for our purpose. We assume that the normal behavior of the data is defined by

a model learned from a training dataset. Here we are no longer trying to detect simple over-densities of records in a certain feature space, but to detect groups of records that are both anomalous and also self-similar in some respect.

Instead of treating these two issues independently, we propose an approach that handles them simultaneously. As in the spatial scan statistic, our goal is to find a set of records that maximizes the likelihood ratio statistic $F(S) = \frac{P(Data \mid H_1(S))}{P(Data \mid H_0)}$, where H_0 is the null hypothesis that there are no anomalies present, and $H_1(S)$ is the alternative hypothesis specifying that the set S is an anomalous group. We assume suitable probability distribution models for both the null and alternative hypothesis, and compute the data likelihoods given these models. More precisely, we learn a probability distribution model from the training dataset, which is assumed to contain no anomalies. Under the null hypothesis H_0, all data records are assumed to be drawn independently from this model. Under the alternative hypothesis $H_1(S)$, the records contained in subset S are assumed to have been drawn from a different probability model, while the rest of the data records are generated from the null model. We assume that data points are conditionally independent given the model, and thus records not contained in subset S have identical likelihoods given $H_1(S)$ and H_0. Thus the likelihood ratio statistic simplifies to:

$$F(S) = \frac{P(Data_S \mid H_1(S))}{P(Data_S \mid H_0)} = \frac{\prod_{i \in S} P(R_i \mid H_1(S))}{\prod_{i \in S} P(R_i \mid H_0)} \tag{4.1}$$

where $Data_S$ represents the subset of the data S and R_i is the ith record in $Data_S$. We note that the probability model parameters, but not the structure, for the alternative hypothesis $H_1(S)$ are learned directly from the records in $Data_S$. Since the number of records in group S may be small and we are using this data to fit a (potentially) large number of model parameters, data sparsity is a serious problem. In particular, learning the model parameters from the data $Data_S$ and evaluating the likelihood $P(Data_S \mid H_1(S))$, results in overfitting of the model. Using this as a part of the scoring function leads to the inclusion of a large number of irrelevant records in the best scoring group, as discussed in §4.3.2.

We use a two part approach to dealing with the problem of overfitting for the alternative hypothesis $H_1(S)$. First, we use Laplacian smoothing in the parameter estimation. Second,

we use a "leave-one-out" method to compute the likelihood, which results in the following pseudo-likelihood:

$$P_{pseudo}(Data_S \mid H_1(S)) = \prod_{i \in S} P(R_i \mid H_1(S - \{R_i\})) \tag{4.2}$$

This means that while computing the likelihood of the record R_i under the alternate hypothesis, we use a probability model with parameters learned from all the records in S minus R_i. Since we do not use the same record to estimate the parameters and to evaluate the likelihood, we expect to reduce the risk of over-fitting. We now define the group score as:

$$F(S) = \frac{P_{pseudo}(Data_S \mid H_1(S))}{P(Data_S \mid H_0)} \tag{4.3}$$

This scoring metric gives a higher score to anomalous records, as well as setting a constraint of similarity between the records in a group. If the records in S are similar to each other, then the alternate hypothesis will be able to model them tightly. This will result in a high value of the likelihood $P_{pseudo}(Data_S \mid H_1(S))$, thus increasing the score $F(S)$. Also, records that are poorly modeled by the null hypothesis will have a low value of the likelihood $P(Data_S \mid H_0)$, again increasing the group score $F(S)$. Hence maximizing this score leads to grouping of similar records and at the same time it prefers records that are anomalous (i.e. records with low likelihood under the null hypothesis).

4.3.1 The AGD Algorithm

We will now describe our method for anomalous group detection (AGD). An overview of the algorithm is given in Figure 1, and we now explain each step in detail. Although any probability distribution model can be used, we choose Bayesian Networks to model the probability distribution, and will specifically refer to them in the following description.

Step 1 of our algorithm is to learn the Bayes Net corresponding to the null hypothesis. We perform structure learning on the training dataset using the Optimal Reinsertion algorithm [Moore and Wong, 2003]. We assume this same Bayes Net structure for both H_0 and $H_1(S)$. We then learn the conditional probability table parameters of H_0 from the training dataset using smoothed maximum likelihood estimation.

1. Learn the probability model for the null hypothesis H_0 from the training data.
2. For all subsets of the data S:
   (a) For each $R_i \in S$:
      i. Fit the alternate hypothesis probability model parameters using $Data_{(S-R_i)}$
      ii. Compute the leave-one-out likelihood $P(R_i \mid H_1(S - \{R_i\}))$.
   (b) Compute the group score, $F(S) = \frac{\prod_{i \in S} P(R_i \mid H_1(S-\{R_i\}))}{\prod_{i \in S} P(R_i \mid H_0)}$.
3. Output the groups with highest score.
4. Perform randomization testing to evaluate the statistical significance of the detected groups.

Figure 4.1: **Anomalous Group Detection Algorithm**

Let us consider a node corresponding to the variable X_m in the Bayes Net. Let X_{Π_m} denote the set of variables corresponding to the parent nodes of X_m. The conditional probability table of X_m has parameters corresponding to the conditional probability values $\theta_{mjk} = P(X_m = j \mid X_{\Pi_m} = k)$. Here we need to estimate θ_{mjk} for each value of m, j and k. To deal with sparsity of the training data, we apply Laplace smoothing to adjust our estimate of each model parameter. We add $\frac{1}{J}$ to each N_{mjk} (the number of instances in the training dataset with $X_m = j$ and $X_{\Pi_m} = k$), where J is the arity of X_m. This makes the total weight of the prior add up to one for each variable X_m and each set of parent values k. The smoothed maximum likelihood estimates of the parameters are given by $\hat{\theta}_{mjk} = \frac{N_{mjk}+1/J}{\sum_{j'}(N_{mj'k}+1/J)}$

In **Steps 2-3**, we wish to find groups of records that maximize the likelihood ratio score $F(S)$. To do so, we search over all possible subsets of the test data. We note that an exhaustive search over all such subsets would require exponential time, but we will describe an efficient heuristic to make this search computationally feasible. For each

subset of the data S, the alternative hypothesis assumes that the records in subset S form an anomalous group.

Step 2(a) of our algorithm computes the pseudo-likelihood of each record under the alternate hypothesis. To compute the pseudo-likelihood, in **Step 2(a)i** we first fit the parameters of the Bayesian Network for the alternative hypothesis $H_1(S - \{R_i\})$. These parameters are estimated from the counts in the subset of the test dataset represented by $S - \{R_i\}$. We follow an approach of smoothed maximum likelihood estimation similar to Step 1 above. In **Step 2(a)ii** we perform inference on the learned alternate hypothesis Bayesian Network model.

Step 2(b) of our algorithm computes the group likelihood ratio score $F(S)$, assuming conditional independence of the records given the models.

Note that Step 2(a) involves $|S|$ iterations of fitting the model parameters and performing inference. In the case of a Bayesian Network model using previously cached counts and a smoothed maximum likelihood estimation of parameters, this step can be done in time independent of the size of the group S. Using the notation from the description of Step 1,

$$P(R_i \mid H_1(S - \{R_i\})) = \prod_m \left[\frac{N_{mjk} + 1/J - 1}{\sum_{j'}(N_{mj'k} + 1/J) - 1} \right]_{\{j=X_m; k=X_{\Pi_m}\}} \tag{4.4}$$

$$P_{pseudo}(Data_S \mid H_1(S)) = \prod_m \prod_k \prod_{j=1}^{J} \left[\frac{N_{mjk} + 1/J - 1}{\sum_{j'}(N_{mj'k} + 1/J) - 1} \right]^{N_{mjk}} \tag{4.5}$$

Here N_{mjk} denotes the corresponding counts in subset of data $Data_S$. Notice that due to the exponentiation term N_{mjk}, this computation can be performed in time proportional to C, the number of non-zero values of N_{mjk} in $Data_S$.

Step 3 of our algorithm outputs the highest scoring groups found in step 2. We use these scores to score the dataset with a measure of anomalousness. We assign the score of the most anomalous group detected as the score of the dataset: $F^*(Data) =$

$\max_{S \in Groups} F(S)$. This is useful for distinguishing between datasets which contain anomalous groups and those without anomalous groups, e.g. distinguishing disease outbreaks from non-outbreak days.

Additionally, to identify individual records which are anomalies, we compute an anomalousness score for each individual record R in the test data, by finding the highest scoring group $S^*(R)$ that contains R. We can then compute the score of record R as $Score(R) = F(S^*(R))$. This gives a high score to any record that is contained in a highly anomalous group, regardless of whether the record is itself anomalous or just similar to other anomalous records.

In **Step 4**, we perform randomization testing to evaluate the statistical significance of the detected groups. To do so, we generate a large number N_{rand} of replica datasets under the null hypothesis that no anomalous groups are present. For each replica, we sample the training data uniformly at random to form a test dataset D_{rand} having the same number of records as the original test dataset, repeat steps 2 and 3 to find the highest scoring groups in the replica dataset, and record the maximum group score $F^*(D_{rand})$. To compute the p-value of a given subset of records S, we can compare the score $F(S)$ (from the original test dataset) to the distribution of maximum group scores from the replica datasets. The p-value is defined as $\frac{N_{beat}+1}{N_{rand}+1}$, where N_{beat} is the number of replica datasets with maximum group scores greater than $F(S)$. Since we are performing the same search procedure (maximization over subsets) for the original dataset and each replica dataset, the randomization testing approach correctly adjusts for the multiple hypothesis tests resulting from maximizing the score over many possible subsets.

We also note that, for a given dataset, the highest scoring subset will have the lowest p-value, and hence the ranking of regions is unchanged by randomization testing. When using the AGD method in practice, we can either choose a p-value threshold, and report all regions with p-values below the threshold, or choose a score threshold, and report all regions S with scores $F(S)$ above the threshold. In our evaluations discussed below, we have plotted the performance of AGD (and four other algorithms) over the entire range of such thresholds, and compared the area under these curves. For this type of evaluation, statistical significance testing by randomization is not necessary.

4.3.2 Search Heuristic

As noted previously, our method calls for searching over all possible subsets of the data. However, an exhaustive search requires exponential time and is thus likely to be computationally infeasible. Instead, we perform an efficient (but approximate) heuristic search in order to speed up the computation. More precisely, we adopt a greedy approach of growing the groups. We grow linearly many groups, starting from each record as an initial seed, and grow the group until no further additions can improve the likelihood ratio score. The algorithm is as follows:

1. Initialize $Groups \leftarrow \{\phi\}$
2. For each record $R_i \in Data_{test}$:
   (a) Initialize $S \leftarrow \{R_i\}$.
   (b) While S has changed over the previous iteration and $size(S) < MaxGroupSize$:
      i. Iterating over each record $R_j \in Data_{test} - Data_S$, find the record that maximizes the score $F(S \cup \{R_j\})$. Let the maximizing record be R_{max}.
      ii. If $F(S \cup \{R_{max}\}) > F(S)$ then set $S = S \cup \{R_{max}\}$; else $Groups = Groups \cup S$.

The anomalousness score of a record R in the test set is then defined as $Score(R) = \max_{S:\, S \in Groups,\, R \in S} F(S)$.

The impact of using the pseudo-likelihood score can be clearly seen during this greedy search procedure. When we use the full-likelihood scoring function as given by eqn. 4.1, overfitting results in an increase of the group score even when a dissimilar record is added to the group. This causes iteration 2(b) to keep adding records to the group until it reaches a size of $MaxGroupSize$. In most cases this results in the addition of many dissimilar records to the group before the iteration stops. The pseudo-likelihood scoring function (eqn. 4.2) helps us avoid this problem. In this case, the group score is increased only by the addition of records that are similar to the existing records within the group.

To evaluate the computational complexity of this search, let us consider a test set of size n. We treat each record as the initial seed and greedily grow the groups to some maximum size G. In our experiments below, we have used $G = 400$. Hence Step 2(b) is repeated at most nG times. We iterate over each record to find the one that best fits the group. Each such comparison can be done in time C, the number of non-zero values of N_{mjk} in $Data_S$. Hence, the overall complexity of the algorithm is $O(n^2GC)$. To make the algorithm efficient, we use a bounding strategy to prune the set of records for which we compute the score $F(S \cup \{R_j\})$ in step 2(b)i. Based on the current best candidate for inclusion in the group, it is possible to compute an upper bound on the null hypothesis likelihood score for any other candidate. Only records that have a null hypothesis likelihood score less than this bound needs to be considered. As we search through the records, we can dynamically update this upper bound based on the current best candidate. In certain cases, it allows us to significantly speedup the computation to determine the best record to add to a group.

4.3.3 Comparison to spatial scan

As noted above, our AGD algorithm can be thought of as a generalization of the spatial scan statistic [Kulldorff, 1997] to arbitrary multivariate datasets without predefined location or count attributes. Here we summarize how the original spatial scan differs from our algorithm described in Figure 4.3.1:

1. The spatial scan searches over a set of contiguous spatial regions that are predefined based on the location attributes of the data, while we perform a heuristic search over arbitrary subsets of the data.

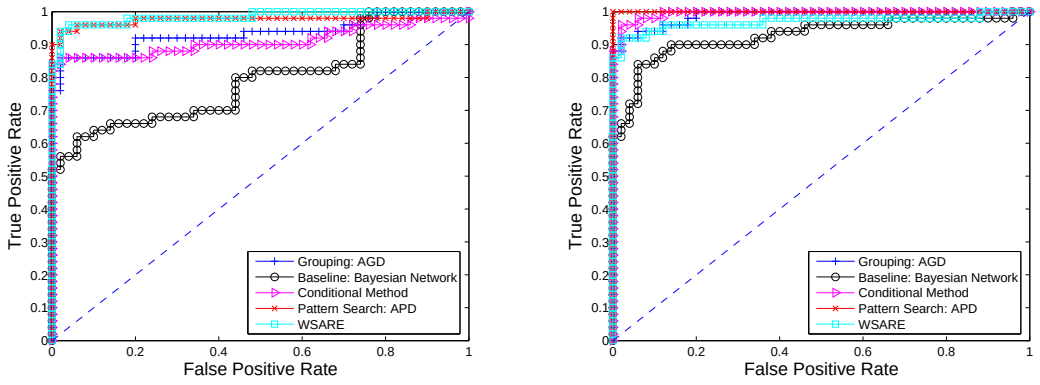
2. In **Step 1**, the spatial scan learns only a single parameter (the uniform incidence rate q_{all}) for the null hypothesis, rather than a probability model relating all variables in the multivariate dataset. Similarly, in **Step 2(a)i**, the spatial scan learns only two parameters (q_{in} and q_{out}) for $H_1(S)$. In **Step 2(a)ii**, it computes the likelihoods under the null and alternative hypotheses using a simple Poisson count model, rather than performing inference on a probability model.

4.4 Datasets

1. PIERS Dataset: Our first dataset consists of records describing containers imported into the country (described in §1.4.1). Since there were no labels in the original data, we create synthetic anomalies by randomly flipping attribute values. We first create a random partition of the dataset into training (100,000 records) and test (1000 records) sets. We modify a random 5% of the test set records to be an anomalous group. To create a group of anomalies G, we first make $Size_G$ identical copies of a randomly chosen record. Each record in the group is then modified by changing the value of up to two randomly chosen attributes. The new values are drawn from the marginal distribution of the corresponding attribute in the training dataset. The records within the group are similar to each other since each pair of records in G differs by at most four attribute values. Each record in the group is anomalous because randomly changing an attribute value breaks the relationship of that attribute with the rest of the attributes. One possible real world scenario where such an anomalous group might occur is when a smuggler smuggles goods using similar methods which have proved successful in the past.

2. Emergency Department Dataset: This real-world dataset contains records of patients visiting Emergency Departments (ED) from hospitals around Allegheny County in the year 2004 (described in §1.4.4). The dataset is injected with simulated ED cases resembling an anthrax release. The simulated cases of anthrax were produced by a state-of-the-art simulator [Hogan et al., 2007] that implements a realistic simulation model of the effects of an airborne anthrax release on the number and spatial distribution of respiratory ED cases. We treat the first two days when the attack symptoms begin to appear as the test data, thus evaluating our ability to detect anthrax attacks within two days of the appearance of symptoms. We train our model on the previous 90 days' data. Note that while we have a model for anthrax release, AGD is not given any information from it. Thus this dataset tests our ability to recognize a realistic, but previously unknown, disease outbreak.

3. KDD Cup 1999 Network Intrusion Detection Dataset: We have also evaluated AGD on the KDD Cup 1999 data [KDDCup, 1999], which contained a wide variety of intrusions simulated in a military network environment (described in §1.4.2). Using all

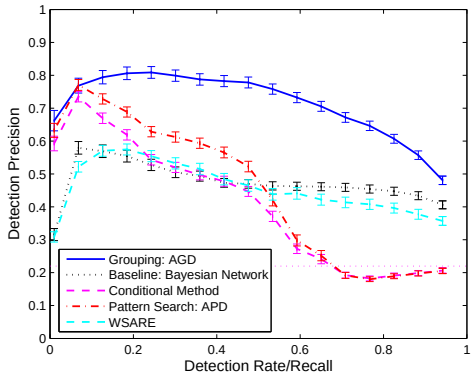


(a) Emergency Department dataset

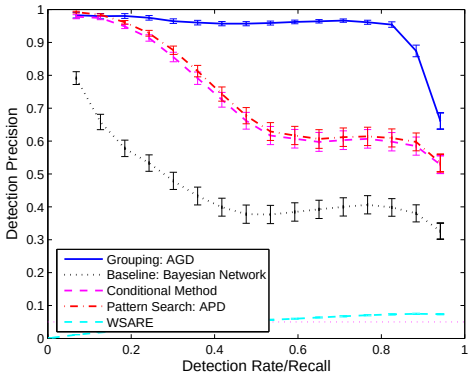
(b) PIERS dataset

Figure 4.2: Algorithm performances for detection of datasets with anomalies

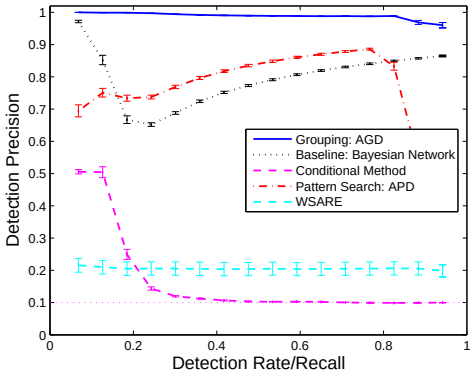
the features in the detection task causes most of the intrusion records to individually stand out from the normal ones as seen in Chapters 2 and 3. Hence, we chose a subset of 22 features that includes the basic features of individual TCP connections and the content features suggested by domain knowledge. This evaluation setup creates groups of self-similar anomalous records that are individually anomalous to a lesser degree. The real valued features were discretized to 5 levels. The goal of the KDD dataset was to produce a good training set for learning methods that use labeled data. Hence, in this case we have labeled anomalies (network attacks) and the proportion of attack instances to normal ones is very large. To create more realistic data, we have reduced the number of attack records to 10% of the test dataset. We have run our algorithms on the 7 most common types of attacks - apache2, guess password, mailbomb, neptune, smurf, snmpguess and warezmaster. Correspondingly, we created seven different test sets containing 10% records of the particular attack type, and 90% normal records. We use the rest of the normal records for training our model.



