

Conditional Anomaly Detection

Xiuyao Song, Mingxi Wu, Christopher Jermaine, Sanjay Ranka

Abstract—When anomaly detection software is used as a data analysis tool, finding the hardest-to-detect anomalies is not the most critical task. Rather, it is often more important to make sure that those anomalies that are reported to the user are in fact interesting. If too many unremarkable data points are returned to the user labeled as candidate anomalies, the software will soon fall into disuse.

One way to ensure that returned anomalies are useful is to make use of domain knowledge provided by the user. Often, the data in question include a set of environmental attributes whose values a user would never consider to be directly indicative of an anomaly. However, such attributes cannot be ignored because they have a direct effect on the expected distribution of the result attributes whose values *can* indicate an anomalous observation. This paper describes a general-purpose method called *conditional anomaly detection* for taking such differences among attributes into account, and proposes three different expectation-maximization algorithms for learning the model that is used in conditional anomaly detection. Experiments over 13 different data sets compare our algorithms with several other more standard methods for outlier or anomaly detection.

Index Terms—Data mining, Mining methods and algorithms.

I. INTRODUCTION

Anomaly detection has been an active area of computer science research for a very long time (see the recent survey by Markou and Singh [1]). Applications include medical informatics [2], computer vision [3][4], computer security [5], sensor networks [6], general-purpose data analysis and mining [7][8][9], and many other areas. However, in contrast to problems in supervised learning where studies of classification accuracy are the norm, little research has systematically addressed the issue of accuracy in general-purpose unsupervised anomaly detection methods. Papers have suggested many alternate problem definitions that are designed to boost the chances of finding anomalies (again, see Markou and Singh’s survey [1]), but there been few systematic attempts to maintain high coverage at the same time that false positives are kept to a minimum.

Accuracy in unsupervised anomaly detection is important because if used as a data mining or data analysis tool, an unsupervised anomaly detection methodology will be given a “budget” of a certain number of data points that it may call anomalies. In most realistic scenarios, a human being must investigate candidate anomalies reported by an automatic system, and usually has a limited capacity to do so. This

naturally limits the number of candidate anomalies that a detection methodology may usefully produce. Given that this number is likely to be small, it is important that most of those candidate anomalies are interesting to the end user.

This is especially true in applications where anomaly detection software is used to monitor incoming data in order to report anomalies in real time. When such events are detected, an alarm is sounded that requires immediate human investigation and response. Unlike the offline case where the cost and frustration associated with human involvement can usually be amortized over multiple alarms in each batch of data, each false alarm in the online case will likely result in an additional notification of a human expert, and the cost cannot be amortized.

A. Conditional Anomaly Detection

Taking into account such considerations is the goal of the research described in this paper. Rather than trying to find new and intriguing classes of anomalies, it is perhaps more important to ensure that those data points that a method does find *are in fact surprising*. To accomplish this, we ask the questions: What is the biggest source of inaccuracy for existing anomaly detection methods? Why might they return a large number of points that are not anomalies? To answer, we note that by definition, “statistical” methods for anomaly detection look for data points that refute a standard null hypothesis asserting that all data were produced by the same generative process. The null hypothesis is represented either explicitly (as in parametric methods [10][11][12]) or implicitly (as in various distance-based methods [9][7][8]). However, the questionable assumption made by virtually all existing methods is that there is no *a priori* knowledge indicating that all data attributes should not be treated in the same fashion.

This is an assumption that is likely to cause problems with false positives in many problem domains. In almost every application of anomaly detection, there are likely to be several data attributes that a human being would never consider to be directly indicative of an anomaly. By allowing an anomaly detection methodology to consider such attributes equally, accuracy may suffer.

For example, consider the application of online anomaly detection to syndromic surveillance, where the goal is to detect a disease outbreak at the earliest possible instant. Imagine that we monitor two variables: *max_daily_temp* and *num_fever*. *max_daily_temp* tells us the maximum outside temperature on a given day, and *num_fever* tells us how many people were admitted to a hospital emergency room complaining of a high fever. Clearly, *max_daily_temp* should never be taken as direct evidence of an anomaly. Whether it was hot or cold on a given day should never directly indicate whether or not we think we

The authors are with Computer and Information Sciences and Engineering Department, University of Florida, Gainesville, FL 32611. E-mail:(xsong, mwu, cjermain, ranka)@cise.ufl.edu

This material is based upon work supported by the National Science Foundation under Grant No. 0325459, IIS-0347408 and IIS-0612170. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

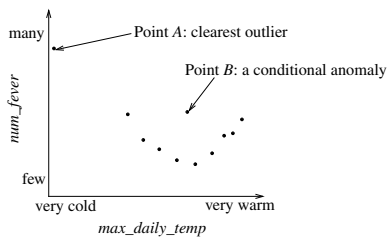


Fig. 1. Syndromic surveillance application.

have seen the start of an epidemic. For example, if the high in Gainesville, Florida on a given June day was only 70 degrees Fahrenheit (when the average high temperature is closer to 90 degrees), we simply have to accept that it was an abnormally cool day, but this does not indicate in any way that an outbreak has occurred.

While the temperature may not directly indicate an anomaly, it is not acceptable to simply ignore *max_daily_temp*, because *num_fever* (which clearly is of interest in detecting an outbreak) may be directly affected by *max_daily_temp*, or by a hidden variable whose value can easily be deduced by the value of the *max_daily_temp* attribute. In this example, we know that people are generally more susceptible to illness in the winter, when the weather is cooler. We call attributes such as *max_daily_temp* *environmental* attributes. The remainder of the date attributes (which the user *would* consider to be directly indicative of anomalous data) are called *indicator* attributes.

