Detecting Anomalous Records in Categorical Datasets

Kaustav Das
Machine Learning Department
Carnegie Mellon University
Pittsburgh PA 15203

Jeff Schneider
Machine Learning Department
Carnegie Mellon University
Pittsburgh PA 15203

ABSTRACT

We consider the problem of detecting anomalies in high arity categorical datasets. In most applications, anomalies are defined as data points that are 'abnormal'. Quite often we have access to data which consists mostly of normal records, along with a small percentage of unlabelled anomalous records. We are interested in the problem of *unsupervised anomaly detection*, where we use the unlabelled data for training, and detect records that do not follow the definition of normality.

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, bayes nets 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 over semi-synthetic as well as real world datasets.

Categories and Subject Descriptors: H.2.8 [Database Management]: Database Applications - Data Mining

General Terms: Algorithms, Performance, Experimentation

Keywords: Anomaly Detection, Machine Learning

1. INTRODUCTION

With the ever increasing amount of data being collected universally, it gets more important and challenging to spot unusual or unexpected observations. Anomaly detection aims to address this issue from a statistical datamining framework.

Anomalies can be defined as anything that is 'different' from 'normal' behavior. Hence, we need to first define 'normality'. Usually this is specified as a probability model from which observations are assumed to be drawn. We also need a similarity measure to compare the observations with respect to the model.

Detecting anomalies in observations is important in many different respects. A traditional form of anomaly detection is in industrial process control. Time-series data from various sensors are monitored to detect out of control processes. In most applications, we are interested in the detection of emergence of new phenomena unexplained by the previous model. For example, in astronomical datasets, astronomers want to detect new interesting space objects. In applications such as bio-surveillance and customs monitoring, detecting suspicious activity in high dimensional data is the goal of anomaly detection. Anomaly detection can also be used to automatically detect data entry errors, which can then be corrected.

However, forming a general framework for anomaly detection is a difficult challenge as the definition of normality is typically very domain specific. This has led to independent efforts for various domains.

First, we give a summary of the related work. We then 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.

2. RELATED WORK

Anomaly detection applied to network intrusion detection has been an active area of research since it was proposed by Denning [28]. Traditional anomaly detection approaches build models of normal data and detect deviations from the normal model in observed data. A survey of these techniques is given in [31]. 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 [16], and a statistical method to determine frequent sequences in intrusion data is presented in [15]. Lee *et al.* [21] uses a decision tree model over normal data and

This work is funded in part by CDC under award 8-R01-HK000020-02 and by NSF under award IIS-0325581.

KDD'07, August 12–15, 2007, San Jose, California, USA.

Ghosh *et al.* [13] uses a neural network to obtain the model. Eskin [12] 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 detecting anomalies in a dataset is used in [22] and [11]. One-class SVMs [23, 14] and Genetic Algorithms [29] have also been used to classify anomalies in this context.

Anomaly detection is also commonly applied in time series data to detect unusual fluctuations compared to past data points [4, 18, 7, 26]. Another area of considerable recent interest is spatial anomaly detection [19].

The methods described so far apply to real valued data or work in a supervised setting when we have labeled training data. We now describe methods that apply to the problem of interest, i.e., on categorical datasets in an unsupervised setting.

2.1 Unsupervised Methods Applied to Categorical Datasets

2.1.1 Association Rule Based Approaches

The task of association rule mining has received considerable attention especially, in the case of market basket analysis [3]. 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. 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.* [8] have 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 $P(!Y|X)$, 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. We have used this method as one of the baseline methods for comparison of our algorithms.

Balderas *et al.* [5] mine hidden association rules, or rules that are not common, but confident. Such rules are assumed to represent the rare anomaly class.

WSARE developed by Wong *et al.* [30] also uses rules to identify anomalies. But in this case, the rules are learnt from a historical dataset, and are applied on a collection of records from the current time interval, to detect unusual counts of various cases.

2.1.2 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 anomalous. 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 [27]. We choose a bayesian network as the standard model against which we compare our algorithm. Hence, we give an overview of this method next.

2.1.3 Anomaly Detection Using Bayes Network

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 Net have been used for detecting anomalies in network intrusion detection [2, 33], detecting malicious emails [9] and disease outbreak detection [32]. Any good structure and parameter learning algorithm is appropriate to learn the model. For our experiments, we used the optimal reinsertion algorithm [25] 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.