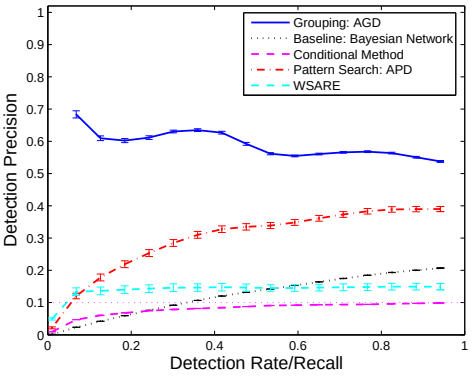
(a) Emergency Department dataset



(b) PIERS dataset



(c) KDD Cup 99: guess password



(d) KDD Cup 99: mailbomb

Figure 4.3: Comparison of detection precision vs. recall for AGD and baseline methods, with standard errors. The dashed line at constant precision is the average performance of the "chance" algorithm that chooses records at random.

Table 4.1: Normalized area under the true positive rate vs. false positive rate curves for AGD and related methods, with standard errors

Dataset	AGD	Bayesian Network	Conditional Method	APD	WSARE
ED	0.932 ± 0.026	0.793 ± 0.041	0.910 ± 0.034	0.976 ± 0.018	**0.984 ± 0.01**
PIERS	0.988 ± 0.006	0.926 ± 0.025	0.994 ± 0.003	**1.0 ± 0.0**	0.970 ± 0.019
apache2	**1.0 ± 0.0**	**1.0 ± 0.0**	**1.0 ± 0.0**	**1.0 ± 0.0**	0.727 ± 0.051
guess passwd	**1.0 ± 0.0**	**1.0 ± 0.0**	0.957 ± 0.016	**1.0 ± 0.0**	0.610 ± 0.045
mailbomb	0.788 ± 0.02	0.82 ± 0.023	0.276 ± 0.036	**0.936 ± 0.03**	0.54 ± 0.048
neptune	**1.0 ± 0.0**	**1.0 ± 0.0**	**1.0 ± 0.0**	**1.0 ± 0.0**	0.695 ± 0.055
smurf	**1.0 ± 0.0**	**1.0 ± 0.0**	0.286 ± 0.031	**1.0 ± 0.0**	0.781 ± 0.048
snmpguess	**1.0 ± 0.0**	0.962 ± 0.023	0.294 ± 0.034	0.935 ± 0.02	0.679 ± 0.052
warezmaster	**1.0 ± 0.0**	**1.0 ± 0.0**	**1.0 ± 0.0**	**1.0 ± 0.0**	0.789 ± 0.042

4.5 Evaluation

We compare the performance of our AGD method to the baseline method, which detects individual records with low likelihoods given the null hypothesis Bayes Net model. In our implementation of the baseline method, we use Optimal Reinsertion [Moore and Wong, 2003] to learn the structure, and perform smoothed maximum likelihood estimation of the network parameters. We also compare the performance to three other related methods discussed in Section 4.2: the Conditional Method [Das and Schneider, 2007], WSARE [Wong et al., 2003], and APD [Das et al., 2008]. We note that the better-performing of the two individual anomaly detectors was used to detect individually anomalous records for APD on each dataset (i.e. we used the Conditional Method for the ED and PIERS datasets, and the Bayes Net method for the KDD Cup datasets).

The procedure for randomly generating the test data and injecting anomalous groups in them was repeated 50 times for each of the nine experiments (ED, PIERS, and seven different KDD Cup attack types). For each experiment, we also produced 50 additional sets of test data (of the same size) with no anomalies injected. These runs are helpful in

Table 4.2: Area under the detection precision vs. recall curves for AGD and related methods, with standard errors

Dataset	AGD	Bayesian Network	Conditional Method	APD	WSARE
ED	**0.729 ± 0.032**	0.479 ± 0.027	0.375 ± 0.026	0.420 ± 0.027	0.465 ± 0.033
PIERS	**0.957 ± 0.014**	0.429 ± 0.053	0.706 ± 0.045	0.720 ± 0.043	0.053 ± 0.003
apache2	**1.0 ± 0.0**	0.973 ± 0.003	0.951 ± 0.004	0.882 ± 0.021	0.215 ± 0.042
guess passwd	**0.991 ± 0.002**	0.773 ± 0.008	0.124 ± 0.005	0.804 ± 0.013	0.205 ± 0.041
mailbomb	**0.587 ± 0.007**	0.136 ± 0.001	0.086 ± 0.001	0.329 ± 0.019	0.146 ± 0.022
neptune	0.993 ± 0.002	0.984 ± 0.003	**1.0 ± 0.0**	0.986 ± 0.003	0.217 ± 0.030
smurf	**0.974 ± 0.003**	0.640 ± 0.006	0.089 ± 0.001	0.889 ± 0.015	0.237 ± 0.032
snmpguess	**0.987 ± 0.002**	0.288 ± 0.002	0.087 ± 0.001	0.521 ± 0.030	0.266 ± 0.034
warezmaster	**0.892 ± 0.014**	0.852 ± 0.009	0.430 ± 0.014	0.677 ± 0.034	0.141 ± 0.021

determining the ability of the algorithms to differentiate between entire datasets containing anomalous groups and those without anomalous groups.

We evaluate the performance of the algorithms in two different ways. First, we examine the ability of the algorithms to identify and distinguish between entire test datasets which have anomalous groups against ones which are normal (i.e. do not have any anomalies). In the Emergency Department data, for example, this corresponds to distinguishing between an anthrax attack occurring and no attack occurring. As noted above, the algorithms are run over 100 test datasets, where half of these datasets contain injected anomalies.

For the three methods that explicitly search over sets of records, the dataset score is set as the score of the most anomalous group (AGD), pattern (APD), or rule (WSARE) detected. For two methods that score records individually (the baseline Bayesian Network method and the Conditional Method), the dataset score is calculated as the sum of the individual scores of all the records. Note that since these methods do not model groups of anomalies, summing up the individual record scores (as opposed to considering the single most anomalous record score) gives significantly better detection performance. We

then examine each method's tradeoff between its false positive rate (proportion of datasets without anomalies that were falsely detected as being anomalous) and its true positive rate (proportion of datasets with anomalies that were correctly detected as being anomalous). This is the standard ROC curve: a higher curve denotes better detection performance, since it corresponds to a higher true positive rate for a given false positive rate. The area under the ROC curve (AUC) can be used as a summary measure, where higher AUC corresponds to better average performance.

For the 50 datasets that contain anomalies, we also evaluate the ability of each algorithm to identify which individual records were anomalous. For example, in the Emergency Department dataset, this corresponds to identifying which patients have been affected by the anthrax attack and which patients are in the Emergency Department due to other causes. We plot the detection precision, i.e. the ratio of number of true positives to the total number of predicted positives, against the detection rate, i.e. the proportion of total true anomalies that are detected. The plots are generated by varying the threshold used to flag anomalies. The standard error estimates are also shown in the plots. Here, a higher curve denotes better performance, since it corresponds to a higher detection precision for a given detection rate.

The Bayesian Network method and the Conditional Method assign a anomalousness score to each individual record which can be directly used to perform this evaluation. For the rest of the methods, the score of a record is assigned as the score of the most anomalous group (AGD), pattern (APD) or rule (WSARE) that it belongs to.

4.6 Results

We first examine the performances of the algorithms in differentiating between test datasets that contain anomalous groups and datasets without injected anomalies. Figures 4.2(a) and 4.2(b) show the ROC curves for the ED and PIERS datasets respectively, and Table 4.1 shows the area under the ROC curve (AUC) for all nine experiments (ED, PIERS, and the seven attack types for KDD Cup).

We can see that, for both the ED and PIERS datasets, AGD performs better than the baseline Bayesian Network method, having a greater true positive rate for a given false positive rate. The AGD method has significantly larger area under the curve than the baseline method (using a paired t-test, $\alpha = 0.05$) in both cases. However in both these cases we observe that APD performs better than AGD. For the KDD Cup network intrusion dataset, AGD is able to perfectly differentiate the datasets (i.e., has a true positive rate = 1 for all false positive rates) for all attack types except mailbomb. AGD also performs well across all nine experiments as compared to APD, WSARE, and the Conditional Method. However, we see that for the Emergency Department data, WSARE gives us the best performance. This is not very surprising, since WSARE was originally developed to detect outbreaks among patients admitted to Emergency Departments, and WSARE performs relatively poorly for the other experiments. APD performs similarly to AGD for this evaluation, but as we demonstrate below, AGD performs substantially better on identifying anomalous records.

Next, we look at the performances of the algorithms in identifying anomalous records in the datasets that contain anomalies. Figure 3 shows the relative performance of the five methods on the ED and PIERS datasets, as well as two of the seven KDD Cup experiments (guess password and mailbomb). In all of the plots, the baseline performance of randomly choosing which records are anomalous is shown by a dashed line. Table 4.2 gives the normalized area under the curve for the detection rate interval [0.1, 0.9], with standard errors, for each method on all nine experiments.

We see that AGD performed significantly better than the baseline Bayes Net method for all nine experiments, demonstrating that using the group information substantially improves our ability to detect which records are anomalous. On eight of the nine experiments, AGD also performed significantly better than the three related methods (APD, WSARE, and the Conditional Method). The one exception was the KDD Cup neptune attack, where the Conditional Method achieved perfect performance (AUC = 1) while AGD achieved near-perfect performance (AUC = 0.993). All differences in AUCs between the best method (performance shown in bold font) and second-best method were found to be significant at $\alpha = 0.05$.

Table 4.3: Comparison of AUCs for precision vs. recall plots for AGD and APD on datasets with different group sizes and group self-similarity

Method	Type of injected anomalies	50	25	10
AGD	Self-similar group	**0.957 ± 0.014**	**0.581 ± 0.051**	0.524 ± 0.058
	Individual anomalies	0.300 ± 0.017	0.183 ± 0.015	0.089 ± 0.015
APD	Self-similar group	0.720 ± 0.043	**0.581 ± 0.046**	**0.638 ± 0.054**
	Individual anomalies	**0.471 ± 0.025**	**0.396 ± 0.031**	**0.268 ± 0.035**

We also evaluate how the size and self-similarity of the anomalous groups impact the relative performance of detection methods APD and AGD. Table 4.3 gives the AUC for the task of identifying anomalous records in the PIERS dataset. We vary the size of the group ($Size_G = 50, 25, 10$) by choosing different proportion of test records to be modified as anomalies. We also evaluate the performance of both the algorithms when the anomalous records are not self-similar (randomly chosed records are modified to be anomalous). We see that AGD performs best when the anomalous groups are larger and more self-similar, while APD outperforms AGD for detecting smaller groups where each individual record in the group is anomalous. When the anomalies do not form a self-similar group, AGD performs very poorly.

In the extreme case of a group consisting of a single, highly anomalous record, we would expect individual anomaly detection methods such as the Conditional Method to outperform both APD and AGD.

4.7 Discussion

We note that, instead of using the maximum likelihood estimates of the parameters in §4.3, we have also explored the use of a Bayesian approach. In this approach, we consider a Dirichlet prior distribution over the parameters, and compute the marginal likelihood of the data as the score function $F(S)$. From a theoretical standpoint, it would seem that this approach might lessen the effect of overfitting while computing the likelihood under

the alternate hypothesis. However, our preliminary empirical results indicate that using the marginal likelihood scoring function is not very effective at addressing overfitting, and the resulting groups still grow without bound (as was the case for the original maximum likelihood approach, motivating our use of the pseudo-likelihood). When we consider the marginal likelihood approach, the pseudo-likelihood is no longer well defined, since the likelihood of the data can no longer be expressed as a product of the individual record likelihoods when integrated over the multinomial parameters. Hence we chose to use the maximum likelihood, rather than marginal likelihood, estimates of the multinomial parameters in our pseudo-likelihood score function.

4.8 Conclusions

In this work we describe a method of generalizing likelihood based anomaly detection (using Bayesian Networks) by integrating the information about *groups* of anomalous records. We evaluate the methods on three real-world datasets, injected with simulated and real anomalies. The performance is evaluated for the tasks of detecting individual anomalous records and distinguishing between datasets having or not having anomalous groups. The Anomalous Group Detection method gives significantly better detection performance over the baseline method for both of these tasks. Additionally, under certain assumptions, AGD is shown to outperform three previously proposed methods (WSARE, APD, and the Conditional Method), substantially improving the identification of anomalous records for all three datasets.

Chapter 5

Detecting Spatio-Temporal Patterns

5.1 Introduction

So far, we have considered the unsupervised problem of detecting any behavior that is different from the normal or usual behavior of the system. We now focus on the semi-supervised case of detecting of certain given types of *events* that affects the data in characteristic patterns. In particular we are interested in characterizing and detecting specific spatio-temporal patterns which correspond to certain event types in datasets with space and time components.

The Spatial scan statistic (described in §1.3.2) is a popular method used to detect spatial clusters of increased counts. It has also been extended to include the time domain to scan over space-time regions [Kulldorff, 2001, Neill et al., 2005c, Neill and Cooper, 2009]. These methods generally assume a frequentist framework, and perform maximum likelihood fitting of the null and alternate hypotheses parameters. They assume a uniform increase of counts (e.g. number of emergency department patients or over the counter sales of medicines in the affected region) over the affected space-time region, and that the affected region does not change with time, over the given time window. These simplifying assumptions can be too restrictive for some types of events. By limiting the effects to be constant over the time window, it is not able to model events that have both increas-

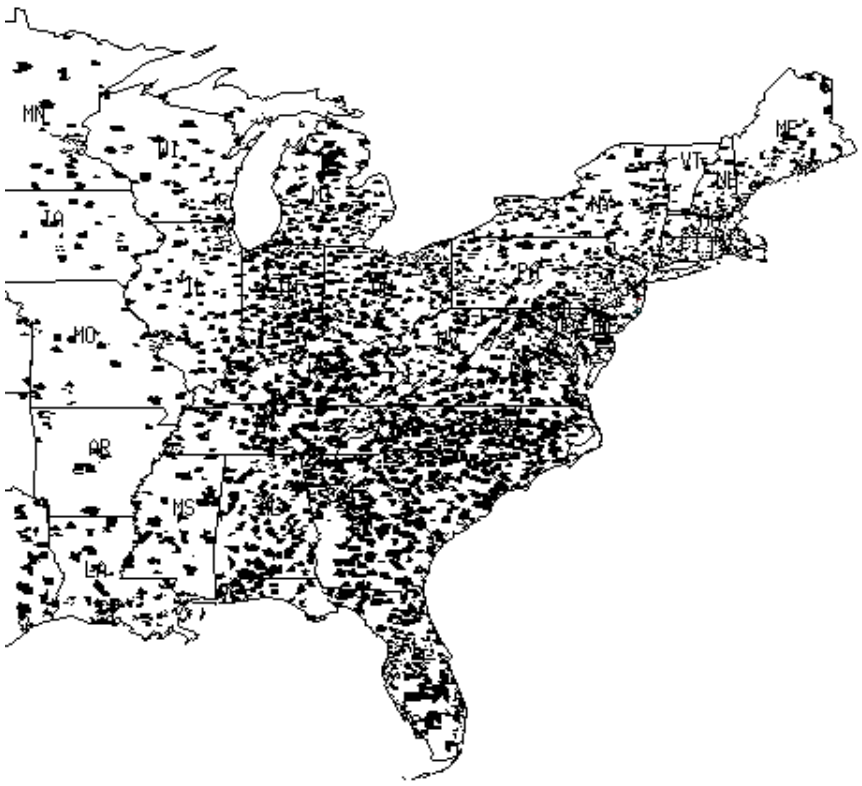


Figure 5.1: Data coverage of the OTC sales data in the eastern part of United States

ing and decreasing effects over time. Additionally, it is not able to differentiate between event types that have different rates of increase (or decrease) of counts. The affected region might expand or contract with time, or more significantly the effect might move to different regions with time.

In this chapter, our main contribution is the generalization of spatio-temporal scan statistics to detect time varying patterns, learned in a semi-supervised framework. We consider two time varying effects. First, the counts at a particular location affected by an event can have a time dependent variation pattern (for example the pattern shown in Figure 5.3) and second, the event can affect distinct spatial regions over time. An example of such an event type is described below.