The anomaly detection methodology considered in this paper, called *conditional anomaly detection*, or *CAD*, takes into account the difference between the user-specified environmental and indicator attributes during the anomaly detection process, and how this affects the idea of an “anomaly”. For example, consider Figure 1. In this Figure, Point *A* and Point *B* are both anomalies or outliers based on most conventional definitions. However, if we make use of the additional information that *max_daily_temp* is not directly indicative of an anomaly, then it is likely safe for an anomaly detection system to ignore Point *A*. Why? If we accept that it is a cold day, then encountering a large number of fever cases makes sense, reducing the interest of this observation. For this reason, the CAD methodology will only label Point *B* an anomaly.

The methodology we propose works as follows. We assume that we are given a baseline data set and a user-defined partitioning of the data attributes into environmental attributes and indicator attributes, based upon which attributes the user decides should be directly indicative of an anomaly. Our method then analyzes the baseline data and learns which values are usual or typical for the indicator attributes. When a subsequent data point is observed, it is labeled anomalous or not depending on how much its indicator attribute values differ from the usual indicator attribute values. However, the environmental attributes are not necessarily ignored, because it may be that indicator attribute values are conditioned on environmental attribute values. If no such relationships are present in the baseline data, then the CAD methodology recognizes this, and will effectively ignore the environmental

attributes. In this case, or in the case when the user is unable or unwilling to partition the attributes (and so by default all attributes are indicator attributes) then the CAD methodology is equivalent to simply performing anomaly detection only on the indicator attributes. If such relationships *are* present, then the CAD methodology learns them and takes them into account.

B. Our Contributions

In addition to describing the parametric CAD methodology, the paper describes three EM-based [13] learning algorithms for the parametric CAD model. The paper describes a rigorous testing protocol and shows that if the true definition of an “anomaly” is a data point whose indicator attributes are not in-keeping with its environmental attributes, then the CAD methodology does indeed increase accuracy. By comparing the CAD methodology and its three learning algorithms against several conventional anomaly detection methods on thirteen different data sets, the paper shows that the CAD methodology is significantly better at recognizing data points where there is a mismatch among environmental and indicator attributes, and at ignoring unusual indicator attribute values that are still in-keeping with the observed environmental attributes. Furthermore, this effect is shown to be statistically significant, and not just an artifact of the number of data sets chosen for testing.

C. Paper Organization

The remainder of the paper is organized as follows. In Section 2, we describe the statistical model that we use to model environmental and indicator attributes, and the relationships between them. Section 2 also details how this model can be used to detect anomalies. Section 3 describes how to learn this model from an existing data set. Experimental results are given in Section 4, related work in Section 5, and the paper is concluded in Section 6. The Appendix gives a mathematical derivation of our learning algorithms.

II. STATISTICAL MODEL

This Section describes the statistical model we make use of for reducing the false positive rate during online anomaly detection. Like other parametric methods (such as those enumerated in Section 2.1 of Markou and Singh’s survey [1]), the detection algorithms described in this paper make use of a two-step process:

- 1) First, existing data are used to build a statistical model that captures the trends and relationships present in the data.
- 2) Then, during the online anomaly detection process, future data records are checked against the model to see if they match with what is known about how various data attributes behave individually, and how they interact with one another.

Like many other statistical methods, our algorithms rely on the basic principle of *maximum likelihood estimation* (MLE) [14]. In MLE, a (possibly complex) parametric distribution

f is first chosen to represent the data. f is then treated as a *generative model* for the data set in question. That is, we make the assumption that our data set of size n was in fact generated by drawing n independent samples from f . Given this assumption, we adjust the parameters governing the behavior of f so as to maximize the probability that f would have produced our data set.

Formally, to make use of MLE we begin by specifying a distribution $f(x|\theta_1, \theta_2, \dots, \theta_k)$, where the probability density function f gives the probability that a single experiment or sample from the distribution would produce the outcome x . The parameter $\Theta = \langle \theta_1, \theta_2, \dots, \theta_k \rangle$ governs the specific characteristics of the distribution (such as the mean and variance).

Since we are trying to model an entire data set, we do not have only a single experiment x ; rather, we view our data set $X = \langle x_1, x_2, \dots, x_n \rangle$ as the result of performing n experiments over f . Assuming that these experiments are independent, the likelihood that we would have observed $X = \langle x_1, x_2, \dots, x_n \rangle$ is given by:

$$L(X|\Theta) = \prod_{k=1}^n f(x_k|\Theta)$$

The MLE problem can then be succinctly stated as: given f and X , can we solve for Θ so as to maximize $L(X|\Theta)$ over all possible values of Θ ? Once we have solved for Θ , then we have a complete model for our data set.

A. The Gaussian Mixture Model

One of the common applications of MLE (particularly in machine learning) is to fit a multi-dimensional data set to a model described using the PDF f_{GMM} (f_{GMM} refers to the PDF for a *Gaussian Mixture Model*; a GMM is essentially a weighted mixture of multi-dimensional normal variables, and is widely used due to its ability to closely model even difficult, non-parametric distributions). A GMM can very easily be used as the basis for anomaly detection and is the most ubiquitous parametric model used for the purpose (see Section 2.1 of Markou and Singh [1]). The overall process begins by first performing the MLE required to fit the GMM to the existing data, using one of several applicable optimization techniques.

Of course, the problem described in the introduction of the paper is that if we allow all attributes to be treated in the same fashion (as all existing parametric methods seem to do), then false positives may become a serious problem. Our solution to this problem is to define a generative statistical model encapsulated by a PDF f_{CAD} that does not treat all attributes identically. This model is described in detail in the next subsection.

