

Anomaly Pattern Detection in Categorical Datasets

Kaustav Das
Machine Learning Department
Carnegie Mellon University
Pittsburgh PA 15203
kaustav@cs.cmu.edu

Jeff Schneider
Machine Learning Department
Carnegie Mellon University
Pittsburgh PA 15203
schneide@cs.cmu.edu

Daniel B. Neill
Heinz School of Public Policy
Carnegie Mellon University
Pittsburgh PA 15203
neill@cs.cmu.edu

ABSTRACT

We propose a new method for detecting 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. 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. 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.

Categories and Subject Descriptors

H.2.8 [Database Management]: Database Applications—*Data Mining*

General Terms

Algorithms, Performance, Experimentation

Keywords

Pattern Detection, Anomaly Detection, Machine Learning

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

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

KDD'08, August 24–27, 2008, Las Vegas, Nevada, USA.
Copyright 2008 ACM 978-1-60558-193-4/08/08 ...\$5.00.

1. INTRODUCTION

We consider the problem of detecting patterns of anomalies in large, multidimensional datasets. This problem has received considerable attention, particularly in the fields of network intrusion detection [14, 8, 6] and the monitoring of public health data for disease outbreak detection [15, 13]. Anomalous pattern detection techniques can also be applied in astrophysical discovery, where astronomers want to detect new and interesting space objects, and in many other application domains.

In general, anomalies can be defined as any observations that are different from the *normal* behavior of the data. However, rather than finding individually anomalous records (which may be due to noise), we are interested in detecting the emergence of new phenomena resulting in patterns of anomalous observations that cannot be explained by a previous model. Here we focus on two application domains, disease surveillance and customs monitoring. In the disease surveillance task, we wish to detect causes such as epidemics or bioterrorist attacks which give rise to patterns of unusual emergency department records. In customs monitoring, we are interested in detecting possible illegal activity, e.g. attempts to import unwanted illegal or dangerous material into the country. In each case, we assume that anomalies are generated by some underlying process which creates records with attribute values that are unexpected given the normal behavior of the data. In addition, 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 subset of their attributes. In this work, we develop a new detection method that can efficiently and accurately detect such patterns.

Many traditional anomaly detection techniques look at the data records individually, and try to determine whether each record is anomalous with respect to the historical distribution of data. Here we consider two such techniques, using a Bayesian Network likelihood model and a conditional anomaly detection method [4] respectively. While such methods of *local anomaly detection* can be used to de-

tect individually 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 our detection performance.

“What’s Strange About Recent Events” (WSARE) [16] 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.

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 [16] 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.

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. A Bayesian network is a popular representation of a probability model over the attributes for categorical data because of its parsimonious use of parameters, and efficient learning and inference techniques. Bayes Nets have been used for detecting anomalies in network intrusion detection [3, 17], detecting malicious emails [5] and disease outbreak detection [15]. A typical anomaly detection approach is to 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. For our experiments, we used the optimal reinsertion algorithm [12] to learn the structure, and then did a maximum likelihood estimation of the network parameters. For testing a record, we compute the likelihood of that record given the Bayesian Network learned from the training data. A lower value of the likelihood indicates that the record is more anomalous. The log-likelihood value is used as the anomalousness score of each record.

Conditional Anomaly Detection. In our previous work [4] we described the conditional anomaly detection method of detecting anomalies in categorical datasets. For any test record t , and any two disjoint sets of attributes A and B , we consider the ratio $r(a_t, b_t) = \frac{P(a_t, b_t)}{P(a_t)P(b_t)}$ where a_t and b_t are the value combinations taken by A and B in t respectively. The probability values are estimated from the corresponding counts in the training dataset. 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. A low value of $r(a_t, b_t)$ also ensures we have seen enough cases of a_t and b_t in the training data to support the hypothesis of negative dependence. A score is then assigned to the record t based on all such r -values corresponding to all possible pairs of attribute sets. The score is defined as the maximum 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), we use $k = 1$ since it has a larger number of attributes.

2.2 WSARE

The WSARE-2 method [16] searches over all possible one or two component 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 only 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 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

	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 1: 2×2 Contingency Table for WSARE

the algorithm (WSARE-3) [15], 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 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) random 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.