We have data of the daily counts of sale of eight different categories of over the counter sales of medicines in stores throughout the US. The data is aggregated at the zip-code level due to privacy concerns. Figure 5.1 shows the data coverage over zip codes in the eastern part of United States. Zip codes for which data is available are shown in black. The obvious use of this data is to detect outbreaks of diseases as investigated in [Goldenberg et al., 2002]. The main limitation in this kind of evaluation is that we do not have any labeled disease outbreaks in this data. Hence, all the evaluations depend on artificial injects of outbreaks. We are interested in events that have a time-varying effect on the counts, and affects a spatial region that moves with time. Our observation of the data indicated that there are significant patterns in the data corresponding to inclement weather. Specifically, a region that is hit by a hurricane exhibits a characteristic temporal pattern of OTC sales (shown in Figure 5.3). Since hurricane warnings are quite accurate, we see a sudden increase in OTC sales just before the hurricane strike. This is followed by a very significant decrease of sales on the day the hurricane passes over the region. Presumably this is because very few people venture out of their homes in such extreme weather. The number of sales again comes back to normal within a few days. Also, the hurricane moves with time (shown in Figure 5.2) and affects different spatial regions over time. Note that different areas may be affected at different start times but then undergo the same characteristic temporal pattern: Figure 5.3 shows an example where some locations' temporal trends (x^0) lead others (x_1) by one day. The National Hurricane Center [NHC] has a comprehensive archive of past hurricane strikes affecting locations within the country. Because of the lack of accurate labels for any other event type in this data, we use the example of hurricane strikes to evaluate our algorithm. We assume a semi-supervised framework, where a small set of locations are labeled by the user as belonging to a particular event type. Our algorithm aims to use this partial labeling to learn a model for the event, and detect other occurrences of similar events in the data. Although we evaluate our methods on examples of hurricane strikes present in the data, this method can be useful in more practical detection problems. For example, in case of an airborne anthrax release [Hogan et al., 2007], we can track the effect of the outbreak as it spreads to different regions. Also, it is possible to detect and differentiate between different disease types based on the rate at which it affects a population.

A Bayesian version of the scan statistics is presented in [Neill et al., 2005b] and [Makatchev and Neill, 2008], where the null and alternate hypotheses are modeled by a hierarchical Bayesian structure. A further generalization of this method is the Multivariate Bayesian Scan Statistics (MBSS) framework presented in [Neill et al., 2007] and [Neill and Cooper, 2009], aimed at detecting specific event types in multivariate spatio-temporal data. Events are characterized by their effect on counts of observation in a particular space-time region. We note that MBSS assumes that an event causes a constant factor increase in the counts and that the affected region remains constant over the relevant time window.

The main contributing ideas in this chapter (incorporating time-varying effects and spatially dependent lags into the event detection setting) can be viewed as a generalization of space-time scan statistics, orthogonal to the Bayesian extension. For example, [Neill et al., 2005a] present a simple frequentist extension of the space-time scan statistic where the effect increases monotonically over time and [Jiang et al., 2008] considers a linear increase over time. However, since MBSS is currently one of the most general methods for space-time scan, we implement our ideas of detecting time varying pattern as a generalization of this framework. Our proposed method: Time Varying Multivariate Bayesian Scan Statistics (TV-MBSS) is able to model these time varying effects in the MBSS framework. Therefore, we start with a detailed description of the MBSS framework. We then describe our contribution: the two time varying effects that generalize this framework. We also present a semi-supervised learning algorithm to learn the time varying effects of an event. We show the evaluation of our algorithm on the example of hurricanes strikes, and compare the performance with that of MBSS.

5.2 Multivariate Bayesian Scan Statistics

This section gives a detailed description of the MBSS method and is adapted from [Neill and Cooper, 2009].

We are given a dataset D consisting of multiple data streams D_m, for $m = 1 \dots M$. Each data stream consists of spatial time series data collected at a set of spatial locations

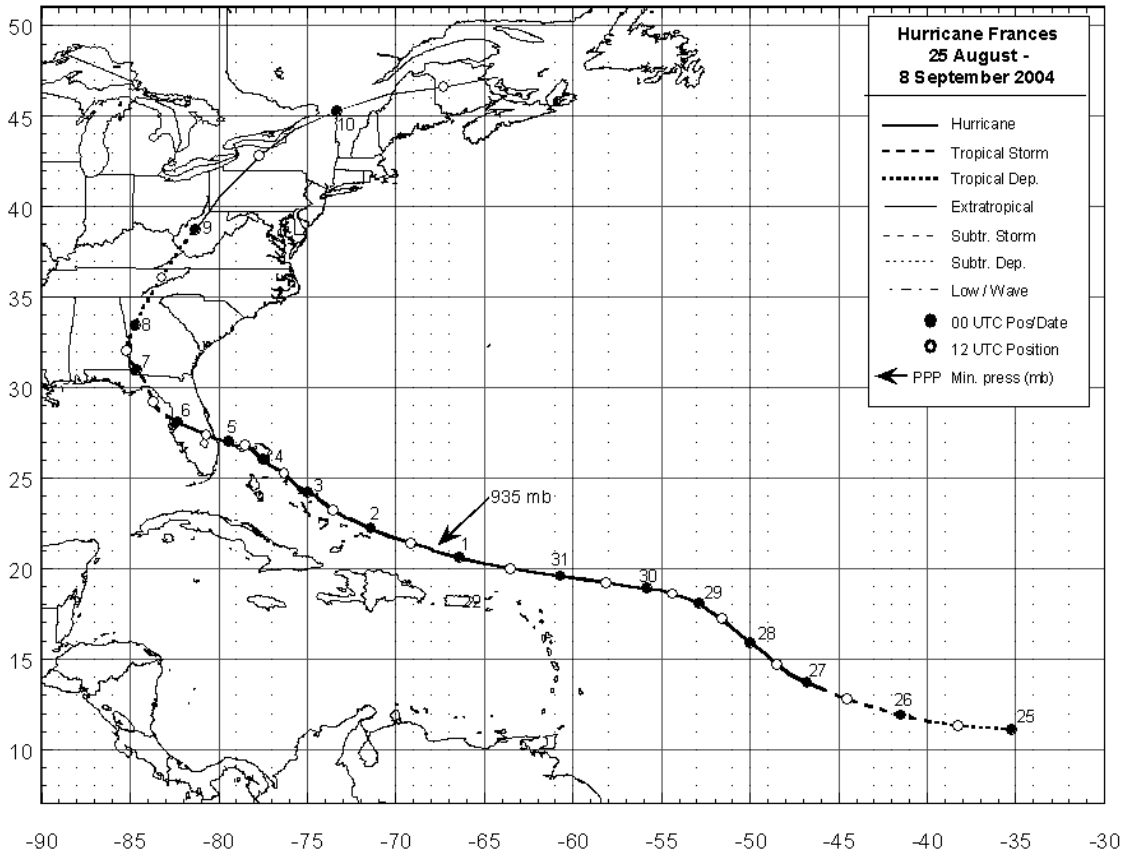


Figure 5.2: Best track positions of Hurricane Frances, Aug-Sept 2004

s_i, for $i = 1 \ldots I$. For each stream D_m and location s_i, we have a time series of counts $c^t_{i,m}$, for $t = 0 \ldots T$. For example, in disease surveillance, we typically have data collected on a daily basis, and aggregated at the zip code level due to data privacy concerns. Thus a given count $c^t_{i,m}$ might represent the number of respiratory emergency department visits, or the number of cough/cold drugs sold, for a given zip code on a given day.

As noted by [Neill and Cooper, 2009], the goals of the MBSS framework are event detection and characterization: to detect any relevant events occurring in the data, identify the type of event, and determine the event duration and affected locations. Thus it needs to compare the set of alternative hypotheses $H_1(S, E_k)$, each representing the occurrence of

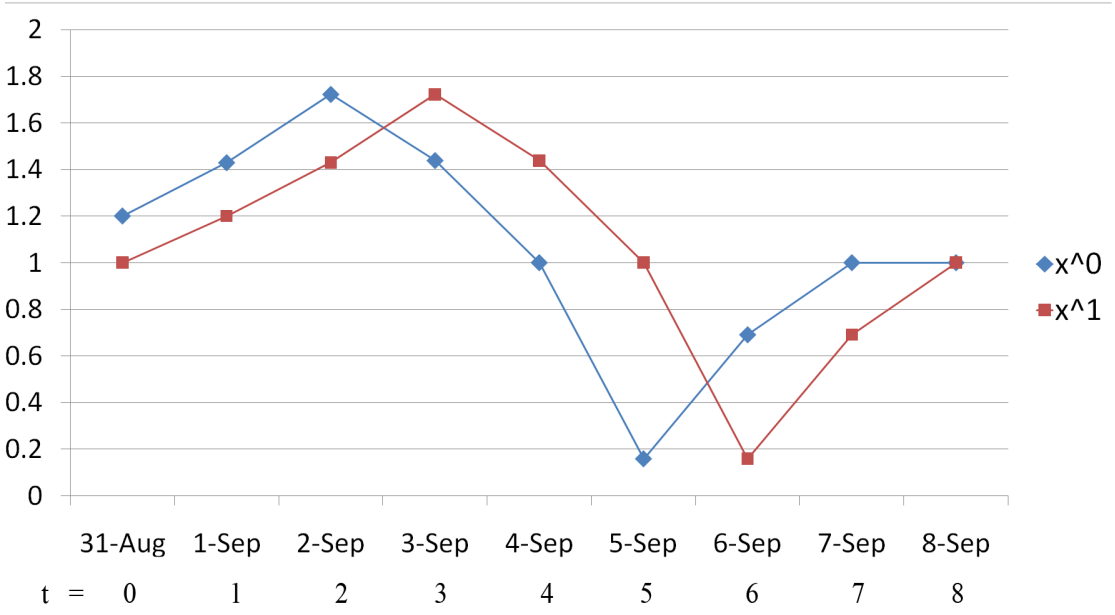


Figure 5.3: Temporal effects pattern on OTC medicine sales corresponding to a hurricane strike (Frances)

some event of type E_k in some space-time region S, against the null hypothesis H_0 that no events have occurred. It is assumed that the set of event types $E = \{E_k\}$, for $k = 1 \ldots K$, is given, and that these events are mutually exclusive (i.e. at most one event occurs in the data). Moreover, each distinct hypothesis $H_1(S, E_k)$ assumes that the given event type E_k has affected all and only those locations $s_i \in S$, and thus all hypotheses $H_1(S, E_k)$ are mutually exclusive.

In the Bayesian framework, the goal is to compute the posterior probability $\text{Pr}(H_1(S, E_k)|D)$ that each event type E_k has affected each space-time region S, as well as the posterior probability $\text{Pr}(H_0 \mid D)$ that no event has occurred. Applying Bayes' Theorem to compute the posterior probability of each hypothesis:

$$\text{Pr}(H_1(S, E_k) \mid D) = \frac{\text{Pr}(D \mid H_1(S, E_k))\text{Pr}(H_1(S, E_k))}{\text{Pr}(D)} \tag{5.1}$$

$$\Pr(H_0 \mid D) = \frac{\Pr(D \mid H_0)\Pr(H_0)}{\Pr(D)} \tag{5.2}$$

In this expression, the posterior probability of each hypothesis is normalized by the total probability of the data, $\Pr(D) = \Pr(D|H_0)\Pr(H_0) + \sum_{S,E_k} \Pr(D|H_1(S,E_k))\Pr(H_1(S,E_k))$.

Since the goal is to compare the posterior probability values for each event type and the null hypothesis, we can use the following posterior likelihood ratio:

$$\frac{\Pr(H_1(S,E_k) \mid D)}{\Pr(H_0 \mid D)} = \frac{\Pr(D \mid H_1(S,E_k))}{\Pr(D \mid H_0)} \frac{\Pr(H_1(S,E_k)}{\Pr(H_0)} \tag{5.3}$$

This formulation is useful in practice since it is easier to compute the likelihood ratio terms than computing the actual likelihood values. Thus, this ratio of posterior probabilities is used as a scoring metric for comparison.

Finally, the posterior probability that a location s_i is affected by the event E_k is given by:

$$\Pr(H_1(s_i, E_k \mid D)) = \sum_{S:s_i \in S} \Pr(H_1(S,E_k) \mid D) \tag{5.4}$$

In the following sections, we consider how the priors $\Pr(H)$ and the likelihoods $\Pr(D|H)$ can be computed for each hypothesis under consideration.

Each prior probability $\Pr(H_1(S,E_k))$ can be decomposed as the product of the prior probability of event type E_k and the conditional probability that subset S is affected by E_k: $\Pr(H_1(S,E_k)) = \Pr(E_k)\Pr(H_1(S,E_k) \mid E_k)$. In this expression, $\Pr(E_k)$ represents the overall prevalence of event type E_k, while $\Pr(H_1(S,E_k) \mid E_k)$ represents its distribution in space and time. As noted above, we assume that all event types are mutually exclusive, so that $\Pr(H_0) + \sum_k \Pr(E_k) = 1$. We also assume that each event only affects a single space-time region S, so that $\sum_S \Pr(H_1(S,E_k) \mid E_k) = 1$ for each event type E_k.

We assume a uniform prior over event types. In this case, we have $\Pr(H_0) = 0.99$, and $\Pr(E_k) = \frac{0.01}{K}$ for all $k = 1 \ldots K$. Similarly, for the distribution of a given event type over regions S, we assume a uniform region prior $\Pr(H_1(S,E_k) \mid E_k) = \frac{1}{N_S}$, where N_S is the total number of space-time regions. Thus we have prior probabilities $\Pr(H_1(S,E_k)) =$

$\frac{0.01}{KN_S}$ for all S and E_k under consideration. Alternatively, if we have a sufficient amount of labeled training data, these prior values can be learned from it.

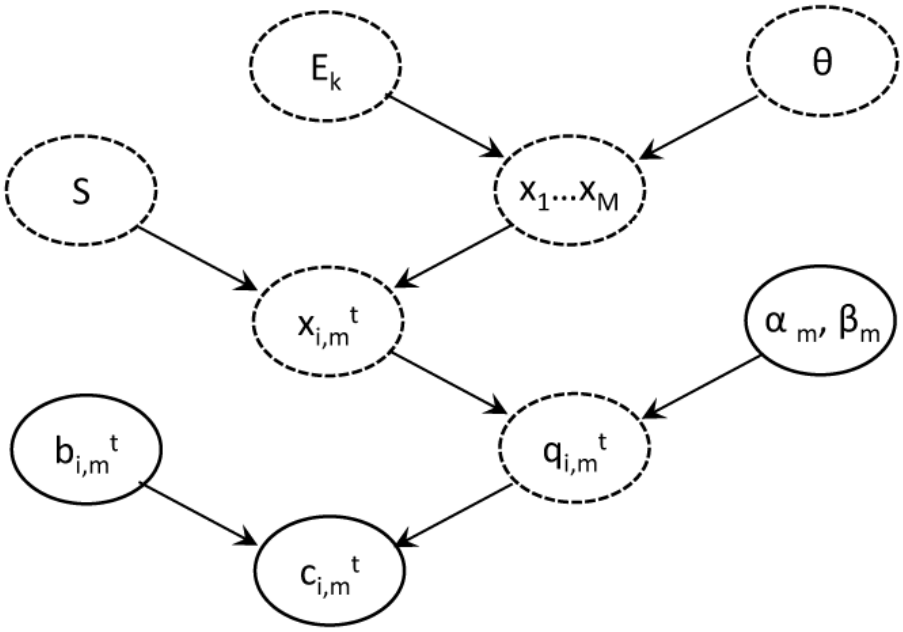


Figure 5.4: Bayesian network representation of the MBSS method. Solid ovals represent observed quantities, and dashed ovals represent hidden quantities

5.2.1 Likelihood computation

The counts $c_{i,m}^t$ are assumed to be generated by an hierarchical Gamma Poisson model as shown in Figure 5.4. The event type k is drawn from a multinomial distribution. The region of effect S depends on the event type. The effects of an event $H_1(S, E_k)$ are determined by $x_{i,m}^t$ for each location s_i, data stream D_m, and time step t. These effects are multiplicative and increase the value of each count $c_{i,m}^t$ by a factor of $x_{i,m}^t$, and thus $x_{i,m}^t = 1$ would signify no effect of the event for the given location, data stream and time step. For

the null hypothesis H_0, $x^t_{i,m}$ is assumed to be 1 everywhere. For an event $H_1(S, E_k)$, the effects are assumed to be constant within all the locations $s_i \in S$ and over the time window of the affected space time region. In this case, the effects can be represented by a vector $x = (x_1 \dots x_M)$ representing the effects on the event on each data stream D_m. The data likelihood for the event can then be represented as the marginal probability value:

$$\Pr(D \mid H_1(S, E_k)) = \sum_X \Pr(D \mid X) \Pr(X \mid H_1(S, E_k)) \tag{5.5}$$

The relative risk $q^t_{i,m}$ has a Gamma distribution with parameters $\{x^t_{i,m}\alpha_m, \beta_m\}$. The count $c^t_{i,m}$ is drawn from a Poisson distribution with mean $q^t_{i,m} b^t_{i,m}$. Here the parameter priors α_m and β_m are learned from the data using a "parametric empirical Bayes" procedure as described in [Neill and Cooper, 2009]. $b^t_{i,m}$ is the baseline counts that represents the expected value of $c^t_{i,m}$ assuming that no events are taking place, and is learned from time series analysis of historical data. The method used here is a 28 day moving average value, adjusted for day of the week effect.

Marginalizing over the values of the relative risk, we see that each count $c^t_{i,m}$ follows a negative binomial distribution with parameters $x^t_{i,m}\alpha_m$ and $\frac{\beta_m}{\beta_m + b^t_{i,m}}$. Since the counts are conditionally independent given the values of $b^t_{i,m}$, $x^t_{i,m}$, α_m, and β_m, the likelihood of the entire dataset $D = \{c^t_{i,m}\}$ for a given set of effects $X = \{x^t_{i,m}\}$ is the product of these conditional probabilities:

$$\begin{aligned} \Pr(D \mid X) &= \prod_{i,m,t} \Pr(c^t_{i,m} \mid b^t_{i,m}, x^t_{i,m}, \alpha_m, \beta_m) \\ &\propto \prod_{i,m,t} \left(\frac{\beta_m}{\beta_m + b^t_{i,m}} \right)^{x^t_{i,m}\alpha_m} \frac{\Gamma(x^t_{i,m}\alpha_m + c^t_{i,m})}{\Gamma(x^t_{i,m}\alpha_m)} \end{aligned} \tag{5.6}$$

In this expression, terms not dependent on the $x^t_{i,m}$ have been removed, since these are constant for all hypotheses under consideration. For the null hypothesis H_0, we have $x^t_{i,m} = 1$ everywhere:

$$\Pr(D \mid H_0) \propto \prod_{i,m,t} \left(\frac{\beta_m}{\beta_m + b^t_{i,m}} \right)^{\alpha_m} \frac{\Gamma(\alpha_m + c^t_{i,m})}{\Gamma(\alpha_m)} \tag{5.7}$$

The probability distribution $\Pr(X \mid H_1(S, E_k))$ is modeled as follows. The model is parameterized in terms of the average effects $\bar{x}_{k,m}$ of each event type E_k on each data stream D_m and the event magnitude θ:

$$x_m^\theta = 1 + \theta(\bar{x}_{k,m} - 1) \tag{5.8}$$

A fixed discrete distribution for θ is assumed, mixing uniformly over $\theta \in \{\frac{1}{4}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}, 1, \frac{3}{2}, 2, 3, 4\}$. The average effects, $\bar{x}_{k,m}$, can either be specified by a domain expert or learned from the training data.