Training

1. Construct a *conditional AD Tree* over the training dataset (§3.1.6).
2. Determine the dependence between all attribute sets up to size k by computing the mutual information between them (§3.1).
3. Construct the cache for denominator counts (§3.2.2).

Testing: Scoring a test record t

1. For each *mutually exclusive* and *dependent* pair of attribute sets A and B (§3.1):
   (a) Compute $r(a_t, b_t)$ (§3.1).
2. Compute the overall *conditional* score of the record t from all the r-values calculated above (§3.1.2).

Figure 1: Conditional Anomaly Test Algorithm.

Training

1. Construct a *marginal AD Tree* over the dataset (§3.3.1).
2. Compute the marginal count histograms over the training data (§3.3.1).

Testing: Scoring a test record t

1. For each attribute set A with up to k attributes:
   (a) Compute $qval(a_t)$ (§3.3).
2. Compute the overall *marginal* score of the record t as the minimum q-value calculated above (§3.3).

Figure 2: Marginal Anomaly Test Algorithm.

3. 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.

Figures 1 and 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.

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.

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

Table 1: Features in Piers Dataset

3.1 Conditional Probability Tests

We will illustrate a problem 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.

Consider a particular test record t and the attributes *ShipperName* and *Country*. We define $P(SN_t, C_t)$ = *P(ShipperName* $=SN_t$*, 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 §3.2.1.

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,

$$\mu(A, B) \geq \beta_\mu \quad (1)$$

where, β_μ is a threshold parameter, set to 0.1 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 $\mathrm{r} = \frac{P(A,B)}{P(A)P(B)}$ has been proposed as a measure of *suspicious coincidence* by Barlow [6]. 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 ex-

pected 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 [10]. Here 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.

3.1.1 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.

3.1.2 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.

$$\begin{aligned} r(a_t, b_t) \times r(c_t, d_t) &= \frac{P(a_t, b_t)P(c_t, d_t)}{P(a_t)P(b_t)P(c_t)P(d_t)} \\ &= \frac{P(a_t, b_t, c_t, d_t)}{P(a_t, c_t)P(b_t, d_t)} \\ &= r([a_t, c_t], [b_t, d_t]) \end{aligned} \quad (2)$$

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.

3.1.3 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.

3.1.4 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'.

$$P(Data|p_A) = Binomial(N, p_A) \quad (3)$$

$$P(p_A|Data) = \frac{P(Data|p_A) * P(p_A)}{P(Data)} \quad (4)$$

$$P(p_A|Data) \sim p_A^{C(a_t)}(1 - p_A)^{N-C(a_t)} \quad (5)$$

$$P(p_A|Data) \sim Beta(C(a_t) + 1, N - C(a_t) + 1) \quad (6)$$

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

3.1.5 Bound on the counts

From eqn. 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 §3.2.2. To make this caching tractable, we compute a lower bound for $C(a_t)$ and $C(b_t)$.

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

$$\Longrightarrow \frac{C(a_t,b_t)+1}{N+2} \times \frac{N+2}{C(a_t)+1} \times \frac{N+2}{C(b_t)+1} \leq \alpha$$

$$\Longrightarrow \frac{C(a_t,b_t)+1}{N+2} \times \frac{N+2}{C(a_t)+1} < \alpha$$

[because, $(N+2) > (C(b_t)+1)$]

$$\Longrightarrow \frac{C(a_t,b_t)+1}{C(a_t)+1} < \alpha$$

$$\Longrightarrow \frac{1}{C(a_t)+1} < \alpha$$

[because, $C(a_t, b_t) \geq 0$]

$$\Longrightarrow C(a_t) > \frac{1}{\alpha} - 1 \quad (7)$$

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.

3.1.6 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 [24]. 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 trade-off 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 $2k$.

3.2 Computational Speedup

3.2.1 Reducing arity

The memory required to build an AD Tree significantly depends on the arity of the attributes. We use the result from eqn. 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.

3.2.2 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 (§3.2) 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.

3.3 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.

$$qval(a_t) = \sum_{x \in X} P(x) \text{ where, } X \equiv \{x : P(x) \leq P(a_t)\} \quad (8)$$

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.

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.

3.3.1 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 [24].

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.

4. EXPERIMENTAL SETUP

4.1 Datasets

4.1.1 Piers Dataset

Our first dataset consists of records describing containers imported into the country. Each record consists of 10 attributes. Most of the attributes are categorical, such as the country of origin, the departing and arriving ports, Shipping line etc. There are three real valued attributes, the size, weight and value of the container. We have categorized these to five discrete levels.