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. Since anomalies are usually rare, any such detector may detect many false positives. In order to successfully determine if a subset of the test data has a 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 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 §2.1. In general, any anomaly detector requires baseline or training examples which corre-

Figure 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 a 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 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.

spond 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, 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 j th 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 will not hold, 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 [7] (using Stirling’s approximation to calculate the factorials) on the 2×2 table (Table 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 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 [2], which guarantees that the expected proportion of the number of false positives will be no greater than α . It can be used to find a cutoff threshold to determine which rules are significant. 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.

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 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 $C_0 | C_1$ and $C_1 | C_0$ are both determined to be significant.

In §4 we compare the anomaly detection performance of our method of pattern detection to both WSARE and WSARE-AUG.

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.

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 [11]. 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

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 j th 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. DATASETS

We evaluate the methods on the three datasets described below.

3.1 PIERS Dataset

Our first dataset consists of records describing containers imported into the country from various ports in Asia. Each record consists of 10 attributes. 7 of them are categorical: the container’s country of origin, departing and arriving ports, shipping line, shipper name, vessel name and the commodity being shipped. There are three real valued attributes, the size, weight and value of the container. We have categorized these to five discrete levels. Some of the attributes such as the shipper name and commodity have very high arity in the order of a few thousand.

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.

3.2 Emergency Department Dataset

This real-world dataset contains records of patients visiting emergency departments (ED) from hospitals around Allegheny county in the year 2004. 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 [9] 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 KDD Cup 1999 Network Intrusion Detection Dataset

We have also evaluated AGD on the KDD Cup 1999 data [1], which 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. In total there are 41 features, most of them taking real values. The real 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

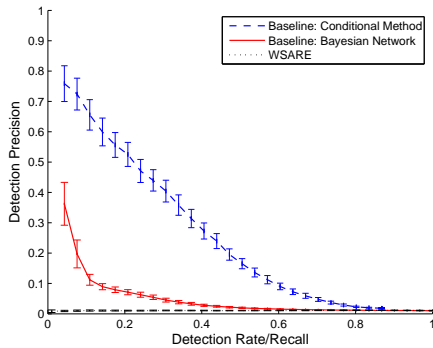


Figure 2: PIERS dataset: Detection precision vs. recall for baseline methods and WSARE, with 95% confidence intervals

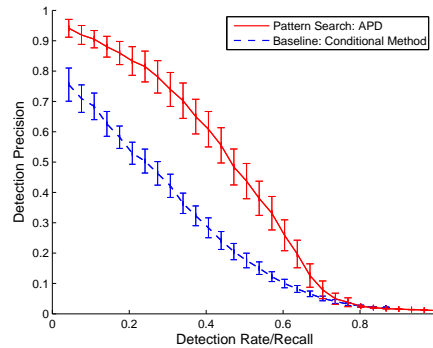


Figure 4: PIERS dataset: Performance comparison between pattern detection and baseline, $PatternRate = 0.2$

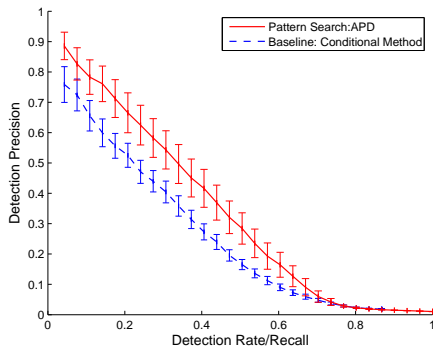


Figure 3: PIERS dataset: Performance comparison between pattern detection and baseline, with 95% confidence intervals, $PatternRate = 0.1$

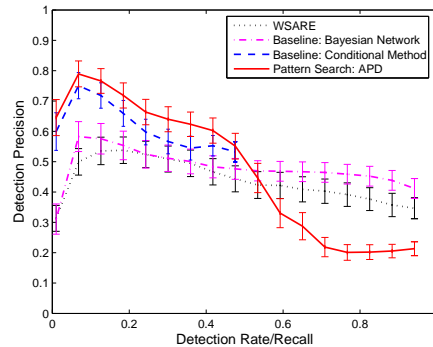


Figure 5: ED Dataset: Detection precision vs. recall for baseline methods and APD

(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, nep-tune, 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.

4. 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 2.3, choosing the better one to use as the base method for pattern detection. We also compare the performance of the related methods: WSARE and WSARE-AUG on these datasets.