Using equations 5.5, 5.6, 5.7 and 5.8 the likelihood ratio can be expressed as:

$$\frac{\Pr(D \mid H_1(S, E_k))}{\Pr(D \mid H_0)} = \sum_\theta \Pr(\theta \mid E_k) \prod_{i,m,t \in S} \frac{\Pr(c_{i,m}^t \mid b_{i,m}^t, x_m^\theta \alpha_m, \beta_m)}{\Pr(c_{i,m}^t \mid b_{i,m}^t, \alpha_m, \beta_m)}$$

$$= \sum_\theta \Pr(\theta \mid E_k) \prod_{i,m,t \in S} \left(\frac{\beta_m}{\beta_m + b_{i,m}^t} \right)^{(x_m^\theta - 1)\alpha_m} \frac{\Gamma(\alpha_m)\, \Gamma(x_m^\theta \alpha_m + c_{i,m}^t)}{\Gamma(x_m^\theta \alpha_m)\, \Gamma(\alpha_m + c_{i,m}^t)}$$

$$= \sum_\theta \Pr(\theta \mid E_k) \prod_{i,t \in S} \prod_m LR_{i,m}^{t,\theta} \tag{5.9}$$

For a each location s_i, time step t and event magnitude θ, a log-likelihood ratio value is precomputed:

$$LLR_i^{t,\theta} = \sum_m \log LR_{i,m}^{t,\theta} \tag{5.10}$$

The likelihood ratio for any region S (eqn. 5.9) can then be computed as a sum over all the log-likelihood values with location, time step belonging in S. This gives the advantage that the expensive log-likelihood computations are only performed a number of times proportional to the number of locations, rather than the (much larger) number of regions.

5.3 Time Varying Multivariate Bayesian Scan Statistics

We now present our method of Time Varying Multivariate Bayesian Scan Statistics (TV-MBSS). The time varying pattern detection can be implemented as an extension to either

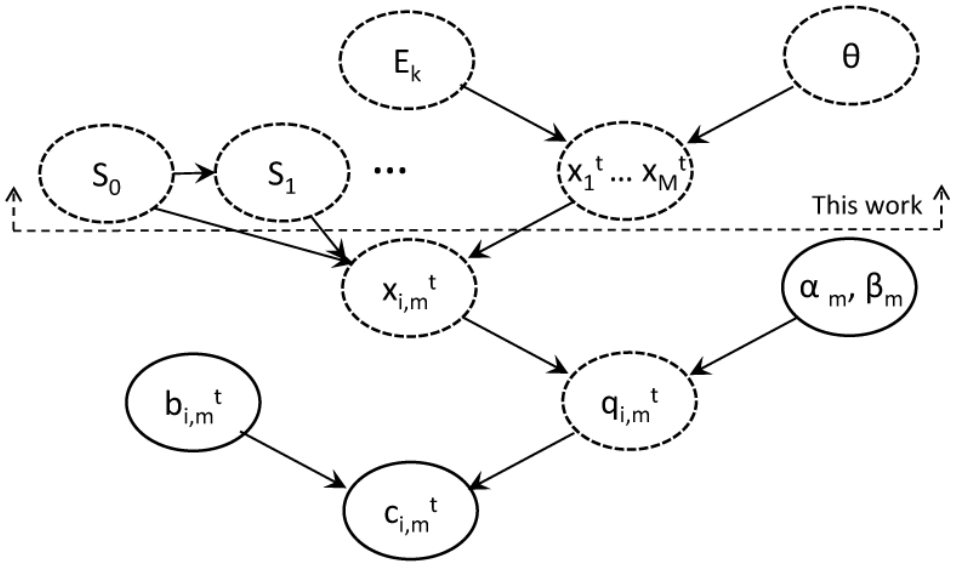


Figure 5.5: Bayesian network representation of the TV-MBSS method. Solid ovals represent observed quantities, and dashed ovals represent hidden quantities

of the scan statistics formulation, the frequentist and the Bayesian. In this work, we implement this as an extension of the MBSS framework.

MBSS makes two simplifying assumptions. The effects $x^t_{i,m}$ of an event $H_1(S, E_k)$ are assumed to be constant over the time window and over all the locations in S. Then $x^t_{i,m} = x_m$ for all $s_i \in S$ and $0 \leq t < -W(S)$, and $x^t_{i,m} = 1$ otherwise. Here $W(S)$ is the time duration of the space time region S. We relax both of these assumptions. Figure 5.5 shows the Bayesian network representation of the TV-MBSS method. The two differences from the original MBSS model (Figure 5.4) are:

1. We now have M time-series patterns $\{x^t_1 \ldots x^t_m\}$ instead of just a vector $\{x_1 \ldots x_m\}$, characterizing the effects of an event type.

2. Instead of a single space-time region S, we now have two (or more) space-time sub-regions denoted by S_0, S_1 .... The subsequent sub-region is dependent on the previous sub-region (they are constrained to be adjacent).

Apart from these differences, we assume the same hierarchical Gamma Poisson model as MBSS. Below we describe how the time varying effects are integrated into this framework.

5.3.1 Time Series Patterns

We now consider the first time varying generalization that an event type can have a time varying pattern of effects at a particular location. We begin by assuming a constant effect over all the locations within the affected region S at a given time step t, i.e. $x^t_{i,m} = x^t_m$ for all $s_i \in S$. Hence, for a given data stream D_m, $\{x^t_m : t = 0 \ldots (W-1)\}$ forms a time-series (e.g. Figure 5.3 gives the time series pattern for a hurricane strike).

Under this assumption, to model an event E_k, we need to specify the average effects $\bar{x}^t_m$, which can be viewed as M time-series patterns of duration W, each corresponding to a data stream D_m. Here we assume W is the length of duration over which the event has an effect on a particular location. The probability distribution $\Pr(X \mid H_1(S, E_k))$ is parameterized in terms of the average effects on each data stream and the event magnitude θ as given by eqn. 5.8.

In out implementation, we assume that the set of regions $\mathcal{S}$ contains all possible circular regions centered at a data location, containing up to $Size_{max}$ locations. $Size_{max}$ is pre-specified by the user, based on expert knowledge about the maximum possible geographical spread of any event.

While this method (which we call TV-MBSS-1) is able to model the time varying nature of the effects at a particular location, it assumes that an event affects all the locations in S synchronously. It cannot model an event that can move with time, and affect geographically adjacent regions with a time lag. For example, when we apply this method to detect hurricane patterns in the OTC sales data, it fails to identify all the locations affected

Figure 5.6: MBSS-TV-1 Detect Algorithm

Parameters: $\mathcal{S}$, For each event type E_k effects $\{x_m^t : m = 1 \ldots M;\ t = 0 \ldots (W-1)\}$

1. Compute the baselines $b_{i,m}^t$ and the parameter priors (α_m, β_m) from historical data, as in the original MBSS algorithm.

2. Compute the log-likelihood ratios $LLR_{i,k}^{\theta}$ corresponding to each location s_i, event type E_k and intensity level θ aggregated over the M data streams and over the time window $t = 0 \ldots (W-1)$; as given by eqn. 5.10.

3. For each $S \in \mathcal{S}$ and each event type E_k, compute the posterior log-likelihood ratio $\frac{\Pr(H_1(S,E_k)|D)}{\Pr(H_0|D)}$, as given by eqn. 5.9.

4. For each location s_i and event type E_k, compute the posterior probability $P(H_1(s_i, E_k)|D)$, that s_i is affected by the event E_k, as given by eqn. 5.4.

by the hurricane, since areas that are farther inland are affected with a time lag of one (or more) days as compared to areas on the coast. Figure 5.7 shows the set of locations identified using TV-MBSS-1, and as we can see, it can only identify a small subset of all the locations affected (as shown in Figure 5.13). To account for this non-stationary nature of the events, we propose another extension of the algorithm in the next section.

5.3.2 Modeling Nonstationary Events

Here we assume that the event moves over time and can affect different regions at different time steps. For ease of presentation, we assume a single event type E_k for the rest of this discussion. Due to computational and practical considerations, we consider up to two distinct, adjacent but non-overlapping regions that are affected on consecutive time steps by the event. Let $t = 0$ represent the time step at which the effects of an event onsets on the overall data and S_t denote the set of locations where the effect of the event onsets at

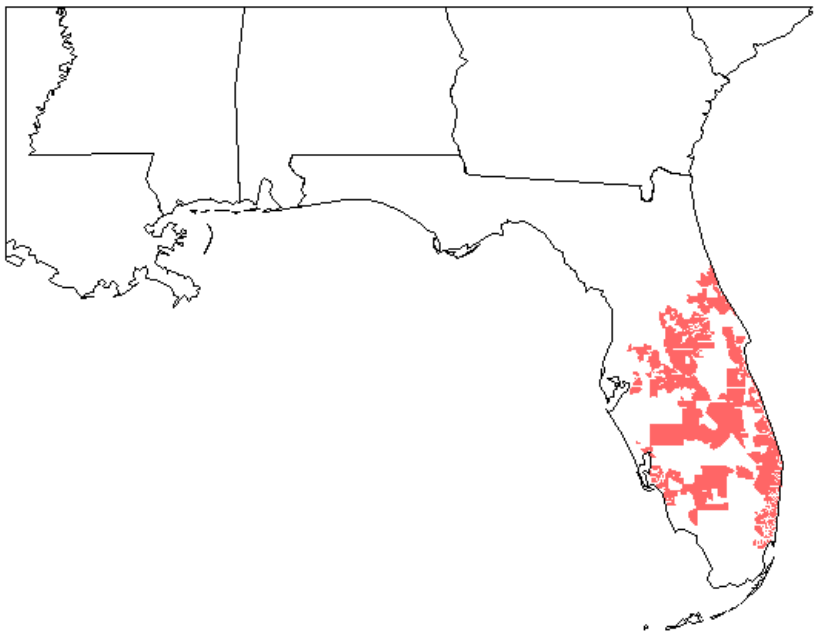


Figure 5.7: Locations detected using TV-MBSS-1 affected by hurricane Frances

time step t. Since we are assuming a movement of the event over two consecutive time steps, we have a pair of sub-regions S_0 and S_1 corresponding to the two consecutive days. In our example of a hurricane strike, S_0 denotes the sub-region where the hurricane hits first, and S_1 denotes the sub-region where it hits the next day. We define $S = S_0 \bigcup S_1$. We assume that all the locations within a given sub-region get affected by the same time-series pattern of effects for a given data stream. However, the time-series patterns corresponding to the two sub-regions are shifted by one time step. Let $\{x_m^t \ : \ t = 0 \ldots (W-1)\}$ denote the time-series pattern of the event for data stream D_m. Let $\{x_{0,m}^t \ : \ t = 0 \ldots W\}$ and $\{x_{1,m}^t \ : \ t = 0 \ldots W\}$ denote the time-series patterns corresponding to the sub-regions S_0 and S_1 respectively. Then,

Figure 5.8: TV-MBSS-2 Detect Algorithm

Parameters: $\mathcal{S}$, For each event type E_k effects $\{x^t_{d,m} : m = 1 \ldots M;\ t = 0 \ldots W\}$

1. Compute the baselines $b^t_{i,m}$ and the parameter priors (α_m, β_m) from historical data, as in the original MBSS algorithm.

2. For each event type E_k, compute the log-likelihood ratios $LLR^\theta_{d,i}$ corresponding to each location s_i, intensity level θ and time lag $d = 0, 1$ aggregated over the M data streams and over the time window $t = 0 \ldots W$ as given by:

$$LLR^\theta_{d,i} = \sum_m \sum_{t=0}^{W} \log LR^{t,\theta}_{d,i,m} \tag{5.11}$$

$$LR^{t,\theta}_{d,i,m} = \left(\frac{\beta_m}{\beta_m + b^t_{i,m}}\right)^{(x^{t,\theta}_{d,m}-1)\alpha_m} \frac{\Gamma(\alpha_m)\,\Gamma(x^{t,\theta}_{d,m}\alpha_m + c^t_{i,m})}{\Gamma(x^{t,\theta}_{d,m}\alpha_m)\,\Gamma(\alpha_m + c^t_{i,m})} \tag{5.12}$$

3. For each $S \in \mathcal{S}$ and event type E_k, compute the posterior log-likelihood ratio $\frac{\Pr(H_1(S,E_k)|D)}{\Pr(H_0|D)}$, as given by eqn. 5.9.

4. For each location s_i and event type E_k, compute the posterior probability $P(H_1(s_i, E_k)|D)$, that s_i is affected by the event E_k, as given by eqn. 5.4.

5. For each location s_i and event type E_k, also, compute the posterior probabilities p^0_i and p^1_i, that s_i is affected by the event E_k on the first and second days respectively.

$$p^0_i = \sum_{\{S\,:\,s_i \in S_0\}} \Pr(H_1(S, E_k) \mid D) \tag{5.13}$$

$$p^1_i = \sum_{\{S\,:\,s_i \in S_1\}} \Pr(H_1(S, E_k) \mid D) \tag{5.14}$$

Note that, $P(H_1(s_i, E_k)|D) = p^0_i + p^1_i$.

$$x^t_{0,m} = x^t_m \text{ for } t = 0 \ldots (W-1),$$
$$= 1 \text{ for } t = W. \tag{5.15}$$

$$x^t_{1,m} = x^{t-1}_m \text{ for } t = 1 \ldots W,$$
$$= 1 \text{ for } t = 0. \tag{5.16}$$

Figure 5.3 shows an example of two such time series patterns lagged by one day.

Note that both the time series $\{x^t_{0,m}\}$ and $\{x^t_{1,m}\}$ have a time duration of length $(W+1)$. This extra day is needed to accommodate the one-day lag between the two time series. Since we are interested in modeling moving events, we additionally assume that S_0 and S_1 are adjacent sub-regions. Specifically, we model the search region S as a pair of non-overlapping but touching circles corresponding to S_0 and S_1 respectively. Figure 5.8 shows the modified algorithm assuming non-stationary events.

5.3.3 Heuristic search procedure

We would like to search over all possible pairs of non-overlapping but touching circles centered at two data points. The number of search regions increases cubically with the number of data points N. Since it is computationally very expensive to compute the posterior for all such possible space time regions, we adopt a heuristic procedure to perform a faster search.

Our goal is to compute the posterior $\Pr(H_1(E_k)|D) \propto \sum_{S \in \mathcal{S}} \Pr(D|H_1(S,E_k))\Pr(H_1(S,E_k)|E_k)$. In practice, we see that only a small set of regions have a significant contribution to the posterior probability value. This leads us to the insight that we can approximate the posterior probability by summing over a truncated set of regions $\mathcal{S}^* \subset \mathcal{S}$, ignoring the regions that have a very low value of $\Pr(D\,|\,H_1(S,E_k))$. Under this assumption, we first identify the set of regions that are most likely to have a high value of the data likelihood given the event model. A region S consists of two sub-regions S_0 and S_1, which are non-overlapping,

Figure 5.9: Heuristic Search procedure for TV-MBSS-2 Detect Algorithm
Parameters: $\mathcal{S}$, For each event type E_k effects $\{x^t_{0,m} : m = 1 \ldots M;\ t = 0 \ldots W\}$

1. Run the TV-MBSS-1 Detect Algorithm using $\mathcal{S}$: set of all circles containing up to $Size_{max}$ locations; and the event model specified by the effects $\{x^t_{0,m} : t = 0 \ldots W\}$.
2. Define $C_0 = \{s_i \ : \ \Pr(H(s_i, E_k)|D) > 0.5\}$ as the first day centers.
3. Define $C_1 = \{s_i \ : \ \text{distance}(s_i, s_j) \leq K, s_j \in C_0\}$ as the second day centers.
4. Initialize $\mathcal{S}^* \leftarrow \phi$.
5. For each pair of locations in $\{\{s_i, s_j\} : s_i \in C_0, s_j \in C_1\}$, construct a set of regions with sub-region centers at $\{s_i, s_j\}$ by varying one of the sub-region radius. Include these regions in the set $\mathcal{S}^*$.

adjacent circles centered at data locations s_i and s_j respectively. This implies that the sum of the radii of the two circles, equals the distance between the data locations s_i and s_j. Thus, given a pair of locations $\{s_i, s_j\}$ as the corresponding centers, we can generate a set of regions by varying the radius of one of the sub-regions, subject to the constraint that the maximum number of locations included in any sub-region is $Size_{max}$. We now focus our attention on identifying pairs of centers that will generate the regions which have a relatively high value of $\Pr(D \mid H_1(S, E_k))$. Specifically, we wish to identify two sets of locations C_0 and C_1, such that the pairs of centers $\{\{s_i, s_j\} \ : \ s_i \in C_0, s_j \in C_1\}$ generate all the regions of interest.

In our experiments, the effect of the event (hurricane) was more pronounced on the area where it hit first. Hence, we first identify the set of most the likely centers C_0 and construct the set C_1 as all the locations (including the ones in C_0) that are within a certain distance of any location in C_0. Given an event type E_k, the steps to generate $\mathcal{S}^*$ are as follows:

By restricting our search over a smaller set of regions, we are effectively choosing a prior distribution over the set of all possible regions, which distributes the probability values uniformly over $S \in \mathcal{S}^*$ and is zero over all regions not belonging to $\mathcal{S}^*$. For the rest of the chapter, we assume this heuristic search region generation for TV-MBSS-2.