Since there were no labels 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 marginals. The 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.

4.1.2 KDD Cup 99 Network Connections Dataset

We have used a network connection records dataset from KDD Cup 1999 [1], which contained 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 included 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 were obtained using some domain knowledge, and included the number of file creation operations, number of failed login attempts, whether root shell was obtained, and others. Finally there were a number of features computed using a two second time window. These included 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 were discretized to 5 levels.

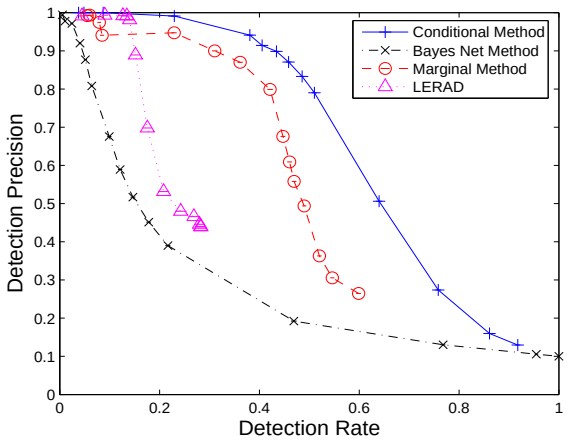
The goal of the KDD dataset was to produce a good training set for learning methods that use labeled data. Hence, the proportion of attack instances to normal ones is very large. To create more realistic data, we have reduced the number of attack records to about 10% of the test dataset. There are a 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.

4.2 Training

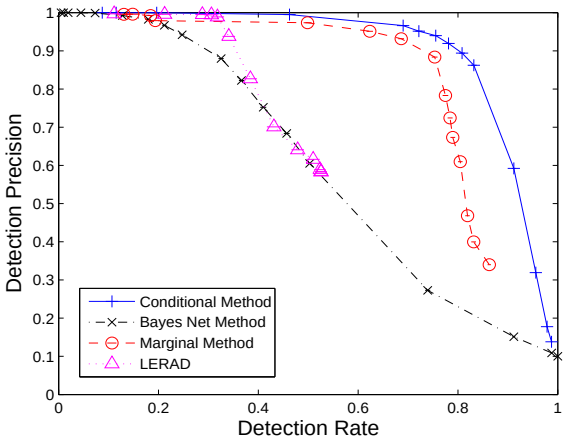
We build our model, which includes the conditional AD Tree, the marginal AD Tree, the mutual information matrix, cache for the denominator counts §3.2.2 and the marginal count histograms using the training data. Building these comprise the training phase.

4.3 Testing

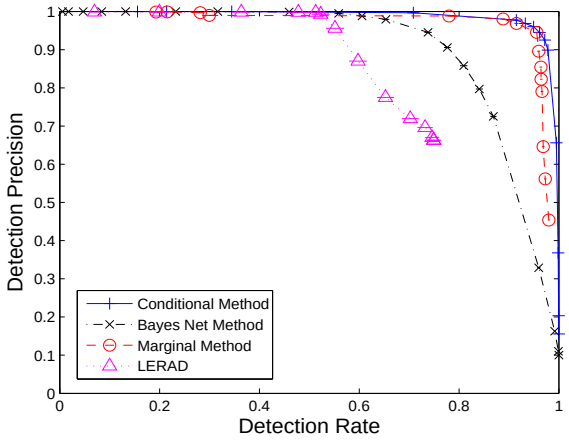
For each test record t, we consider every possible pair of composite attributes, that are *mutually exclusive* and *depen-*



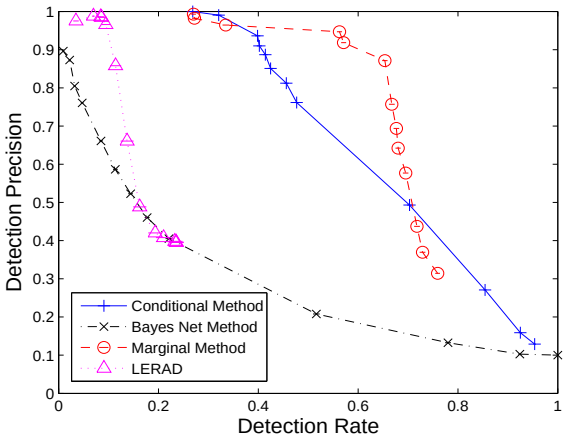
(a) Algorithm performances for $l = 1$



(b) Algorithm performances for $l = 3$



(c) Algorithm performances for $l = 7$



(d) Algorithm performances for inserted records from different month

Figure 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.

dent (see eqn.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 (§3.1.2) to assign score to a record. For the KDD dataset, we have also considered the partitioning method described in §3.1.1. Here we consider all possible *mutually exclusive* subsets A, B and C to compute the ratio $rval(a_t, b_t|c_t)$.