B. Conditional Anomaly Detection

f_{CAD} can be described intuitively as follows. We assume that each data point is an ordered pair of two sets of attribute values (x, y) , where x is the set of environmental attribute values, and y is the set of indicator attribute values. Our PDF will be of the form $f_{CAD}(y|\Theta, x)$. The PDF f_{CAD} is *conditioned* on x , which implies that our generative model

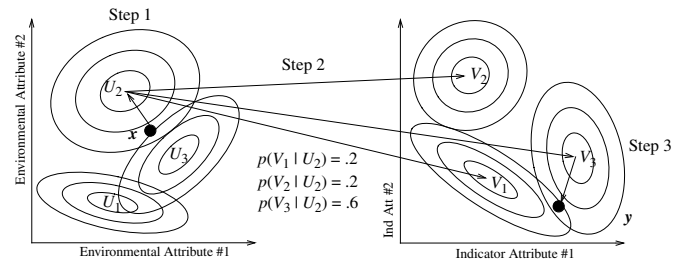


Fig. 2. The generative model used in conditional anomaly detection. Given x , a vector of environmental attribute values, we first determine which Gaussian from U generated x (step 1). Next, we perform a random trial using the mapping function to see which Gaussian from V we map to (step 2). In our example, we happen to choose V_3 , which has a probability 0.6 of being selected given that x is from U_2 . Finally, we perform a random trial over the selected Gaussian to generate y (step 3).

does not generate the environmental attributes associated with a data point. The PDF gives the likelihood that a single experiment with input x will give output y . Thus, a data point's environmental attributes x are taken as input, and used along with the model parameters Θ to generate the set of indicator attributes y . The net result is that when we perform a MLE to fit our model to the data, we will learn how the environmental attributes map to the indicator attributes. Formally, our model uses the following three sets of parameters, which together make up the parameter set Θ :

- A GMM U consisting of n_U Gaussians (or multidimensional normal “point clouds”), each of dimensionality d_U . The purpose of U is to model the distribution of the data set's environmental attributes. The i^{th} Gaussian in U is denoted by U_i and the weight of the i^{th} Gaussian is denoted by $p(U_i)$. The distribution parameters of the i^{th} Gaussian in U are given by $U_i = \langle \mu_{U_i}, \Sigma_{U_i} \rangle$.
- A set of n_V additional Gaussians of dimensionality d_V which we refer to collectively as V . Each Gaussian in V models a portion of the data space for the indicator attributes. The distribution parameters of the j^{th} Gaussian in V are given by $V_j = \langle \mu_{V_j}, \Sigma_{V_j} \rangle$.
- A probabilistic mapping function $p(V_j|U_i)$. Given a Gaussian $U_i \in U$, this function gives the probability that U_i maps to V_j . In other words, this function gives the probability that a tuple's indicator attributes are produced by the Gaussian V_j , if we know that the tuple's environmental attributes were produced by the Gaussian U_i .

Given these parameters, we assume the following generative process for each data point in the historical data set:

- 1) The process begins with the assumption that the values for a data point's environmental attributes were produced by a sample from U . Let U_i denote the Gaussian in U which was sampled. Note that the generative process does not actually produce x ; it is assumed that x was produced previously and the Gaussian which produced x is given as input.
- 2) Next, we use U_i to toss a set of “dice” to determine which Gaussian from V will be used to produce y . The probability of choosing V_j is given directly by the mapping function, and is $p(V_j|U_i)$.

- 3) Finally, y is produced by taking a single sample from the Gaussian that was randomly selected in Step (2). The process of generating a value for y given a value for x is depicted above in Figure 2.

It is important to note that while our generative model assumes that we know which Gaussian was used to generate x , in practice this information is not included in the data set. We can only infer this by computing a probability that each Gaussian in U was the one that produced x . As a result, $f_{CAD}(y|\Theta, x)$ is defined as follows:

$$f_{CAD}(y|\Theta, x) = \sum_{i=1}^{n_U} p(x \in U_i) \sum_{j=1}^{n_V} f_G(y|V_j) p(V_j|U_i)$$

Where:

- $p(x \in U_i)$ is the probability that x was produced by the i^{th} Gaussian in U , and is computed using Bayes' rule:

$$p(x \in U_i) = \frac{f_G(x|U_i)p(U_i)}{\sum_{k=1}^{n_U} f_G(x|U_k)p(U_k)}$$

- $f_G(y|V_j)$ is the likelihood that the j^{th} Gaussian in V would produce y , and is the standard Gaussian distribution (see the Appendix).
- $p(V_j|U_i)$ is the probability that the i^{th} Gaussian from U maps to the j^{th} Gaussian from V . This is given directly as a parameter in Θ .

Given the function $f_{CAD}(y|\Theta, x)$, the problem of modeling a data set (X, Y) can be succinctly stated as:

Choose Θ so as to maximize

$$\Lambda = \sum_{k=1}^n \log f_{CAD}(y_k|\Theta, x_k) \text{ over all possible values for } \Theta.$$

In this expression, Λ is referred to as the *log-likelihood* of the resulting model.

C. Detecting Conditional Anomalies

Like other parametric anomaly detection methods, the methodology for detecting future anomalies using the CAD model is based on the observation that once an optimal value for Θ has been learned, the function f_{CAD} can be used to give a meaningful ordering over all data points past and future, from the most astonishing to the most typical. When a new point is observed that has a small value for f_{CAD} , this means that it occurs in a portion of the data space with low density, and it should be flagged as an anomaly.