5.3.4 Detecting patterns over three or more days

So far, we have assumed that the event affects up to two non-overlapping regions on consecutive days. Generalizing this to k (more than two) days would mean searching over all possible regions S comprised of k non-overlapping adjacent circles. In the worst case there can be O(N^{k+1}) such regions, where N is the number of data locations and $k << N$, and is prohibitively expensive to compute for $k > 2$. Instead, we use an iterative procedure, using the 'pair of circles' approach described above, to extend the region S one sub-region at a time. Similar to the two day case, $t = 0$ represent the time step at which the effects of an event onsets on the overall data and S_t denote the set of locations where the effect of the event onsets at time step t. The effect pattern affecting the sub-region S_j is a j time-step lagged version of the overall pattern, padded with ones appropriately:

$$\begin{aligned} x^t_{j,m} &= 1 \text{ for } 0 \leq t < j \\ &= x^{t-j}_m \text{ for } j \leq t < j + W \\ &= 1 \text{ for } j + W \leq t < (W + k) \end{aligned} \tag{5.17}$$

Below, we give the heuristic procedure to identify an event that affects k different non-overlapping adjacent regions, with the pattern lagged by one time-step from one adjacent region to the next. We run the TV-MBSS-2 algorithm (using the heuristic search from §5.3.3) for each consecutive pair of days, starting from the first two days. After each run, the locations corresponding to the first sub-region are fixed, and are removed from the dataset. The locations corresponding to the second sub-region are then treated as the first day centers for the next iteration of TV-MBSS-2.

At the end of the iteration, we have k fixed sub-regions that are non-overlapping (since

Figure 5.10: Extending TV-MBSS-2 Detect Algorithm for three or more days

Parameters: For each event type, time series effects

$\{x^t_{d,m} : \ m = 1 \ldots M; \ t = 0 \ldots W + k - 1; \ d = 1 \ldots k\}$

1. Using the parameter estimates for the first two time-steps $x_m^{(k-1)}$ and $x_m^{(k-2)}$ run the TV-MBSS-2 algorithm.

2. Identify the set of locations belonging to the first sub-region as $S_{first} = \{s_i : p_i^0 > 0.5\}$, where p_i^0 is given by eqn. 5.13. Remove all locations in S_{first} from the data.

3. Identify the set of locations belonging to the second sub-region as $S_{second} = \{s_i : p_i^1 > 0.5\}$, where p_i^1 is given by eqn. 5.14. Initialize the first day centers of TV-MBSS-2 with S_{second}.

4. Use the parameter estimates for the next two time-steps, and run the TV-MBSS-2 algorithm.

5. Repeat steps 2,3 and 4 for $(k-1)$ times.

at each iteration we remove the locations within the last sub-region) and approximately adjacent (since the centers of the next sub-region come from an adjacent region). Since the sub-regions are fixed, this procedure is no longer fully Bayesian. The score of each sub-region is the log of the posterior likelihood ratio (eqn. 5.3). The final score for the overall composite region S is the sum of all the k sub-region scores.

5.4 Learning the model

The algorithm in Figure 5.8 assumes that the time series of effects x_m^t is known for each event type. In this section, we present a semi-supervised framework to learn these effect parameters from data. As previously mentioned, data labels are usually manually gener-

ated by human experts and can be very difficult to obtain for a large data set. Instead, we assume that we have a few labeled examples of each event type of interest. Since we assume that the event types are mutually exclusive (i.e. up to one event type affects the data on at any given time) we can independently learn the effects for each event type. For this discussion, let us focus our attention on any particular event type E_k (for example, hurricanes). For this event type, we are given a few hand labeled locations that are affected by this event. Our goal here is to learn the event model and then use it to detect other occurrences of events in the data, corresponding to this event type. In our example, we would like to learn how a hurricane strike affects the OTC sales of a region, and use this model to detect other hurricane strikes present in the data.

As previously mentioned, in our case we model events that affect up to two adjacent but non-overlapping regions (sets of locations) on consecutive days. The space time region S consists of a pair of sub-regions S_0 and S_1, corresponding to the two consecutive days.

Then, for each location s_i, the data label is a pair of the form: $\{p_i^0, p_i^1\}$, where p_i^t is the probability that $s_i \in S_t$ for $t \in \{0, 1\}$. In the case of the provided hand labels, either exactly one of these values is equal to one (the event affects the location with the corresponding lag) or both the values are zero (the event does not affect the location). Note that we do not require the user to hand label each location s_i in the data. The user labels a small set of locations that are affected by the event (with the corresponding time lag) and we assume $p_i^0 = p_i^1 = 0$ for all other locations. This small set of labels is usually enough to initialize the time series pattern(s) corresponding to the event, and we follow an Expectation Maximization (EM) based procedure to learn the actual pattern(s).

We also assume that we are given the length W of the time duration over which the event has an effect on a particular location. The range $t = 0 \ldots W$ represents the entire time duration of interest. For a given data stream m the effect parameters x_m^t can be described by as a time series of length W. Figure 5.3 shows an example time-series of the effects of an hurricane on the OTC sales data ($W = 8$).

Our algorithm is based on the Expectation Maximization (EM) framework, where we iterate over estimating the model parameters and inferring the data labels using the learned model.

Figure 5.11: Learning the time series pattern of effects for an event type

1. Initialize: For each location s_i initialize data labels $\{p_i^0, p_i^1\}$ using the manually labeled locations.

2. M Step: We learn the expected value of x_m^t, the effect of the event on the data steam D_m at the time step t. Since x_m^t is a multiplicative factor, the maximum likelihood estimate is given by a weighted average of the ratio of the actual and expected counts, taking into account the proper time lag:

$$x_m^t = \sum_{s_i \in S} \left[p_i^0 \frac{c_{i,m}^t}{\frac{\alpha_m}{\beta_m} b_{i,m}^t} + p_i^1 \frac{c_{i,m}^{t+1}}{\frac{\alpha_m}{\beta_m} b_{i,m}^{t+1}} \right] \text{ for } t = 0 \ldots (W-1)$$

3. E Step: Using the parameter estimates x_m^t, run the TV-MBSS-2 detect algorithm Figure 5.6. Update the data labels $\{p_i^0, p_i^1\}$ based on the output (as given by eqns. 5.13 and 5.14).

4. Repeat: Steps 2 and 3 until stopping criteria is met.

In the EM framework, the stopping criteria in Step 4 is reaching convergence (data labels remain constant over successive iterations). However, in our experiments we have observed that in most cases there is very little change after the second iteration. Since Step 3 is computationally expensive, in our evaluations, we stop after the second iteration.

5.4.1 Using Alternate Event Explanations

As mentioned in §5.2 we assume that up to one event E_k affects the data at any given time, i.e., $\Pr(H_0) + \sum_k \Pr(E_k) = 1$. This assumes that we have a comprehensive list $\{E_k\}$ of all events that can affect the data. In practice, in most realistic situations, it is not possible to have a complete list of all such possible causes. Specifically, in our example of monitoring the OTC sales data, we are interested in the event type "hurricane", although the data contains a lot of variations due to other causes. Most such variations (including

noise) appear over very short time durations, and can be seen as a single day upward or downward spike in the counts. Since we are interested in detecting an extended temporal pattern (over $W = 8$ days), we would like to ignore any single day variation of the counts. This motivates us to extend the null hypothesis H_0 to include these alternate events (single day spikes). In this case, we define a series of $W + 1$ upward-spike and downward-spike events. For $i = 0 \ldots W$, the ith upward-spike event, H^i_{0-up} is characterized by a time-series of effects (identical over all the data streams), with $x^t = 1$ for all $t \neq i$, and $x^i = 2$. Similarly, the jth downward-spike event H^j_{0-down}, is characterized by a time-series of effects, with $x^t = 1$ for all $t \neq j$, and $x^j = 0.5$.

The data likelihood under this alternate null H^*_0, is then given by:

$$\Pr(D \mid H^*_0) = \max_{i,j} \left\{ \Pr(D \mid H_0), \Pr(D \mid H^i_{0-up}), \Pr(D \mid H^j_{0-down}) \right\} \tag{5.18}$$

We then use this likelihood value in eqn. 5.3 to compute the event score. In the following sections, this method of using alternate event explanations for the null hypothesis is called TV-MBSS-Alt.

5.5 Evaluation and Results

We evaluate the TV-MBSS method on a dataset of Over the Counter (OTC) sales (as described in §1.4.3). This dataset consists of OTC medicine sales in pharmacies all over the US. It spans over two years, March 2004 - March 2006. In this work, we evaluate the methods in their effectiveness to detect the sales pattern caused by hurricane strikes from the Atlantic ocean. Figure 5.1 shows the data coverage over zip codes in the eastern part of United States. We wish to detect both when and where the hurricane hit. For the ease of evaluation, we consider a single data stream ($M = 1$), the aggregate over all the categories of sales in a zip code. As previously mentioned in §5.1, this is a proof of concept evaluation. This method can be applied to detect any event type which has a specific temporal pattern of effects and which may have a region of effect which moves from one day to the next.

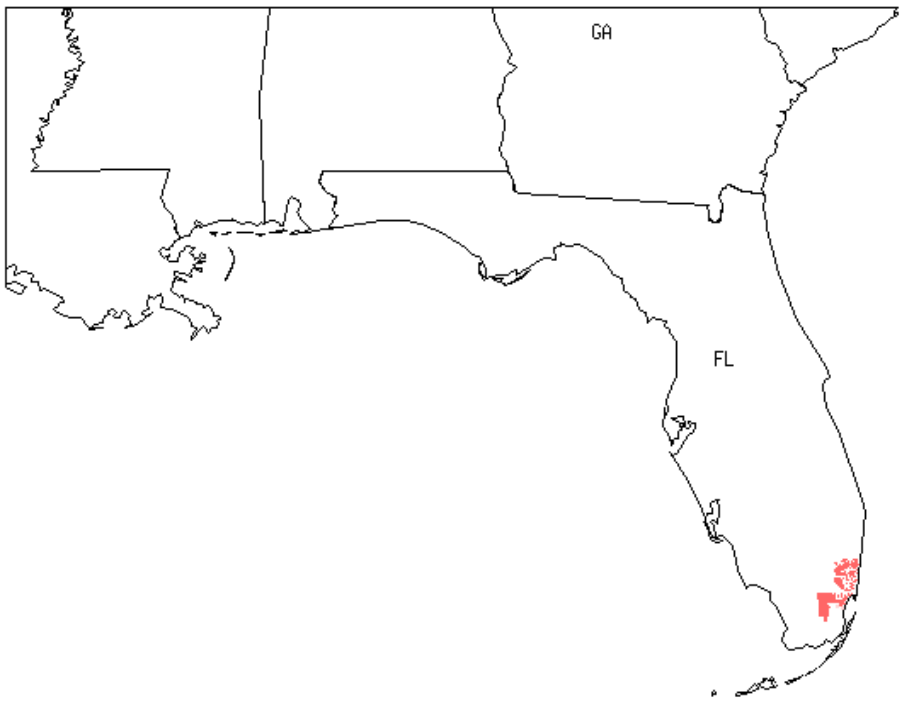


Figure 5.12: Partial hand labels of locations affected by hurricane Frances used for semi-supervised learning

We start by presenting results of our algorithm when applied to learn and detect the pattern corresponding to a particular hurricane. We then present a more comprehensive evaluation over the entire hurricane season for 2004 and 2005, comparing our method to MBSS.

5.5.1 Learning to detect a single hurricane

We first compiled a small list of zip codes that were affected by hurricane Frances on September 5th 2004 (as shown in Figure 5.12). We fixed the duration of the effect at a location, W = 8 days. We use $k = 3$ in this case, i.e. the effect of the hurricane starts appearing on three consecutive days in three non-overlapping adjacent regions. We choose

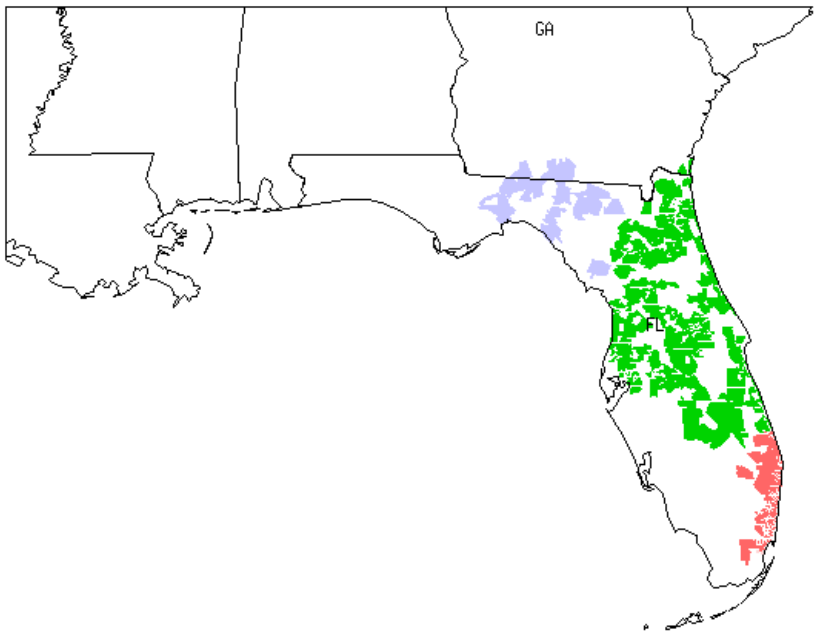


Figure 5.13: Locations detected using TV-MBSS as affected by hurricane Frances

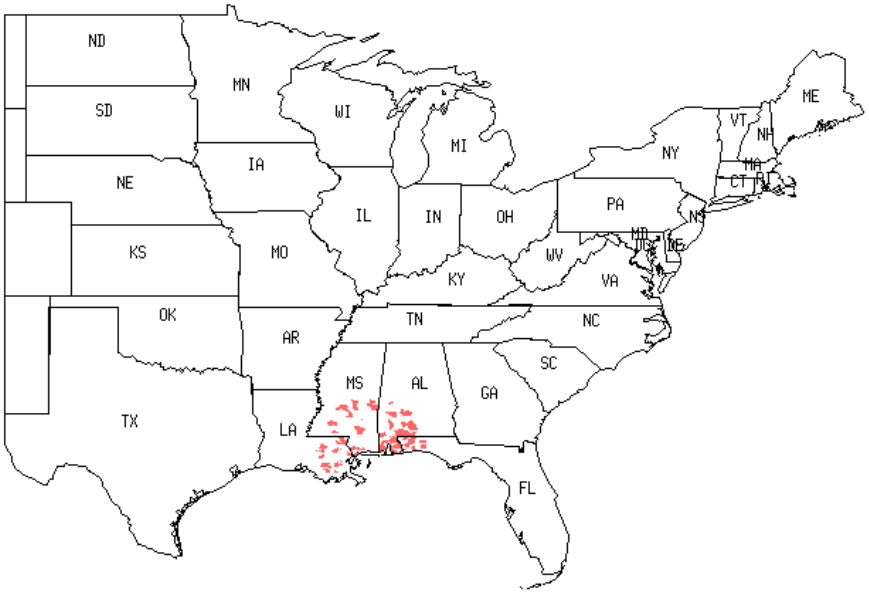
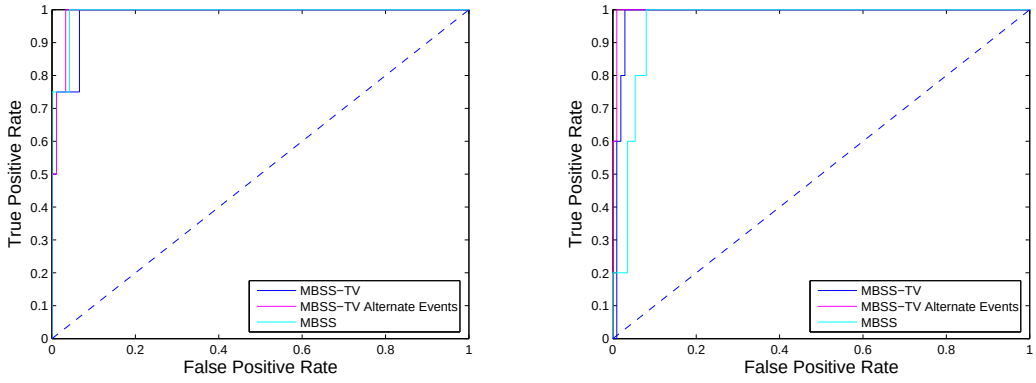


Figure 5.14: Locations detected using TV-MBSS as affected by hurricane Katrina

this value of k since by the fourth day after initial landfall, the hurricane had lost strength and turned into a tropical depression, and no longer had any effect on the OTC medicine sales. Therefore, the region of interest S consists of three sub-regions: S_0, the sub-region where the hurricane hit the first day, S_1 and S_2, the regions where it hit on subsequent days.

Figure 5.13 shows the output of the algorithm. Each zip code is color coded to represent the probability that it belongs to one of the sub-regions: S_0 (red), S_1 (green) and S_2 (blue). Higher probability is indicated by more intense color. We see that the progression of the hurricane effects match quite well with the hurricane track information from NHC shown in Figure 5.2.

Figure 5.14 shows the output of the algorithm run on August 26th 2005, using the pattern learned from hurricane Frances. In this case we detect the effect of hurricane Katrina. Although Katrina caused major devastation near the coast, we see that the effects of Katrina are not pronounced as it moved inland on subsequent days. The zip codes



(a) 2004 (b) 2005

Figure 5.15: ROC curve comparing our methods with MBSS for the task of hurricane detection in OTC dataset

marked in red corresponds to the region affected on the day Katrina made landfall.