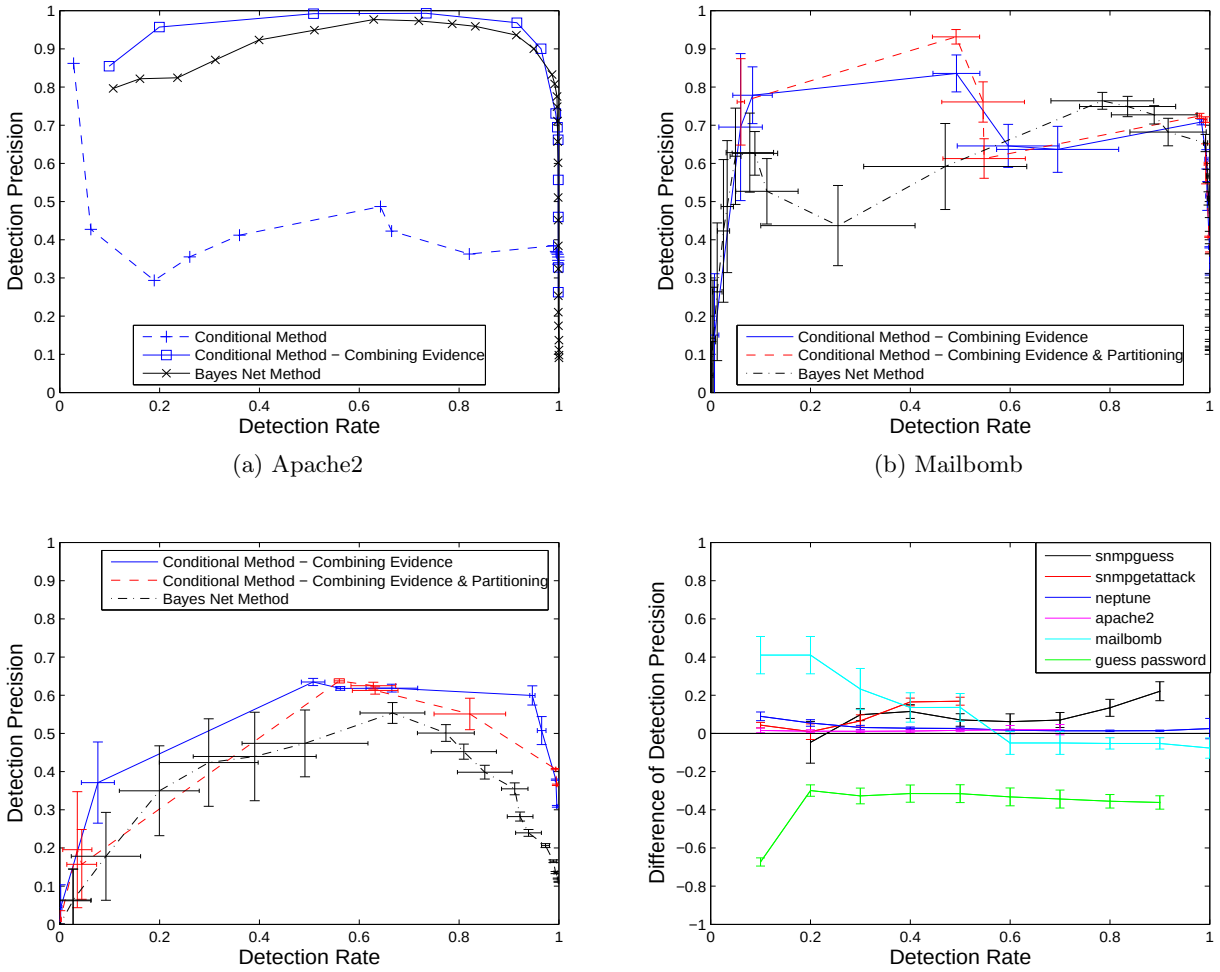
4.4 Evaluation

We evaluate our methods against a likelihood based approach using a bayes network representation and association rule based learner LERAD [8]. 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 bayes 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.