In order to determine the cutoff value for f_{CAD} below which a new point is determined to be anomalous, we allow the user to first pick a fraction ϵ from 0 to 1, and then we choose the f_{CAD} value such that exactly a fraction ϵ of the data points in the training set would have been flagged as anomalies. A high value of ϵ means that the method is less selective, and hence it increases the chance that an anomalous event will be flagged for further investigation. However, at the same time this will tend to increase the rate of false positives.

Given ϵ and the size of the training data set n , in order to choose a cutoff such that exactly $\epsilon\%$ of the training points would be considered anomalous, we simply sort the training

data based upon their f_{CAD} values, from lowest to highest. Next, let $c = \epsilon n$. This is the index of the cutoff point in the existing data set. Any future observation which is found to be more unusual than the c^{th} existing point will be termed an anomaly. To check if a point (x_{new}, y_{new}) is anomalous, we simply check if $f_{CAD}(y_{new}|\Theta, x_{new}) < f_{CAD}(y_c|\Theta, x_c)$. If yes, then the new observation is flagged as an anomaly.

III. LEARNING THE MODEL

Of course, defining a model like f_{CAD} is not enough. It is also necessary to define a process whereby it is possible to adjust the model parameters so that the model can be fitted to the existing data. In general, this type of maximization is intractable. This Section defines several learning algorithms for computing the model. The Section begins with a high-level introduction to the *Expectation Maximization* [13] framework that serves as a basis for all three of the algorithms.

A. The Expectation Maximization Methodology

While EM is usually called an ‘‘algorithm’’, in reality it is a generic technique for solving MLE problems. EM can either be easy or nearly impossible to apply to a specific MLE problem, but in general its application is non-trivial. We now turn to a description of the EM methodology.

As mentioned previously, most interesting MLE problems are computationally intractable. In practice, this intractability results from one or more ‘‘hidden’’ variables that cannot be observed in the data, and must be inferred simultaneously with the MLE. In the conditional anomaly detection problem, these hidden variables are the identities of the Gaussians that were used to produce the environmental and indicator attributes of each data point in the historical data set. The fact that these variables are not observable makes the problem difficult; if these values were included in the data, then solving for Θ becomes a mostly straightforward maximization problem relying on college-level calculus. The EM algorithm is useful in precisely such cases. EM is an iterative method, whose basic outline is as follows:

The Basic Expectation Maximization Algorithm

- 1) While the model continues to improve:
 - a) Let Θ be the current ‘‘best guess’’ as to the optimal configuration of the model.
 - b) Let $\bar{\Theta}$ be the next ‘‘best guess’’ as to the optimal configuration of the model.
 - c) **E-Step:** Compute Q , the expected value of Λ with respect to $\bar{\Theta}$ over all possible values of the hidden parameters. The probability of observing each possible set of hidden parameter values (required to compute the expectation of Λ) is computed using $\bar{\Theta}$.
 - d) **M-Step:** Choose $\bar{\Theta}$ so as to maximize the value for Q . $\bar{\Theta}$ then becomes the new ‘‘best guess’’.

EM sidesteps the problem of not knowing the values of the hidden parameters by instead considering the expected value of the PDF with respect to *all possible values* of the hidden parameters. Since computing the expected value of Λ

with respect to the hidden parameters requires that we be able to compute the probability of every possible configuration of the hidden parameters (and to do this we need to know the optimal configuration Θ), at each iteration EM computes this probability with respect to the best guess so far for Θ . Thus, EM does not compute a globally optimal solution at each iteration; rather, it computes a best guess as to the answer, which is guaranteed to improve at each iteration. EM has the desirable property that while it does not always converge to the globally optimal solution, it does always converge to a locally optimal solution, where no combination of slight “tweaks” of the various values in Θ can improve the value of Λ .

B. The Direct-CAD Algorithm

The remainder of this Section outlines three different EM algorithms for learning the CAD model over a data set. The first of the three algorithms attempts to learn all of the parameters simultaneously: the Gaussians that govern the generation of the environmental attributes, the Gaussians that govern the generation of the indicator attributes, and the mapping function between the two sets of Gaussians. This Section only outlines the approach, and gives the equations that are used to implement the algorithm; the complete mathematical derivation of the EM algorithm is quite involved and is left to the Appendix II of the paper. As in any EM algorithm, the Direct-CAD algorithm relies on the two classic steps: the *E-Step* and the *M-Step*.

E-Step. To begin the process of deriving an appropriate EM algorithm, we first need to define the hidden parameters in the context of the conditional anomaly detection problem. In conditional anomaly detection, the hidden parameters are the identities of the clusters from U and V that were used to produce each data point. To denote these parameters, we define:

- α_k tells which cluster in U produced the k^{th} value in X . $\alpha_k^{(i)}$ denotes the assertion “the k^{th} set of environmental attributes was produced by the i^{th} cluster in U .”
- β_k tells which cluster in V produced the k^{th} value in Y . $\beta_k^{(j)}$ denotes the assertion “the k^{th} set of indicator attributes was produced by the j^{th} cluster in V .”