5.5.2 Comparison with MBSS

In our next evaluation, we compared the effectiveness of our methods with MBSS. Here, MBSS searches over space time regions to detect increases in count with a time window varying between 1 to 4 days. For TV-MBSS, we use the hurricane effects pattern learned in the previous section (from the effects of hurricane Frances). We use the heuristic search procedure (§5.3.3) to search over pairs of circles. We also evaluate the TV-MBSS with Alternate event explanations as described in §5.4.1. For each method, we perform a separate run corresponding to each day (assuming that day as $t = 0$) of the hurricane season (July 1 - Oct 31) for the years 2004 and 2005. We note that there are four hurricanes in 2004 (Charlie, Frances, Ivan and Jeanne) and five hurricanes in 2005 (Dennis, Katrina, Ophelia, Rita and Wilma), that has an effect on the data. For each method, and each run (corresponding to each day), we compute the log-likelihood ratio of posterior probabilities: $\log \frac{\Pr(H_1(S,E)|Data)}{\Pr(H_0|Data)}$ as given by eqn. 5.3. Here E denotes the event type 'hurricane'. This gives the corresponding score for that run. A higher value indicates that it is more

Table 5.1: Area under the curves for ROC curves in Figure 5.15, with standard errors

Method	2004	2005
TV-MBSS	**0.981 ± 0.016**	0.984 ± 0.004
TV-MBSS-Alt	**0.989 ± 0.008**	**0.996 ± 0.002**
MBSS	**0.989 ± 0.011**	0.959 ± 0.013

likely to be affected by a hurricane, where the effect pattern onsets at $t = 0$ for that run.

We observe that for all the methods, the effects of a single hurricane are usually detected in more than one run. This is because in some cases, even if the search pattern is time shifted with respect to the actual pattern in the data, and is not an exact match, it still provides a better match than the null hypothesis (of no effects). In such cases, we consider the maximum score over these multiple runs as the true positive score of detecting the hurricane. We set a threshold score, and any run that has a score greater than this threshold is flagged as a detect. The runs in which regions not affected by a hurricane are detected, are treated as false positives. We have used the ground truth information about hurricane hits from the National Hurricane Centre archives [NHC].

We then examine each method's tradeoff between its false positive rate (proportion of runs without hurricane effects that were falsely detected) and its true positive rate (proportion of runs with hurricane effects that were correctly detected) by varying the threshold score. This is the standard ROC curve: a higher curve denotes better detection performance, since it corresponds to a higher true positive rate for a given false positive rate. The area under the ROC curve (AUC) can be used as a summary measure, where higher AUC corresponds to better average performance.

Figure 5.15 shows the ROC curve for all the three methods for 2004 and 2005. Table 5.1 gives the corresponding area under the curve values. The AUC for the best method, and for the other methods which are not significantly less than the best are shown in bold font. We see that for 2004, all the methods have very similar performances. For 2005, we see that both TV-MBSS and TV-MBSS with Alternate events perform significantly better than MBSS.

5.6 Conclusions

In this chapter we presented a set of methods to characterize and detect events that have time varying effects, and which can move over different locations from one time step to the next. We implement our ideas as an extension (TV-MBSS) to the state of art scan statistics MBSS. As a proof of concept evaluation, we use data from OTC sales to model and identify hurricane strikes in the south-east coast of United States. We show that TV-MBSS is able to correctly model the time varying pattern of effects, as well as the time lagged patterns as the hurricane moves inland. We do a ROC analysis of the effectiveness of detection, and show that for some cases TV-MBSS performs significantly better than MBSS.

Chapter 6

Searching through composite time series

6.1 Introduction

In this chapter we focus on the purely temporal domain for detecting anomaly patterns. The drastic decrease of data storage costs, availability of cheap sensors along with automation of systems have resulted in proliferation of time series data. In most cases the data is multivariate in nature, and the effect of an anomaly can potentially be observed across more than one of these series. Many classical statistical methods deal with univariate data, and less has been done about multivariate data. A traditional method of applying univariate methods in these cases would be to reduce the dimension using some dimensionality reduction technique (e.g. PCA). But, an anomaly detected in a weighted linear combination of the data might not be meaningful to the end user. A majority of such techniques suffer from the lack of user interpretability of the results. This motivates our approach of search through simple arithmetic combinations of time series.

Modern surveillance systems are characterized by the need to analyze many variables simultaneously, and the traditional method of setting upper and lower bounds for a single variable are no longer appropriate. Data mining methods must address the complex interactions between variables, the dangers of multiple hypothesis testing, and the computational issues caused by large data sets. [Wong, 2004] gives an overview of time series

anomaly detection methods.

We consider the problem of detecting an anomalous increase of values in multivariate time series data. The problem stems from the fact that the increase can be spread over multiple variables. As an example, consider the time series of counts of patients visiting emergency departments every day. For each possible symptom we have a corresponding time series. A particular disease such as an influenza outbreak, will affect the count of multiple syndromes. In this case, we need to simultaneously consider all the variables to detect the presence of an anomaly. We are concerned with prospective surveillance, where we need to detect a disease outbreak as soon as possible.

To combine information from multiple time series we examine a novel technique which is simple but powerful. Composite time series are constructed by simple addition and subtraction of the individual time series. We search through all possible composite time series for an anomaly. Using just simple arithmetic operations like addition and subtraction provides an easy physical interpretation of the composite series. It is also able to detect anomalies sooner than other traditional methods.We compare the performance of our algorithm with related methods such as Vector Autoregression on semi-synthetic health data.

6.2 Related Methods

In this section we describe various multivariate techniques that can detect a shift in the data.

6.2.1 Vector Auto Regression

The time series is modeled as a standard VAR(p) model [Hamilton, 1994]. Let the number of variables be n. Let $\mathbf{X_t}$ denote the $(n \times 1)$ vector of values at time t.

$$\mathbf{X_t} = \mathbf{C} + \sum_{i=1}^{p} \mathbf{\Phi_i} \mathbf{X_{t-i}} + \epsilon_\mathbf{t}$$

where, $\mathbf{C}$ denotes an $(n \times 1)$ vector of constants, $\mathbf{\Phi_i}$ are $(m \times m)$ coefficient matrices and the $(n \times 1)$ vector $\epsilon_\mathbf{t}$ is the residual vector. Here $E[\epsilon_\mathbf{t}] = 0$. The coefficients $\mathbf{\Phi_i}$ can be estimated from data using ordinary least squares (OLS) linear regression.

The expected value of $\mathbf{X_t}$ given the past p days' data is given by

$$E[\mathbf{X_t}] = \mathbf{C} + \sum_{i=1}^{p} \mathbf{\Phi_i} \mathbf{X_{t-i}}$$

At each time step, we compare the actual and expected values of $\mathbf{X_t}$. We signal an alarm when $\mathbf{X_t}$ deviates significantly from $E[\mathbf{X_t}]$. Quantitatively, we compute the Mahalanobis distance:

$$D^2 = (\mathbf{X_t} - E[\mathbf{X_t}])^T \mathbf{\Sigma}^{-1} (\mathbf{X_t} - E[\mathbf{X_t}])$$

where, $\mathbf{\Sigma}$ is the sample variance-covariance matrix for the past p days' data.

An alarm is signaled when D exceeds a threshold h. Here h is the parameter which controls the number of false positives.

6.2.2 Vector Moving Average

This method is a special case of the Vector Autoregression as described above. We assume that the expected value of $\mathbf{X_t}$ is the mean of the past p days' values.

$$E[\mathbf{X_t}] = \frac{1}{p} \sum_{i=1}^{p} \mathbf{X_{t-i}}$$

We compute the Mahalanobis distance as mentioned previously, and signal an alarm when $D > h$.

6.2.3 Hotelling T Squared Test

We model the distribution of the mean of the recent p days' data. Let

$$\bar{\mathbf{X}} = \frac{1}{p} \sum_{i=0}^{p-1} \mathbf{X_{t-i}}$$

and $\mathbf{\Sigma}$ be the sample variance-covariance matrix for the past p days' data.

The statistic T^2 is defined as [Hotelling, 1947]:

$$T^2 = n(\bar{\mathbf{X}} - \mu)^T \mathbf{\Sigma}^{-1} (\bar{\mathbf{X}} - \mu)$$

Here, μ is the fixed expected value of the mean. T^2 is distributed as $\frac{p(n-1)}{n-p} F_{(p,n-p)}$, with $F_{(p,n-p)}$ representing the F distribution with p and $n-p$ degrees of freedom. We signal an alarm when $P(x \geq T^2) < \alpha$, where α controls the rate of false positives. Application of Hotelling T^2 in multivariate quality control has been investigated in [Hong and Hardin, 2004].

6.3 Detection method: CUSUM

Before presenting our algorithm, we describe a popular method used in detecting anomalies in time series. CUSUM was originally developed to detect changes in the quality of output of continuous production process. It can quickly detect a shift in the mean of a process. As the name suggests, CUSUM maintains a cumulative sum of deviations from a reference value r. Let us consider a time series where at time t we have measurement $X(t)$. The one-sided CUSUM calculation is as follows:

$$C(0) = 0 \tag{6.1}$$

$$C(t) = max(0, X(t) - (\mu_0 + L) + C(t-1)) \tag{6.2}$$

μ_0 is the in-control process mean. From the equations above, if the X_m values are close to the mean, then the $C(t)$ values will be some small value. However once a positive shift from the mean occurs, the $C(t)$ value will increase rapidly. L is known as the slack value or allowance. In the equation above, any values within L units of μ_0 will be effectively ignored. The allowance L is usually set to be the midpoint between the in-control process mean μ_0 and the out-of-control process mean μ_1.

Alerts are raised whenever $C(t)$ exceeds a threshold decision interval H. The cumulative sum is then reset to zero. The Average Run Length (ARL) is controlled by this parameter. The ARL is the average number of time steps before an alert is raised under the assumption that the process is in-control.

The CUSUM algorithm described here has been extensively used in biosurveillance systems. It has been used for influenza surveillance [Tillett and Spencer, 1982], detection of salmonella outbreaks [Hutwagner et al., 1997] and in the Early Aberration Reporting system [Hutwagner et al., 2003]. CUSUM algorithms have also been extended to incorporate spatial information such as [Raubertas, 1989] and [Rogerson, 1997].

6.3.1 Modified CUSUM

In this work we use a modified CUSUM as the detection method. We have found this method to be very effective in detecting upward shifts in time series.

We calculate the cumulative sum of deviation similar to equation 6.2. Instead of maintaining the cumulant starting at t=0, we consider only the last CW (Cumulant Window) number of time steps. This means that the current Cumulant at time t, will be independent of any data before the time T - CW. We signal an alarm if the current cumulant value is greater than H. This modification does not affect the performance of the algorithm significantly, and is actually desired in our case, as explained later. This also allows us to speed up the computation as described in section 6.7.

In the original algorithm, H is usually taken as a fixed threshold value. We have set H = hσ, a multiple of the standard deviation σ of the time series. We need to calculate and update the σ value at each time step. In our method σ is the sample standard deviation of the series calculated over a sliding window of the last N days. Thus, H is dynamically updated based on the behavior of the variable. Also, since we do not know the out-of-control process mean μ_1, we set $L = l\sigma$, for some constant l. L too gets updated at each time step. The in-control process mean μ_0 is taken as the moving average over the last N days. This dynamic updation of the parameters at each time step is a significant modification of the original CUSUM algorithm. This allows us to model non-stationary

time series variables.

6.3.2 Multivariate CUSUM

An analogous Multivariate version has also been applied to surveillance data. Crosier's multivariate cumulative sum (MCUSUM) method [Crosier, 1988] has been applied to syndromic data from multiple hospitals [Stoto, 2004] and Pignatiello's MCUSUM [Pignatiello and Runger, 1990] applied to yearly, spatially distributed counts of breast cancer incidence [P. A. Rogerson, 2004]. We have implemented the MCUSM method from [Pignatiello and Runger, 1990] and compared it against our method.

6.4 Proposed Method: Parallel Monitoring of Composite Series

A common feature of all the multivariate methods is that the statistic on which the alarm is set, does not have an intutive physical interpretation in terms of the variables. However, if we monitor the individual variables in parallel, we can identify the variable that has an anomalous behavior in case of an alarm.

Also, as mentioned in [Burkom et al., 2004]:

> These (multivariate) methods are omnidirectional, a property that can be useful in detecting an earlier signal, but can also cause false alerts if a change in the covariance matrix occurs that is irrelevant to any outbreak signal of interest

They do not specifically check for increases in individual series. In our experiments this causes them to perform worse than parallel monitoring of univariate series.

The novel method that we suggest involves parallel monitoring of not only the individual variables, but also simple arithmetic combinations of them. This retains the advantage of easy interpretability while giving a better performance as shown in our experiments.

This method of using combinations of time series is orthogonal to the univariate detection method used to monitor each series. We have chosen CUSUM as the detection algorithm because of its superior and robust performance in detecting slight increases over the normal value. In the following sections we describe this algorithm in more detail.

6.5 Search space

As mentioned previously, we perform a parallel monitoring of the time series variables and arithmetic combinations of them. Here we describe the composite series that are monitored in parallel for any increase from expected values.

Let X_1, ..., X_k be k random time series variables, and $X_i(t)$ denote the value of X_i at time step t.

Addition We create time series of the form:

$$Y = X_{i_1} + X_{i_2} + \ldots + X_{i_m};\ i_1, \ldots, i_m \in \{1, 2, \ldots, k\}$$

This means that at each time step t,

$$Y(t) = X_{i_1}(t) + X_{i_2}(t) + \ldots + X_{i_m}(t);\ i_1, \ldots, i_m \in \{1, 2, \ldots, k\}$$

Here we can choose the indices i_1, i_2, ..., i_m in $\binom{k}{m}$ ways. If we consider summations of up to k terms, the total number of such composite series = $\binom{1}{m} + \binom{2}{m} + \ldots + \binom{k}{m}$.

Subtraction As in the case for addition, we create time series of the form

$$Y = X_{i_1} - X_{i_2};\ i_1, i_2 \in \{1, \ldots, k\}$$

Here we consider combination of just 2 series. There are $\binom{n}{2}$ such composite series.

Motivation of the addition and subtraction operations:

1. Addition: We assume that an outbreak simultaneously causes an increase in the value of more than one variable. The detection accuracy of any anomaly detection method will depend on the signal to noise ratio (SNR) of the outbreak. Here, the anomalous increase in the value is the signal we want to detect, and the standard deviation of the variable is the noise. We now show a simple situation where the composite additive series will have a better SNR than either of the individual series. Consider two random time series variables X_1 and X_2. Assume that they have equal standard deviations, $\sigma_{X_1} = \sigma_{X_2} = \sigma$. Let a be the actual anomalous increase in the values of X_1 and X_2.

   Let $Y = X_1 + X_2$. Now,

   $$\sigma_Y^2 = \sigma_{X1}^2 + \sigma_{X2}^2 - 2 * r * \sigma_{X1}\sigma_{X2} = 2 * \sigma(1 - r)$$

   where r is the Pearson correlation coefficient between X_1 and X_2. By definition, r$\geq$-1. Hence, $\sigma_Y \leq 2{*}\sigma$. The SNR of the individual variables is $\frac{a}{\sigma}$. The SNR of the composite series Y is $\frac{2a}{\sigma_Y} \geq \frac{a}{\sigma}$. In general if the two variables are uncorrelated or negatively correlated, and the anomalous increase is positively correlated, then we can expect a better SNR for the composite time series.

   We note that if there is a very strong positive correlation between the variables, then the noise (variance) will increase proportionately to the signal (outbreak). In such cases, the increased false positive rate (due to multiple hypothesis testing) can lead to a worse performance.

2. Subtraction:

   Considering series of the form $Y = X_1 - X_2$ can be helpful if there is a positive correlation between X_1 and X_2. If these two random variables are positively correlated, then any anomalous increase present in X_1, but not in X_2, will be more pronounced in Y. This is because the noise will tend to cancel, whereas the signal will be left unaffected. Note that there is an increase of the false positive rate due to multiple hypothesis testing. Hence we expect an improvement using the subtraction operator only when there is a high positive correlation among the variables.

6.6 Outbreak simulation

Because there were no known outbreaks in our datasets, we assumed artificial outbreaks by adding ramp increases. We call these outbreaks as attacks, since one of the motivations of this work is to detect bioterrorist attacks.

$$\begin{aligned} attack(t) &= atttack\_height * \frac{(t - t_{start})}{(t_{start} - t_{end})}; \\ &\quad \text{for } t_{start} \geq t \geq t_{end} \\ &= 0 \text{ otherwise} \end{aligned} \tag{6.3}$$

The attacks are spread through more than one time series. We randomly choose m of the k time series to add an attack. We choose m random weights $w_1, \ldots, w_m$ uniformly from the set $\{(w_1, w_2, ..., w_m) | 0 \leq w_i \leq 1, \Sigma w_i = 1\}$. We then add a weighted attack to each of these m time series:

$$X_i^{attack}(t) = X_i(t) + w_i * attack(t); \text{for i = 1, ..., m}$$

We spread the attack to more than one time series so that it becomes difficult to detect it from any individual variable. The effect of attack becomes more evident when we combine more than one variable.

6.7 Search Algorithm

As mentioned in section 6.5, the number of composite time series can be very large. Let C_i denote the C value of the composite time series TS_i. One approach to monitor all these series individually would be to store the C_i values corresponding to each of these series and update them at each time step. At each time step, we signal an alarm if the C_i value of any of the composite time series exceeds the corresponding hσ_i. We also need to store and update each σ_i value at each time step.

Let m be the maximum number of individual series in a composite series. In cases where k is large this method will require an exponential amount of memory depending on

m. We now describe a branch and bound approach that does not require us to store all the C_i and σ_i values.

The main idea is to determine whether a composite series can possibly signal alert without explicitly calculating the C_i value. If we are able to eliminate a majority of the series by using an appropriate bound, then we need only calculate the C_i and σ_i values only for a small fraction of them.

First we note that at a particular time step, if $X_i(t)$ - (μ_0^i + K) < 0, then we can ignore the composite series i. This is because at this time step, the C_i value will decrease, and it cannot signal a new alert. μ_0^i is taken as the moving average of X_i over a past window of N days. σ_i^2 is calculated as the sample variance of the last N days. For simplicity we assume that the mean μ_0^i has been subtracted from X_i for each i, as a preprocessing step. We have fixed N = 21 days and $CW = 4$ days in all our experiments.