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 2 and 3 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 2 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

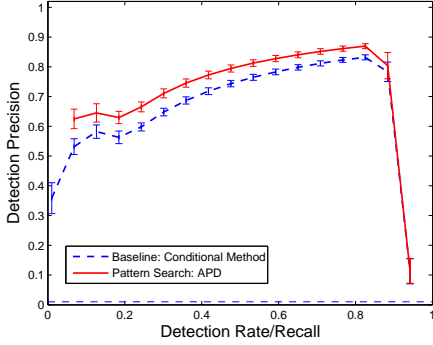


Figure 6: KDD Cup 99: guess password; Performance comparison between pattern detection and baseline

WSARE is almost the same as the nominal 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 (§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 compares the performance of our proposed anomaly pattern detector (APD) with the baseline method of conditional anomaly detection (§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 4 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. And for the higher value of $PatternRate$ we see a greater improvement in performance as expected. We also evaluated the performance of APD 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 each anomalous record 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 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

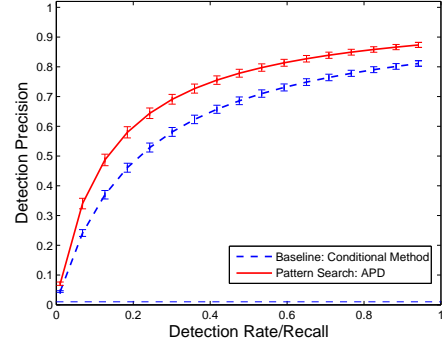


Figure 7: KDD Cup 99: smurf; Performance comparison between pattern detection and baseline

Table 3: Normalized area under the curves for KDD Cup 99 Dataset comparing Baseline and AGD, with 95% CI

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

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 §3.1. 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 [10] 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 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 5 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. [16] 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 extends the curve beyond recall rate 0.5, but decreases the precision rate below the other methods in that range.

Figures 6 and 7 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. 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 smpguess. 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 of precision.

5. CONCLUSION AND FUTURE WORK

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, we also plan to evaluate our methods on real, known disease outbreaks in the future.

6. REFERENCES

- [1] 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.
- [2] 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.
- [3] 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.
- [4] K. Das and J. Schneider. Detecting anomalous records in categorical datasets. In *Proc. ACM Knowledge Discovery and Data Mining*, Aug 2007.
- [5] S. Dong-Her, C. Hsiu-Sen, C. Chun-Yuan, and B. Lin. Internet security: malicious e-mails detection and protection. *Industrial Mgmt. and Data Sys.*, 104:613 – 623, Sep 2004.
- [6] E. 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.
- [7] P. Good. *Permutation Tests - A Practical Guide to Resampling Methods for Testing Hypotheses*. Springer-Verlag, 2nd edition, 2000.
- [8] P. Helman and J. Bhango. A statistically base system for prioritizing information exploration under uncertainty. In *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, volume 27(4), pages 449–466, 1997.
- [9] W. R. Hogan, G. F. Cooper, G. L. Wallstrom, M. M. Wagner, and J.-M. 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.
- [10] P. Jaccard. The distribution of flora in the alpine zone. *The New Phytologist*, 11(2):37–50, 1912.
- [11] A. Moore and M. S. Lee. Cached sufficient statistics for efficient machine learning with large datasets. *Journal of Artificial Intelligence Research*, 8:67–91, March 1998.
- [12] A. Moore and W.-K. 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.
- [13] D. B. Neill, A. W. Moore, F. Pereira, and T. Mitchell. Detecting significant multidimensional spatial clusters. In *Advances in Neural Information Processing Systems*, volume 17, pages 869–876, 2005.
- [14] C. Warrender, S. Forrest, and B. A. Pearlmutter. Detecting intrusions using system calls: Alternative data models. In *IEEE Symposium on Security and Privacy*, pages 133–145, 1999.
- [15] W. K. Wong, A. Moore, G. Cooper, and M. Wagner. Bayesian network anomaly pattern detection for disease outbreaks. In *Twentieth Intl. Conf. on Machine Learning*, pages 808–815, Aug 2003.
- [16] W. K. Wong, A. W. Moore, G. Cooper, and M. Wagner. Rule-based anomaly pattern detection for detecting disease outbreaks. In *Proceedings of the 18th National Conference on Artificial Intelligence*. MIT Press, 2002.
- [17] N. Ye and M. 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.