Given these variables, we can then derive the Q function required by the E-Step of the EM algorithm. Let $Q(\Theta, \bar{\Theta})$ denote the expected value of Λ over all possible α, β given the current parameter values Θ . Then, as derived in the Appendix II:

$$\begin{aligned} Q(\Theta, \bar{\Theta}) &= E\left[\sum_{k=1}^n \log f_{CAD}(x_k, y_k, \alpha_k, \beta_k | x_k, \bar{\Theta}) | X, Y, \Theta\right] \\ &= \sum_{k=1}^n \sum_{i=1}^{n_U} \sum_{j=1}^{n_V} \left\{ \left[\begin{aligned} &\log f_G(y_k | V_j, \bar{\Theta}) + \log p(V_j | U_i, \bar{\Theta}) \\ &+ \log f_G(x_k | U_i, \bar{\Theta}) + \log p(U_i) \\ &- \log f(x_k | \bar{\Theta}) \end{aligned} \right] \times b_{kij} \right\} \end{aligned}$$

Where:

$$b_{kij} = \frac{f_G(x_k | U_i) p(U_i) f_G(y_k | V_j) p(V_j | U_i, \Theta)}{\sum_{t=1}^{n_U} \sum_{h=1}^{n_V} \{f_G(x_k | U_t) p(U_t) f_G(y_k | V_h) p(V_h | U_t, \Theta)\}}$$

M-Step. In the M-Step, we now use standard calculus to maximize the value of Q over all $\bar{\Theta}$. This results in a series of equations that can be used to compute the new parameter values that will be used during the next iteration. The details of the derivation of each of the equations are reasonably involved and are given in the Appendix. However, the use of the equations is relatively straightforward, and requires only that we perform standard arithmetic and linear algebra operations over the previous model parameters and the data set. The formulas for the components of $\bar{\Theta}$ are as follows:

$$\begin{aligned} \bullet \overline{p(U_i)} &= \sum_{k=1}^n \sum_{j=1}^{n_V} b_{kij} / \sum_{k=1}^n \sum_{h=1}^{n_V} \sum_{j=1}^{n_V} b_{kij} \\ \bullet \bar{\mu}_{U_i} &= \sum_{k=1}^n \sum_{j=1}^{n_V} b_{kij} x_k / \sum_{k=1}^n \sum_{j=1}^{n_V} b_{kij} \\ \bullet \bar{\Sigma}_{U_i} &= \sum_{k=1}^n \sum_{j=1}^{n_V} b_{kij} (x_k - \bar{\mu}_{U_i})(x_k - \bar{\mu}_{U_i})^T / \sum_{k=1}^n \sum_{j=1}^{n_V} b_{kij} \\ \bullet \bar{\mu}_{V_j} &= \sum_{k=1}^n \sum_{i=1}^{n_U} b_{kij} y_k / \sum_{k=1}^n \sum_{i=1}^{n_U} b_{kij} \\ \bullet \bar{\Sigma}_{V_j} &= \sum_{k=1}^n \sum_{i=1}^{n_U} b_{kij} (y_k - \bar{\mu}_{V_j})(y_k - \bar{\mu}_{V_j})^T / \sum_{k=1}^n \sum_{i=1}^{n_U} b_{kij} \\ \bullet \overline{p(V_j | U_i)} &= \sum_{k=1}^n b_{kij} / \sum_{k=1}^n \sum_{h=1}^{n_V} b_{kih} \end{aligned}$$

Given the update rules described above, our EM algorithm for learning the conditional anomaly detection model is then as follows:

The Direct-CAD Algorithm

- 1) Choose an initial set of values for $\langle \mu_{V_j}, \Sigma_{V_j} \rangle$, $\langle \mu_{U_i}, \Sigma_{U_i} \rangle$, $p(V_j | U_i)$, $p(U_i)$, for all i, j .
- 2) While the model continues to improve (measured by the improvement of Λ):
 - a) Compute b_{kij} for all k, i , and j as described under E-Step above;
 - b) Compute $\langle \bar{\mu}_{V_j}, \bar{\Sigma}_{V_j} \rangle$, $\langle \bar{\mu}_{U_i}, \bar{\Sigma}_{U_i} \rangle$, $\overline{p(V_j | U_i)}$, $\overline{p(U_i)}$ for all i, j as described under **M-Step** above;
 - c) Set $\langle \mu_{V_j}, \Sigma_{V_j} \rangle := \langle \bar{\mu}_{V_j}, \bar{\Sigma}_{V_j} \rangle$, $\langle \mu_{U_i}, \Sigma_{U_i} \rangle := \langle \bar{\mu}_{U_i}, \bar{\Sigma}_{U_i} \rangle$, $p(V_j | U_i) := \overline{p(V_j | U_i)}$, $p(U_i) := \overline{p(U_i)}$ for all i, j .

The time and space requirements of this algorithm can be derived as follows. To compute b_{kij} in the E-Step, we first compute $f_G(x_k | U_i)$ and $f_G(y_k | V_j)$, which takes $O(nn_U d_U^2)$ time and $O(nn_V d_V^2)$ time, respectively. Then it takes $O(nn_U n_V)$ time to compute all the b_{kij} values. In the M-Step, we update the value for each component of $\bar{\Theta}$. It takes $O(nn_U n_V d_U^2)$ time to update $\bar{\Sigma}_{U_i}$, and $O(nn_U n_V d_V^2)$ time to update $\bar{\Sigma}_{V_j}$. To update the rest of the components, the time complexity is $O(nn_U n_V (d_U + d_V + n_V))$. Adding them together, the time complexity for one iteration (E-Step and M-Step) is $O(nn_U n_V (d_U^2 + d_V^2))$. The total time complexity of the conditional anomaly detection algorithm depends on

the convergence speed, i.e., the number of iterations, which depends on the data set itself.

Accordingly, in the E-Step, storing the results for $f_G(x_k|U_i)$, $f_G(y_k|V_j)$ and b_{kij} requires $O(nn_U n_V)$ space. Storing the updated components of $\bar{\Theta}$ requires $O(n_U d_U^2 + n_V d_V^2)$ space.