6.7.1 Searching through the additive space

We search through additions of all possible combinations of m time series from the k series. The search is done in a depth first manner. We find a lower bound on the standard deviation of the sum of two random variables. Let X_1 and X_2 be two random variables, and σ_{X_1} and σ_{X_2} be the corresponding standard deviations. Let Y = X_1 + X_2. The standard deviation of Y is given by σ_Y^2 = $\sigma_{X_1}^2$ + $\sigma_{X_2}^2$ - 2*r*σ_{X_1} σ_{X_2}. Here r is the Pearson correlation coefficient of X_1 and X_2. We can obtain a lower bound for σ_Y when r=-1. We can better this bound if we can assume that r is lower bounded by a higher value.

Now, let $\hat{\sigma}_{X_1} \leq \sigma_{X_1}$ and $\hat{\sigma}_{X_2} \leq \sigma_{X_2}$, where $\hat{\sigma_{X_1}}$ and $\hat{\sigma}_{X_2}$ are lower bounds on the standard deviation of X_1 and X_2. Let $\hat{r}$ be a lower bound on the correlation coefficient of X_1 and X_2. Define, $\hat{\sigma}_Y^2$ = $\hat{\sigma}_{X_1}^2$ + $\hat{\sigma}_{X_2}^2$ - 2*$\hat{r}$*$\hat{\sigma}_{X_1}$ $\hat{\sigma}_{X_2}$. Under these assumptions it can be shown that $\hat{\sigma}_Y^2 \leq \sigma_Y^2$, ie $\hat{\sigma}_Y$ gives a lower bound on the standard deviation of $Y = X_1 + X_2$.

Our depth first search algorithm is as follows. We describe our search algorithm as a recursion:

For each time step t:

Initialize

1. Update the standard deviations σ_1, σ_1, ..., .σ_k.
2. S $\leftarrow \phi$.
3. DfsRecur(S,0)

DfsRecur(S, $\hat{\sigma}_{X_S}$)

1. Let max_index = the maximum index number among the series present in S.
2. X_S = X_{i_1} + ... + X_{i_p}, where X_{i_1} , ... , $X_{i_p} \in$ S
3. If $X_S \leq h\hat{\sigma}_{X_S}$, then goto step 8
4. Calculate the value of σ_{X_S}. This step requires O(N) time, where N is the moving-average window size.
5. If $X_S \leq h\sigma_{X_S}$, then goto step 8
6. Calculate the value of C_S, the cumulative sum for the composite series S. We need only consider CW days in the past to calculate this value.

$$C(0) = 0 \tag{6.4}$$

$$C(i) = max(0, X_S(t - CW + i) - (\mu_0 + L) + C(i - 1)), \text{ for } i = 1, \ldots CW \tag{6.5}$$

$$C_S = C(CW) \tag{6.6}$$

   If $C_S \geq h\sigma_{X_S}$, then signal an alert.

7. If $\mid S \mid$ = m, return

8. For each i such that $max\_index < i \leq k$

   (a) $S' = S \cup X_i$

   (b) if $|S'| > m$ then return

   (c) $\hat{\sigma}_{X_{S'}}$ = sqrt($\hat{\sigma}^2_{X_S}$ + $\hat{\sigma}^2_i$ - 2$\hat{r}$ $\hat{\sigma}_{X_S}$ $\hat{\sigma}_i$).

   (d) $DfsRecur(S', \hat{\sigma}_{X_{S'}})$

Here m is the maximum number of series that are considered in one composite series X_S. It first calculates a lower bound of the standard deviation of a composite series without explicitly calculating it from the past data. This lower bound allows us to determine if the current value of the composite series can possibly signal an alert. We can avoid calculating the exact standard deviation and cumulative sum by this bounding procedure. In a fraction of cases we actually need to perform the exact calculations.

We use exactly the same procedure to search through the difference series as well. The only difference in this case is that $\hat{\sigma}^2_Y$ = $\hat{\sigma}^2_{X_1}$ + $\hat{\sigma}^2_{X_2}$ - 2*$\hat{r}$*$\hat{\sigma}_{X_1}$ $\hat{\sigma}_{X_2}$, where Y is the difference series of X_1 and X_2.

6.8 Datasets

We use three datasets in our experiments. We use the method described in §6.6 to inject simulated anomalies into these datasets.

1. Over the Counter medicine sales data (OTC) in US (described in Chapter 1, §1.4.3. Each sale belongs to one of the following categories:

   (a) Baby/Child Electrolytes

   (b) Cough/Cold

   (c) Internal Analgesics

   (d) Stomach Remedies

(e) Thermometers

We have 5 time series corresponding to each of the above categories for a period of about 2 years.

2. Emergency department dataset from the regions around Pittsburgh (described in §1.4.4. This data spans 668 days.

   The PRODROME attribute describes the category of the patient's complaint upon arrival at the emergency department. It can have 7 possible values. Correspondingly, we get 7 time series of the count of patients each day.

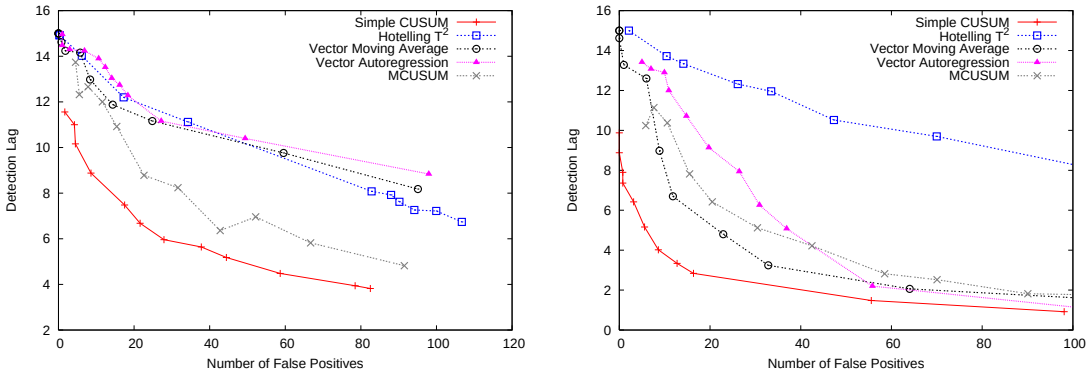
3. Stock Prices Dataset: We consider the daily stock prices of the following 12 companies: Dell, Sun, GE, IBM, Microsoft, GM, Nissan, Toyota, Sony, Ford, BP and Exxon Mobil for a period of 4 years.

6.9 Results

To measure the performance of the algorithms, we need to measure their false positive rate and the corresponding detection lag. Detection lag is the time difference between the start of the attack and the first instance when an alert is signaled with the attack underway. A plot of the number of false positives vs the detection lag is called an AMOC (Activity Monitoring and Control Chart) curve.

To get a point on the AMOC curve we do the following:

1. Fix a value of h, where, $H = h\sigma$, is the CUSUM threshold.
2. For i = 1 to 50,
   (a) Inject a random attack of duration 15 days in the data. The attack is spread over at most three individual variables.
   (b) Estimate the baseline trend values using Moving Average with a slide window of length 21 days.



(a) OTC dataset

(b) ED dataset

Figure 6.1: AMOC Curves comparing Related Methods

(c) Run the modified CUSUM algorithm on the residues. Keep track of the number of false positives and the detection lag. If no alert is signaled within the duration of the attack, the detection lag is taken as the duration of attack.

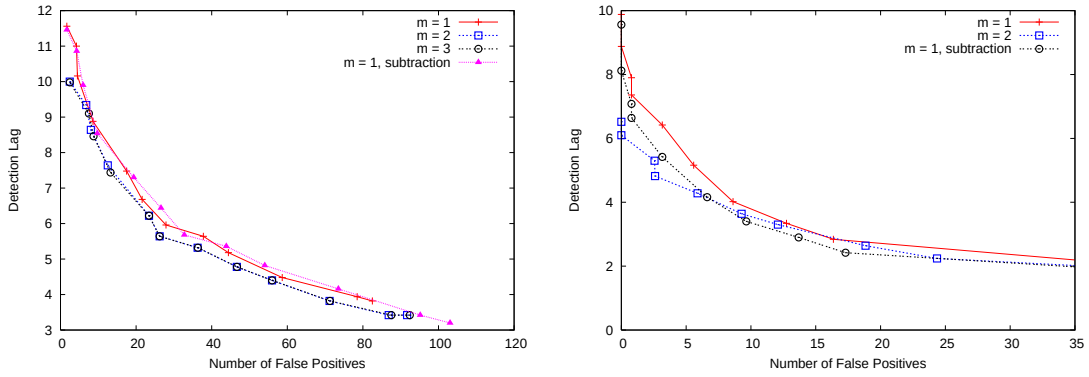
3. Calculate the average number of false positives and the average detection lag over the 50 random attack simulations.

This gives us a point on the AMOC curve. We then vary h to obtain the entire curve.

We ran our algorithm on each dataset, with different values of m (the maximum number of series in a composite series). We compared the CUSUM algorithm with VAR, Vector Moving Average, Hotelling T^2 and MCUSUM. Both VAR and Vector Moving Average used a 3-day slide window (p=3). Hotelling T^2 used the last 10 day's values for calculating the mean.

6.9.1 OTC Dataset

Fig 6.1(a) shows the comparison between CUSUM and the other related methods as explained in section 6.2 for the OTC dataset. We run CUSUM on the individual series independently for the Simple CUSUM method (m=1). We see that CUSUM significantly



(a) OTC dataset

(b) ED dataset

Figure 6.2: AMOC Curves comparing different combinations of time series

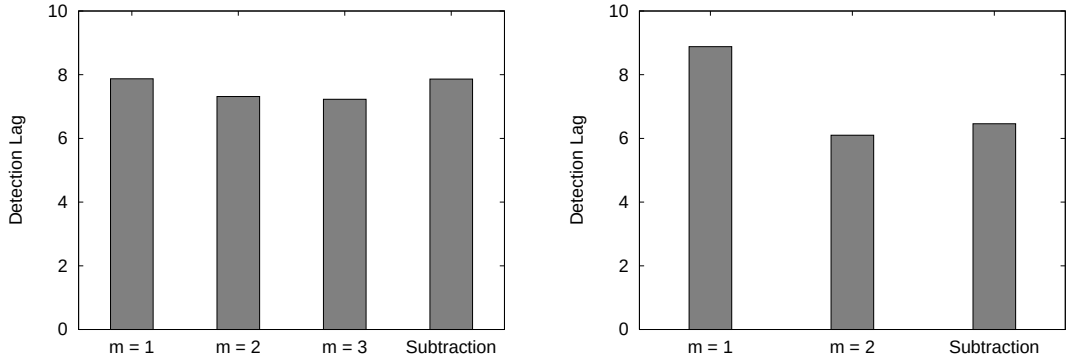
outperforms the other methods. For the same False Positive rate, it gives a much lower Detection Lag.

Fig 6.2(a) shows the curves for CUSUM where m varies as 1, 2 and 3. The fourth curve corresponds to considering the difference series as explained in section 6.7. We see that there is an improvement in the detection lag time when we consider summation of two or more series. The performance of the two series and three series algorithms are similar. But the difference operation does not seem to give an improvement. For a fixed false positive rate of 15 for the entire duration, Fig 6.3(a) shows the corresponding Detection Lags. The detection lag is 7.87 days for m = 1. It improves by about 8% to 7.23 days for m = 3.

6.9.2 Emergency Department Dataset

Fig 6.1(b) shows the comparison between CUSUM and the other related methods. Similar to the OTC dataset, we see that CUSUM significantly outperforms the other methods.

The AMOC curves for this dataset are shown in Fig 6.2(b). There is a significant difference in the detection lag time for very low ($<$10) false positive rate. For example, for no false positives over the entire duration, the detection lags are 8.88, 6.1 and 6.46, for m = 1, 2 and the difference operator respectively. This is illustrated in the bar chart Fig



(a) OTC dataset: 15 false positives

(b) ED dataset: 0 false positive

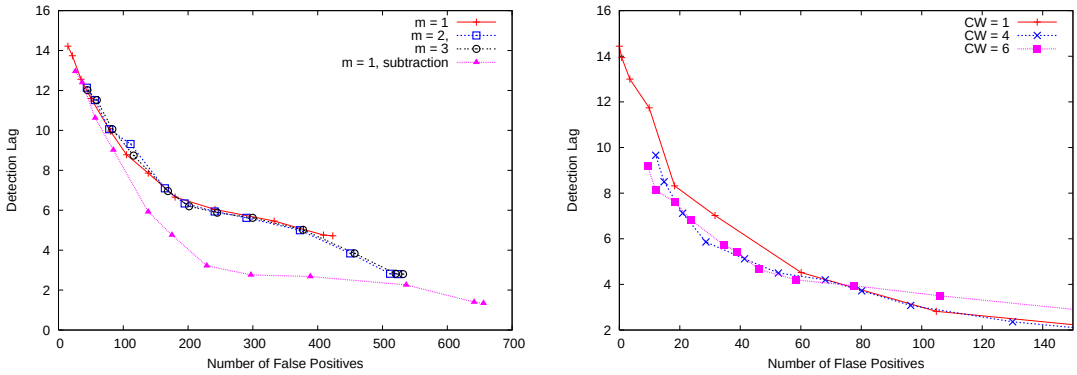
Figure 6.3: Improvement in Detection lag using the proposed methods corresponding to fixed number of false positives over the duration

6.3(b). We see an improvement of 2.78 days or 31% in detection lag when considering more than one series. In applications such as disease outbreak detection, we need to have a low false positive rate. Having a high false positive rate makes the system almost useless because it becomes infeasible to investigate each alarm. Hence, our result in the low false positive range is significant.

Effect of Cusum Window(CW)

CW denotes the number of previous days that are considered to calculate the cumulative for the current day. In eqn 6.5, when CW = 1 and L = 0, C_S measures the deviation of the current value from the expected mean. The CUSUM test in this case becomes identical to the one sample Gaussian test (computing the p-value of a sample). In our experiments, we have set L = σ, which empirically give the best results. Hence for CW = 1, our test is similar to the simple Gaussian test, except for the effect of L. L defines a threshold such that we are concerned only about increases that are above that threshold.

Another advantage of CUSUM over the Gaussian testing is that it considers samples from CW past days. If there is a gradual increase in the time series, it can utilize past



(a) Stock Prices dataset: Plot of Detection Time vs. False Positives

(b) ED dataset: Plot of Detection Time vs False Positives for ED dataset with varying CW

Figure 6.4: Performance comparisons over the ED and Stock Prices datasets.

information to make a better decision. It can be expected that higher CW values will be helpful when the expected detection lag is long. But if the expected detection lag is close to one day, then higher CW values won't be helpful. This is because in this case the attack mostly gets detected on the first day, and the data from previous days do not provide any helpful information.

Fig 6.4(b) shows the AMOC curves for m=1 (considering individual series), with different values of CW. We see that for large (>70) false positive rate, CW = 1 performs best. But, the portion of the curves that correspond to lower false positive rates show that higher CW values perform better. Most applications in practice, including disease detection require a very low false positive rate. Hence having a larger CW value is preferable in these conditions.

Computational Speedup

Table 6.1 gives an indication of the advantage of using a lower bound on the standard deviation of the composite series. The first column 'Num Series Considered' corresponds to the number of composite time series that are tested for anomaly over the entire time

Table 6.1: Number of instances that required exact calculation of σ in the Emergency Department Dataset

	Num Series considered	Num Calculated
m = 2	93,923	886
m = 3	428,571	5,587

period. The column 'Num Calculated' corresponds to the cases where we actually needed to perform the exact computation of σ. We see that for m = 2 and 3, we need to perform the expensive computation of σ in only a small fraction of the cases considered.

6.9.3 Stock Prices Dataset

The AMOC curves for this dataset are shown in Fig 6.4(a). We see that m = 2 and 3 performs similar or worse than m=1. This is not very surprising since there is a high positive correlation between the variables. As noted earlier, in the presence of positive correlation, considering the summation of two or more series can cause the false positive rate to increase without producing a significant decrease in the detection lag. We see that in this case, when we consider the difference operator, the AMOC curve is significantly better. This shows that the difference operator is able to exploit the positive correlation present in the dataset.

6.10 Conclusions

We show that by using simple arithmetic combinations of time series, we get a simple yet powerful technique of detecting variations in multivariate time series data. We compare this approach against other related methods, and show the performance improvements on real world datasets (injected with synthetic events). One of the main advantages of our method is its easy interpretability of the results.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

In this thesis we consider the problem of detecting anomalies in large datasets. A main focus is on detecting collective anomalies (e.g. *groups, patterns* or *space-time regions*). In Chapters 2-4, we focus on categorical valued datasets. We start with the investigation of detecting individual record anomalies, and propose a novel way of testing records by considering all possible combinations of attribute values (Chapter 2). We show that this method is especially useful when some of the attributes have a very high arity, and when many of the attribute values are rare.

We then consider the problem of detecting anomalous groups of records which are generated by a common process. Chapters 3 and 4 address two possible variations of this scenario. In Chapter 3 we describe the Anomaly Pattern Detection (APD) algorithm, which assumes that there is some self-similarity (may be low) among the anomalous records, and that they are sufficiently anomalous to stand out by themselves. In this scenario, we use a local anomaly detector, followed by a rule based pattern detector to identify the anomalous records. We show that this approach performs better than either of its component methods. In Chapter 4 we consider the alternate scenario, where the anomalous groups of records are strongly self-similar, but each individual record might not be anomalous on their own.

In this case, we use ideas from spatial scan statistics and the Bayesian network probability model, to develop the Anomalous Group Detection (AGD) algorithm. We evaluate the performances of APD and AGD on real world container shipment, emergency department and network intrusion datasets. A common feature of these algorithms is that they do not assume any contextual information (§1.1.2), but rather perform a combinatorial search over the space of all possible subsets (of attributes or of records). We employ various techniques to perform the search efficiently. Also, since these methods make few assumptions about the data, they are very general algorithms and can be applied to data from a wide variety of domains.