5. RESULTS

5.1 Containers Dataset

In Figure 3 we show the comparison our methods (conditional and marginal) against the bayes net likelihood method and LERAD [8] on the CBP 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.



(a) Apache2

(b) Mailbomb

(c) Snmpguess

(d) Comparison for all attack types

Figure 4: Performance over the Network Connections KDD Cup 99 dataset

In Figure 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 3(b) and 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 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 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.

5.2 KDD Cup 99 Network Connections Dataset

On the network connections dataset, we see that some attack types are easier to detect than others. Figure 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 (§3.1.2) from different attribute sets. Here, we have also compared the performance of the partitioning method §3.1.1.

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

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: Time and Space requirement for Bayes Network Method

Dataset	Training Size	Test Size	Number of Attributes	k	Training Time (secs)	Testing Time (secs)	Memory (MB)	Marginal Memory (MB)
Piers	500,000	10,000	10	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	41	1	10.2	15	323	222
				2	44	7145	332	2618

Table 3: Time and Space requirement for Conditional and Marginal Methods

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 4(a), the original conditional method performs worse than the bayes net 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 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.

6. FUTURE WORK

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 especially 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.

An important challenge here is to define what can be considered as a group. We need to specify a similarity measure, and group records on the basis of it. If the data has temporal and/or spatial components, they provide a natural measure for grouping. For temporal analysis, we propose to group records on the basis of a temporal unit such as a day. We can extend the method of comparison of marginal and conditional probability distributions of all records in the current day, to the historical data. Similarly, for a spatial analysis, grouping can be predetermined such as by zip code or area. It can also be computed dynamically similar to spatial scan. Apart from this, we can also use the association based dissimilarity measures such as methods presented in [20, 17] for grouping records.

Another possible improvement is the way we deal with real valued attributes. Since we deal with actual probability values P rather than probability densities p, all the real values are discretized to perform the analysis. But by discretizing the values we lose some information, such as the ordering of values. While estimating counts to determine the P values, we can borrow information from neighboring bins in case of discretized attributes.

Currently, we have a fixed number of levels for discretization. It is conceivable 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.

7. REFERENCES

[1] The third international knowledge discovery and data mining tools competition dataset kdd99-cup, 1999.

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

[3] 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.

[4] Rich Tsui Andrew Moore, Greg Cooper and Mike Wagner. Summary of biosurveillance-relevant

technologies.

[5] 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.

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

[7] 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.

[8] P. K. Chan, M. V. Mahoney, and M. H. Arshad. A machine learning approach to anomaly detection.

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

[10] 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.

[11] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo. A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data, 2002.

[12] 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.

[13] 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.

[14] 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.*

[15] P. Helman and J. Bhangoo. 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.

[16] 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.

[17] E. Keogh, S. Lonardi, and C. A. Ratanamahatana. Towards. parameter-free data mining. In *Proc. of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 206 – 215, 2004.

[18] 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, 2002.

[19] M. Kulldorff. A spatial scan statistic. pages 1481–1496, 1997.

[20] Si Quang Le and Tu Bao Ho. An association-based dissimilarity measure for categorical data. In *Pattern Recognition Letters archive*, volume 26, pages 2549 – 2557, 2005.

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

[22] 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.

[23] 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.

[24] 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.

[25] Andrew Moore and Weng-Keen Wong. Optimal reinsertion: A new search operator for accelerated and more accurate bayesian network structure learning. In *Proceedings of the 20th International Conference on Machine Learning (ICML '03)*, pages 552–559, August 2003.

[26] 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.

[27] Dan Pelleg. Scalable and practical probability density estimators for scientific anomaly detection. In *Doctoral Thesis, Carnegie Mellon University*, 2004.

[28] Denning PJ. Working sets past and present. In *IEEE Transactions on Software Engineering*, volume 6, 1980.

[29] 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.

[30] G. Cooper W-K Wong, A. W. Moore 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.

[31] 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.

[32] Weng-Keen Wong, Andrew Moore, Gregory Cooper, and Michael Wagner. Bayesian network anomaly pattern detection for disease outbreaks. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 808–815. AAAI Press, August 2003.

[33] 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.