C. The GMM-CAD-Full Algorithm

Unfortunately, the algorithm described in the previous Subsection has a major potential drawback. The EM approach is known to be somewhat sensitive to converging to locally optimal solutions that are far from the globally optimal solution, particularly when it is used in conjunction with very complex optimization problems. While standard methods such as multiple runs with randomized seeds may help [15], the problem remains that the CAD problem is certainly complex as MLE problems go, since the task is to simultaneously learn two sets of Gaussians as well as a mapping function, and the problem is complicated even more by the fact that the function f_{CAD} is only conditionally dependent upon the set of Gaussians corresponding to the distribution of the environmental attributes.

As a result, the second learning algorithm that we consider in this paper makes use of a two-step process. We begin by assuming that the number of Gaussians in U and V is identical; that is, $n_U = n_V$. Given this, a set of Gaussians is learned in the *combined* environmental and indicator attribute space. These Gaussians are then projected onto the environmental attributes and indicator attributes to obtain U and V . Then, only after U and V have been fixed, a simplified version of the Direct-CAD algorithm is used to compute only the values $p(V_i|U_j)$ that make up the mapping function. The benefit of this approach is that by breaking the problem into two, much easier optimization tasks, each solved by much simpler EM algorithms, we may be less vulnerable to the problem of local optima. The drawback of this approach is that we are no longer trying to maximize f_{CAD} directly; we have made the simplifying assumption that it suffices to *first* learn the Gaussians, and *then* learn the mapping function.

The GMM-CAD-Full Algorithm

- 1) Learn a set of $n_U (d_U + d_V)$ -dimensional Gaussians over the data set $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$. Call this set Z .
- 2) Let μ_{Z_i} refer to the centroid of the i^{th} Gaussian in Z , and let Σ_{Z_i} refer to the covariance matrix of the i^{th} Gaussian in Z . We then determine U as follows:
 - For $i = 1$ to n_U do:
 - $p(U_i) := p(Z_i)$
 - For $j = 1$ to d_U do:
 - $\mu_{U_i}[j] := \mu_{Z_i}[j]$
 - For $k = 1$ to d_U do:
 - $\Sigma_{U_i}[j][k] := \Sigma_{Z_i}[j][k]$
 - 3) Next, determine V as follows:
 - For $i = 1$ to n_V do:
 - For $j = 1$ to d_V do:
 - $\mu_{V_i}[j] := \mu_{Z_i}[j + d_U]$
 - For $k = 1$ to d_V do:
 - $\Sigma_{V_i}[j][k] := \Sigma_{Z_i}[j + d_U][k + d_U]$

- 4) Run a second EM algorithm to learn the mapping function. While the model continues to improve (measured by the improvement of Λ):
 - a) Compute b_{kij} for all k, i , and j as in the previous Subsection.
 - b) Compute $\overline{p(V_j|U_i)}$ as described in the previous Subsection.
 - c) Set $p(V_j|U_i) := \overline{p(V_j|U_i)}$.

The time required for the GMM-CAD-Full algorithm consists of two parts: the time to learn the Gaussians, which is $O(nn_U (d_U + d_V)^2)$, and the time to learn the mapping function by simplified Direct-CAD algorithm, which is $O(nn_U^3)$. The memory required for GMM-CAD-Full algorithm is $O(nn_U + n_U (d_U + d_V)^2)$.

D. The GMM-CAD-Split Algorithm

There is a third obvious way to learn the required model, that is closely related to the previous one. Since the previous algorithm breaks up the optimization problem into two smaller problems, there is no longer any reason to think that the GMMs for U and V should be learned simultaneously. Learning them simultaneously might actually be overly restrictive, since learning them simultaneously will try to learn U and V so that every Gaussian in U is “mated” to a Gaussian in V through a covariance structure. The CAD model has no such requirement, since the Gaussians of U and V are related only through the mapping function. In fact, the algorithm from the previous Subsection throws out all of this additional covariance information since $\Sigma_{Z_i}[j][k]$ is never used for $(j \leq d_U, k > d_U)$ or $(j > d_U, k \leq d_U)$.

As a result, the third algorithm we consider learns U and V directly as two separate GMMs, and then learns the mapping function between them. The algorithm is outlined below.

The GMM-CAD-Split Algorithm

- 1) Learn U and V by performing two separate EM optimizations.
- 2) Run Step (4) from the GMM-CAD-Full Algorithm to learn the mapping function.

In GMM-CAD-Split algorithm, learning Gaussians in U and V needs $O(nn_U (d_U^2 + d_V^2))$ time. Learning the mapping function needs $O(nn_U^3)$ time. The memory requirement for the algorithm is $O(nn_U + n_U (d_U^2 + d_V^2))$.

E. Choosing the Complexity of the Model

Whatever learning algorithm is used, it is necessary to choose as an input parameter the number of clusters or Gaussians in the model. In general, it is difficult to choose the optimal value of the number of Gaussians in a GMM, and this is a research problem on its own [16]. However, our problem is a bit easier because we are not actually trying to cluster the data for the sake of producing a clustering; rather, we are simply trying to build an accurate model for the data. In our experiments, we generally found that a larger number of clusters in the CAD model results in more accuracy. It is true that a very large number of clusters could cause over-fitting, but in the CAD model, the computational cost associated with

any model large enough to cause over-fitting makes it virtually impossible to over-fit a data set of reasonable size. Thus, our recommendation is to choose the largest model that can be comfortably computed. In our implementation, we use 40 clusters for both U and V .