Next, we investigate the problem of learning and detection of time varying space-time patterns in data. In Chapter 5 we generalize the state of art technique - Multivariate Bayesian Scan Statistics (MBSS) to detect time varying events. We use this Time Varying - Multivariate Bayesian Scan Statistics (TV-MBSS) method on over the counter medication sales data to learn and identify space time regions affected by hurricane strikes thus enabling us to differentiate between changes in patterns of behavior due to inclement weather and those which may be due to outbreaks of disease. Finally, in Chapter 6 we consider a simple yet powerful technique of arithmetic combination of time series to detect increase in count in multivariate time series data. We compare this method with a host of other related methods, and show that it outperforms all of them.

7.2 Future Work

- The algorithms in Chapter 2, 3 and 4 exclusively deal with categorical valued datasets. Real valued attributes are discretized into a fixed number of quantile ranges as a pre-processing step. But by discretizing the values we lose some information, such as the ordering of values.

  Currently, we have a fixed number of levels for discretization. It is possible that different real attributes have varying characteristics, and discretizing into the same number of levels is not the best solution. We can use different clustering techniques to determine appropriate levels. k-Means clustering [MacQueen, 1967], SAX [Lin

et al., 2003] and Gaussian Mixture Models [Morchen and Ultsch, 2005] are candidate techniques that will be evaluated for this purpose.

- While in our Bayesian network methods we exclusively deal with categorical valued attributes, we can easily generalize them to handle datasets containing real valued attributes as well, using Bayesian Network models containing both categorical and real valued nodes as shown in [Monti and Cooper, 1998] and [Monti, 1999].

- None of the datasets used for evaluation in this work have labeled outbreaks or anomalies. For the Emergency Department datasets, while we believe that the chosen BARD outbreak simulation is a highly realistic model of anthrax release, for a more robust analysis, we need to evaluate our methods on real, known disease outbreaks. Similarly, in the domains of detecting illegal container shipment, network intrusion detection and over the counter medication sales monitoring, we hope to evaluate our algorithms on naturally occurring events in the data.

- For the time series detection method in Chapter 6, apart from using addition and subtraction, other arithmetic operations such as division can be used to create composite series. In this case, we need to find an efficient way to compute the standard deviation of the composite series since the combinations would no longer be linear.

- The main advantage of our time series detection method is the easy interpretability of an alert. But, not all combinations of time series are meaningful to the end user. We can have an user interface that can specify which combinations to consider. Alternatively, it might be possible to learn meaningful combinations through a more interactive system.

Bibliography

R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, pages 487–499, 1994.

Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, 1993.

E. Aleskerov, B. Freisleben, , and B. Rao. Cardwatch: A neural network based database mining system for credit card fraud detection. In *Proceedings of IEEE Computational Intelligence for Financial Engineering*, pages 220–226, 1997.

M. Augusteijn and B. Folkert. Neural network classication and novelty detection. *International Journal on Remote Sensing*, 23(14):2891–2902, 2002.

M.-A. Balderas, F. Berzal, J.-C. Cubero, E. Eisman, and N. Marn. Discovering hidden association rules. In *Proc. International Workshop on Data Mining Methods for Anomaly Detection (KDD 05)*, 2005.

H. B. Barlow. Unsupervised learning. In *Neural Computation*, volume 1, page 295311, 1989.

Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society Series B*, 57: 289–300, 1995.

R. Borisyuk, M. Denham, F. Hoppensteadt, Y. Kazanovich, and O. Vinogradova. An oscillatory neural network model of sparse distributed memory and novelty detection. In *BioSystems*, pages 265–272, 2000.

George Box, Gwilym M. Jenkins, and Gregory Reinsel. *Time Series Analysis: Forecasting and Control*. Prentice Hall, 3 edition, 1994.

S. Brin, R. Motwani, J. D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket data. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 265–276, 1997.

A. Bronstein, J. Das, M. Duro, R. Friedrich, G. Kleyner, M. Mueller, S. Singhal, and I. Cohen. Bayesian networks for detecting anomalies in internet-based services. In *Intl. Symposium on Integrated Network Mgmt.*, 2001.

H. Burkom, J. Coberly, S. Murphy, Y. Elbert, and K. Hurt-Mullen. Public health monitoring tools for multiple data streams. In *Proceedings of the 2004 National Syndromic Surveillance Conference*, 2004.

P. K. Chan, M. V. Mahoney, and M. H. Arshad. A machine learning approach to anomaly detection, technical report cs-2003-06. Technical report, Department of Computer Sciences, Florida Institute of Technology, 2006.

D. Chen, X. Shao, B. Hu, and Q. Su. Simultaneous wavelength selection and outlier detection in multivariate regression of near-infrared spectra. *Analytical Sciences*, 21(2): 161–167, 2005.

G.F. Cooper, D.H. Dash, J.D. Levander, W.K. Wong, W.R. Hogan, and M.M. Wagner. Bayesian biosurveillance of disease outbreaks. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 94–103, 2004.

G.F. Cooper, J.N. Dowling, J.D. Levander, and P. Sutovsky. A bayesian algorithm for detecting cdc category a outbreak diseases from emergency dept chief complaints. In *Proceedings of the National Syndromic Surveillance Conference*, 2006.

R. B. Crosier. Multivariate generalizations of cumulative sum quality-control schemes. *Technometrics*, 30:291–303, 1988.

Kaustav Das and Jeff Schneider. Detecting anomalous records in categorical datasets. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug 2007.

Kaustav Das, Andrew Moore, and Jeff Schneider. Belief state approaches to signaling alarms in surveillance systems. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2004a.

Kaustav Das, Andrew Moore, and Jeff Schneider. Early detection of insider trading in option markets. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 420–429, 2004b.

Kaustav Das, Jeff Schneider, and Daniel Neill. Anomaly pattern detection in categorical datasets. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug 2008.

Kaustav Das, Jeff Schneider, and Daniel Neill. Detecting anomalous groups in categorical datasets. In *Submitted to the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2009.

P. J. Denning. Working sets past and present. In *IEEE Transactions on Software Engineering*, volume 6, 1980.

Shih Dong-Her, Chiang Hsiu-Sen, Chan Chun-Yuan, and Binshan Lin. Internet security: malicious e-mails detection and protection. *Industrial Mgmt. and Data Sys.*, 104:613 – 623, Sep 2004.

S. Edelman, B. P. Hiles, H. Yang, and N. Intrator. Probabilistic principles in unsupervised learning of visual structure: human data and a model. In *Advances in Neural Information Processing Systems 14*, 2002.

Eleazar Eskin. Anomaly detection over noisy data using learned probability distributions. In *Proc. 17th International Conf. on Machine Learning*, pages 255–262. Morgan Kaufmann, San Francisco, CA, 2000.

Eleazar Eskin, Andrew Arnold, Michael Prerau, Leonid Portnoy, and Sal Stolfo. A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data. In *Applications of Data Mining in Computer Security*, 2002.

J. H. Friedman and N. I. Fisher. Bump hunting in high-dimensional data. *Statistics and Computing*, 9(2):123–143, 1999.

A. Ghosh and A. Schwartzbard. A study in using neural networks for anomaly and misuse detection. In *In Proceedings of the 8th USENIX Security Symposium*, 1999.

J. Glaz and N. Balakrishnan. *Scan Statistics and Applications*. Birkhauser, 1999.

A. Goldenberg, G. Shmueli, A.R. Caruana, and E.S. Fienberg. Early statistical detection of anthrax outbreaksby tracking over-the-counter medication sales. In *Proceedings of the National Academy of Sciences*, volume 99, pages 5237 – 5240, 2002.

P. Good. *Permutation Tests - A Practical Guide to Resampling Methods for Testing Hypotheses.* Springer-Verlag, 2nd edition edition, 2000.

J. D. Hamilton. *Time Series Analysis*. Princeton University Press, 1994.

K.A. Heller, K.M. Svore, A. Keromytis, and S.J. Stolfo. One class support vector machines for detecting anomalous windows registry accesses. In *Proc. of the workshop on Data Mining for Computer Security*, 2003.

P. Helman and J. Bhangoo. A statistically base system for prioritizing information exploration under uncertainty. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 27(4):449–466, 1997.

S. A. Hofmeyr, Stephanie Forrest, and A. Somayaji. Intrusion detect using sequences of system calls. In *Journal of Computer Security*, volume 6, pages 151–180, 1998.

William R. Hogan, Gregory F. Cooper, Garrick L. Wallstrom, Michael M. Wagner, and Jean-Marc Depinay. The bayesian aerosol release detector: An algorithm for detecting and characterizing outbreaks caused by an atmospheric release of bacillus anthracis. *Statistics in Medicine*, 26:5225–5252, Sep 2007.

B. Hong and M. Hardin. A report of the properties of the multivariate forecast-based processing scheme. In *Proceedings of the Joint Statistical Meetings; Toronto, Canada: American Statistical Association*, Aug 2004.

H. Hotelling. *Techniques of Statistical Analysis*. New York: McGraw-Hill, 1947.

L. Hutwagner, W. Thompspn, G. M. Seeman, and T. Treadwell. The bioterrorism preparedness and response early aberration reporting system(ears). *Journal of Urban Health*, 80:i89–i96, 2003.

L. C. Hutwagner, E. Maloney, N. H. Bean, L. Slutsker, and S. Martin. Using laboratory-based surveillance data for prevention: An algorithm for detecting salmonella outbreaks. *Emerging Infectious Diseases*, 3:395–400, 1997.

P. Jaccard. The distribution of flora in the alpine zone. *The New Phytologist*, 11(2):37–50, 1912.

X. Jiang. *A Bayesian Network Model for Spatio-Temporal Event Surveillance*. PhD thesis, University of Pittsburgh, Department of Biomedical Informatics, 2008.

Xia Jiang, Michael M. Wagner, and Gregory F. Cooper. *Modeling the Temporal Trend of the Daily Severity of an Outbreak Using Bayesian Networks*, volume 156 of *Studies in Computational Intelligence*. Springer, 2008.

KDDCup. The third international knowledge discovery and data mining tools competition, kdd cup 1999. In *The Fifth International Conference on Knowledge Discovery and Data Mining*, 1999.

Eamonn Keogh, Stefano Lonardi, and Bill Chiu. Finding surprising patterns in a time series database in linear time and space. In *Proc. ACM Knowledge Discovery and Data Mining*, pages 550–556, 2002a.

Eamonn J. Keogh, Stefano Lonardi, and Bill Yuan chi Chiu. Finding surprising patterns in a time series database in linear time and space. In *KDD*, pages 550–556, 2002b.

M. Kulldorff. A spatial scan statistic. *Communications in Statistics: Theory and Methods*, pages 1481–1496, 1997.

M. Kulldorff. Prospective time-periodic geographical disease surveillance using a scan statistic. *Journal of the Royal Statistical Society A*, 164:61–72, 2001.

M. Kulldorff and N. Nagarwalla. Spatial disease clusters: detection and inference. *Statistics in Medicine*, 14:799–810, 1995.

Wenke Lee and Salvatore Stolfo. Data mining approaches for intrusion detection. In *Proceedings of the 7th USENIX Security Symposium*, 1998.

Kingsly Leung and Christopher Leckie. Unsupervised anomaly detection in network intrusion detection using clusters. In *Proc. 28th Australasian CS Conf.*, volume 38 of *CRPITV*, 2005.

Kun-Lun Li, Hou-Kuan Huang, Sheng-Feng Tian, and Wei Xu. Improving one-class svm for anomaly detection. In *Proc. of International Conference on Machine Learning and Cybernetics*, 2003.

J. Lin, E. Keogh, S. Lonardi, , and B. Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of 8th ACM SIGMOD, DMKD workshop*, pages 2–11, 2003.

J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, 1967.

Maxim Makatchev and Daniel Neill. Learning outbreak regions in bayesian spatial scan statistics. In *Proceedings of the ICML/UAI/COLT Workshop on Machine Learning for Health Care Applications*, 2008.

D.C. Montgomery. *Introduction to Statistical Quality Control third ed.* John Wiley and Sons, 1996.

S. Monti. *Learning hybrid bayesian networks from data*. PhD thesis, University of Pittsburgh, Intelligent Systems Program, 1999.

S. Monti and G. F. Cooper. A multivariate discretization method for learning bayesian networks from mixed data. In *Proceedings of Conference on Uncertainty in Artificial Intelligence*, page 40441, 1998.

Andrew Moore and Mary Soon Lee. Cached sufficient statistics for efficient machine learning with large datasets. *Journal of Artificial Intelligence Research*, 8:67–91, March 1998.

Andrew Moore and Weng-Keen Wong. Optimal reinsertion: A new search operator for accelerated and more accurate bayesian network structure learning. In *20th Intl. Conf. on Machine Learning*, pages 552–559, Aug 2003.

Andrew Moore, Greg Cooper, Rich Tsui, and Mike Wagner. Summary of biosurveillance-relevant technologies, February 2002. URL `http://www.cs.cmu.edu/~awm/biosurv-methods.pdf`.

Fabian Morchen and Alfred Ultsch. Optimizing time series discretization for knowledge discovery. In *Proceedings of the 11th ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 660–665, 2005.

D. B. Neill, A. W. Moore, M. R. Sabhnani, and K. Daniel. Detection of emerging space-time clusters. In *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2005a.

Daniel Neill and Andrew Moore. Rapid detection of significant spatial clusters. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, August 2004.

Daniel Neill, Andrew Moore, and Gregory Cooper. A bayesian spatial scan statistic. In et al. Y. Weiss, editor, *Advances in Neural Information Processing Systems*, volume 18, pages 1003–1010, 2005b.

Daniel B. Neill and Andrew W. Moore. Anomalous spatial cluster detection. In *Proceedings of the KDD 2005 Workshop on Data Mining Methods for Anomaly Detection*, August 2005.

Daniel B. Neill, Andrew W. Moore, Francisco Pereira, and Tom Mitchell. Detecting significant multidimensional spatial clusters. In *Advances in Neural Information Processing Systems*, volume 17, pages 869–876, 2005c.

D.B. Neill and G.F. Cooper. A multivariate bayesian scan statistic for early event detection and characterization. *Machine Learning*, 2009.

D.B. Neill, A.W. Moore, and G.F. Cooper. A multivariate bayesian scan statistic. *Advances in Disease Surveillance*, 2(60), 2007.

NHC. National hurricane center. URL `http://www.nhc.noaa.gov/`.

I. Yamada P. A. Rogerson. Monitoring change in spatial patterns of disease: comparing univariate and multivariate cumulative sum approaches. *Statistics in Medicine*, 23:195–214, 2004.

E. S. Page. Continuous inspection scheme. *Biometrika*, 41:100115, 1954.

P. Patel, E.Keogh, J.Lin, and S.Lonardi. Mining motifs in massive time series databases. In *Proceedings of IEEE International Conference on Data Mining (ICDM'02)*, pages 370–377, December 2002.

Dan Pelleg. *Scalable and Practical Probability Density Estimators for Scientific Anomaly Detection*. PhD thesis, Carnegie Mellon University, 2004.

J. J. Pignatiello and G. C. Runger. Comparisons of multivariate cusum charts. *J Qual Technol*, 22:173–186, 1990.

R. F. Raubertas. An analysis of disease surveillance data that uses the geographic locations of reporting units. *Statistics in Medicine*, 8:267–271, 1989.

P. A. Rogerson. Surveillance systems for monitoring the development of spatial patterns. *Statistics in Medicine*, 16:2081–2093, 1997.

Liaquat M. Sheikh, Basit Tanveer, and Syed M. A. Hamdani. Interesting measures for mining association rules. In *Proceedings of the 8th IEEE International Multitopic Conference*, pages 641– 644, 2004.

S. Shekhar, C.T. Lu, and P. Zhang. Detecting graph-based spatial outliers: algorithms and applications (a summary of results). In *Proceedings of the 7th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 371–376, 2001.

T. Shon, Y. Kim, C. Lee, and J. Moon. A machine learning framework for network anomaly detection using svm and ga. In *Proc. from the Sixth Annual IEEE Systems, Man and Cybernetics (SMC) Information Assurance Workshop*, pages 176– 183, 2005.

X. Song, M. Wu, C. Jermaine, and S. Ranka. Conditional anomaly detection. *IEEE Transactions on Knowledge and Data Engineering*, 19(5):631–645, 2007.

M. Stoto. Multivariate methods for aberration detection: a simulation report using the district of columbia's syndromic surveillance data. In *Proceedings of the 2004 National Syndromic Surveillance Conference*, 2004.

J. Theiler and D. M. Cai. Resampling approach for anomaly detection in multispectral images. In *Proceedings of SPIE Volume 5093*, pages 230–240, 2003.

H. E. Tillett and I. L. Spencer. Influenza surveillance in england and wales using routine statistics. *Journal of Hygine*, 88:83–94, 1982.

Christina Warrender, Stephanie Forrest, and Barak A. Pearlmutter. Detecting intrusions using system calls: Alternative data models. In *IEEE Symposium on Security and Privacy*, pages 133–145, 1999.

B. Whitehead and W. Hoyt. A function approximation approach to anomaly detection in propulsion system test data. In *Proc. AIAA/SAE/ASME/ASEE 29th Joint Propulsion Conference*, 1993.

Weng-Keen Wong. *Data Mining for Early Disease Outbreak Detection*. PhD thesis, Carnegie Mellon University, 2004.

Weng-Keen Wong, Andrew Moore, Greg Cooper, and Mike Wagner. Rule-based anomaly pattern detection for detecting disease outbreaks. In *Proceedings of the 18th National Conference on Artificial Intelligence*. MIT Press, 2002.

Weng-Keen Wong, Andrew Moore, Gregory Cooper, and Michael Wagner. Bayesian network anomaly pattern detection for disease outbreaks. In *Twentieth Intl. Conf. on Machine Learning*, pages 808–815, Aug 2003.

Jiong Yang, Wei Wang, and Philip S. Yu. Infominer: mining surprising periodic patterns. In *Knowledge Discovery and Data Mining*, pages 395–400, 2001.

Nong Ye and Mingming Xu. Probabilistic networks with undirected links for anomaly detection. In *IEEE Systems, Man, and Cybernetics Information Assurance and Security Workshop*, pages 175–179, June 2000.