IV. BENCHMARKING

In this Section, we describe a set of experiments aimed at testing the effectiveness of the CAD model. Our experiments are aimed at answering the following three questions:

- 1) Which of the three learning algorithms should be used to fit the CAD model to a data set?
- 2) Can use of the CAD model reduce the incidence of false positives due to unusual values for environmental attributes compared to obvious alternatives for use in an anomaly detection system?
- 3) If so, does the reduced incidence of false positives come at the expense of a lower detection level for actual anomalies?

A. Experimental Setup

Unfortunately, as with any unsupervised learning task, it can be very difficult to answer questions regarding the quality of the resulting models. Though the quality of the models is hard to measure, we assert that with a careful experimental design, it is possible to give a strong indication of the answer to our questions. Our experiments are based on the following key assumptions:

- Most of the points in a given data set are not anomalous, and so a random sample of a data set's points should contain relatively few anomalies.
- If we take a random sample of a data set's points and perturb them by swapping various attribute values among them, then the resulting set of data points should contain a much higher fraction of anomalies than a simple random sample from the data set.

Given these assumptions, our experimental design is as follows. For each data set, the following protocol was repeated ten times:

- 1) For a given data set, we begin by randomly designating 80% of the data points as *training data*, and 20% of the data points as *test data* (this latter set will be referred to as *testData*).
- 2) Next, 20% anomalies or outliers in the set *testData* are identified using a standard, parametric test based on Gaussian mixture modeling – the data are modeled using a GMM and new data points are ranked as potential outliers based on the probability density at the point in question. This basic method is embodied in more than a dozen papers cited in Markou and Singh's survey [1]. However, a key aspect of this protocol is that *outliers are chosen based only on the values of their environmental attributes* (indicator attributes are ignored). We call this set *outliers*.
- 3) The set *testData* is then itself randomly partitioned into two identically-sized subsets: *perturbed* and

nonPerturbed, subject to the constraint that *outliers* \subset *nonPerturbed*.

- 4) Next, the members of the set *perturbed* are perturbed by swapping indicator attribute values among them.
- 5) Finally, we use the anomaly detection software to check for anomalies among $testData = perturbed \cup nonPerturbed$.

Note that after this protocol has been executed, members of the set *perturbed* are no longer samples from the original data distribution, and members of the set *nonPerturbed* are still samples from the original data distribution. Thus, it should be relatively straightforward to use the resulting data sets to differentiate between a robust anomaly detection framework, and one that is susceptible to high rates of false positives. Specifically, given this experimental setup, we expect that a useful anomaly detection mechanism would:

- Indicate that a relatively large fraction of the set *perturbed* are anomalies; and
- Indicate that a much smaller fraction of the set *nonPerturbed* are anomalies; and
- Indicate that a similarly small fraction of the set *outliers* are anomalies.

This last point bears some further discussion. Since these records are also samples from the original data distribution, the only difference between *outliers* and (*nonPerturbed* – *outliers*) is that the records in *outliers* have exceptional values for their environmental attributes. Given the definition of an environmental attribute, these should not be considered anomalous by a useful anomaly detection methodology. In fact, the ideal test result would show that the percentage of anomalies among *outliers* is *identical* to the percentage of anomalies among *nonPerturbed*. A high percentage of anomalies among the set *outliers* would be indicative of a tendency towards false alarms in the case of anomalous (but uninteresting) values for environmental attributes.

B. Creating the Set Perturbed

The above testing protocol requires that we be able to create a set of points *perturbed* with an expectedly high percentage of anomalies. An easy option would have been to generate perturbed data by adding noise. However, this can produce meaningless (or less meaningful) values of attributes. We wanted to ensure that the perturbed data is structurally similar to the original distribution and domain values so that it is not trivial to recognize members of this set because they have similar characteristics compared to the training data (that is, the environmental attributes are realistic and the indicator attributes are realistic, but the relationship between them may not be). We achieve this through a natural perturbation scheme:

Let D be the set of data records that has been chosen from *testData* for perturbation. Consider a record $z = (x, y)$ from D that we wish to perturb. Recall from Section 2.2 that x is a vector of environmental attribute values and y is a vector of indicator attribute values. To perturb z , we randomly choose k data points from D ; in our tests, we use $k = \min(50, |D|/4)$. Let $z' = (x', y')$ be the sampled record such that the Euclidean distance between y and y' is maximized over all k of our sampled data points. To perturb z , we simply create a new data point (x, y') , and add this point to *perturbed*.

This process is iteratively applied to all points in the set D to *perturbed*. Note that every set of environmental attribute values present in D is also present in *perturbed* after perturbation. Furthermore, every set of indicator attributes present in *perturbed* after perturbation is also present in D . What has been changed is only the *relationship* between the environmental and indicator attributes in the records of *perturbed*.

We note that swapping the indicator values may not always produce the desired anomalies (it is possible that some of the swaps may result in perfectly typical data). However, swapping the values *should* result in a new data set that has a larger fraction of anomalies as compared to sampling the data set from its original distribution, simply because the new data has *not* been sampled from the original distribution. Since some of the perturbed points may not be abnormal, even a perfect method should deem a small fraction of the *perturbed* set to be normal. Effectively, this will create a non-zero false negative rate, even for a “perfect” classifier. Despite this, the experiment is still a valid one because each of the methods that we test must face this same handicap.

C. Data Sets Tested

We performed the tests outlined above over the following thirteen data sets¹. For each data set, the annotation $(i \times j)$ indicates that the data set had i environmental and j indicator attributes, which were chosen after a careful evaluation of the data semantics. A brief description of the environmental and indicator attributes for each data set is also given. The thirteen data sets used in testing are:

- 1) *Synthetic* data set (50×50). Synthetic data set created using the CAD model. 10,000 data points were sampled from the generative model f . f has 10 Gaussians in U and V respectively. The centroids of the Gaussians are random vectors whose entries are chosen on the interval $(0.0, 1.0)$. The diagonal of the covariance matrix is set to $1/4$ of the average distance between the centroids in each dimension. The weight of the Gaussians is evenly assigned and the mapping function $P(V_j|U_i)$ for a certain value of i is a permutation of geometric distribution.
- 2) *Algae* data set (11×6). The data is from UCI KDD Archive. We removed the data records with missing values. The environmental attributes consist of the season, the river size, the fluid velocity, and some chemical concentrations. The indicator attributes are the distributions of different kinds of algae in surface water.
- 3) *Streamflow* data set (205×100). The data set depicts the river flow levels (indicator) and precipitation and temperature (environmental) for California. We create this data set by concatenating the environmental and indicator data sets. The environmental data set is obtained from National Climate Data Center (NCDC). The indicator data set is obtained from U.S. Geological Survey (USGS).
- 4) *ElNino* data set (4×5). The data is from UCI KDD Archive. It contains oceanographic and surface meteorological readings taken from a series of buoys positioned throughout the equatorial Pacific. The environmental attributes are the spatio-temporal information. The indicator attributes are the wind direction, relative humidity and temperature at various locations in the Pacific Ocean.
- 5) *Physics* data set (669×70). The data is from KDD Cup 2003. For a large set of physics papers, we pick out frequency information for key words (environmental) and a list of most referred articles (indicator). One data record, which represents one physics paper, has 0/1-valued attributes, corresponding to the appearance/absence of the key word (or referred-to article) respectively.

¹we have conducted an experiment on a pre-labeled data set as suggested. Refer to Appendix I for the results

- 6) *Bodyfat* data set (13×2). The data is from CMU statlib. It depicts body fat percentages (indicator) and physical characteristics (environmental) for a group of people.
- 7) *Houses* data set (8×1). The data is from CMU statlib. It depicts house price (indicator) in California and related environmental characteristics (environmental) such as median income, housing median age, total rooms, etc.
- 8) *Boston* data set (15×1). The data is from CMU statlib. We removed some attributes that are not related to environmental-indicator relationship. The data depicts the house value of owner-occupied homes (indicator) and economic data (environmental) for Boston.
- 9) *FCAT-math* data set (14×12). The data is from National Center for Education Statistics (NCES) and Florida Information Resource Network (FIRN). We removed the data records with missing values. It depicts Mathematics achievement test results (indicator) of grade 3 of year 2002 and regular elementary schools’ characteristics (environmental) for Florida.
- 10) *FCAT-reading* data set (14×11). The data is also from NCES and FIRN. We processed the data similarly as with the FCAT-math data set. It depicts the reading achievement test scores (indicator) and regular elementary schools’ characteristics (environmental) for Florida.
- 11) *FLFarms* data set (114×52). The data set depicts the Florida state farms’ market value in various aspects (indicator) and the farms’ operational and products statistics (environmental).
- 12) *CAFarms* data set (115×51). The data set depicts the California state farms’ market value in various aspects (indicator) and the farms’ operational and products statistics (environmental).
- 13) *CAPeaks* data set (2×1). The data set depicts the California peaks’ height (indicator) and longitude and altitude position (environmental).

D. Experimental Results

We performed the tests described above over six alternatives for anomaly detection:

- 1) Simple Gaussian mixture modeling (as described at the beginning of this Section). In this method, for each data set a GMM model is learned directly over the training data. Next, the set *testData* is processed, and anomalies are determined based on the value of the function f_{GMM} for each record in *testData*; those with the smallest f_{GMM} values are considered anomalous.
- 2) k^{th} -NN (k th nearest neighbor) outlier detection [9] (with $k = 5$). k^{th} -NN outlier detection is one of the most well-known methods for outlier detection. In order to make it applicable to the training/testing framework we consider, the method must be modified slightly, since each test data point must be scored in order to describe the extent to which it deviates from the training set. Given a training data set, in order to use the 5^{th} method in order to determine the degree to which a test point is anomalous, we simply use the distance from the point to its 5^{th} in the training set. A larger distance indicates a more anomalous point.
- 3) Local outlier factor (LOF) anomaly detection [8]. LOF must be modified in a matter similar to k^{th} -NN outlier detection: the LOF of each test point is computed with respect to the training data set in order to score the point.
- 4) Conditional anomaly detection, with the model constructed using the Direct-CAD Algorithm.
- 5) Conditional anomaly detection, with the model constructed using the GMM-CAD-Full Algorithm.
- 6) Conditional anomaly detection, with the model constructed using the GMM-CAD-Split Algorithm.

For each experiment over each data set with the GMM/CAD methods, a data record was considered an anomaly if the

probability density at the data point was less than the median probability density at all of the test data points. For 5th-NN/LOF, the record was considered an anomaly if its distance was greater than the median distance over all test data points. The median was used because exactly one-half of the points in the set *testData* were perturbed. Thus, if the method was able to order the data points so that all anomalies were before all non-anomalies, choosing the median density should result in a recall² rate of 100% and a false positive rate of 0% (a false positive rate of 0% is equivalent to a precision³ of 100%). Also note that since exactly one-half of the points are perturbed and each method is asked to guess which half are perturbed, it is always the case that the recall is exactly equal to the precision when identifying either anomalous or non-anomalous records in this experiment.

The results are summarized in the two tables below. The first table summarizes and compares the quality of the recall/precision obtained when identifying perturbed points in *testData*. Each cell indicates which of the two methods compared in the cell was found to be the “winner” in terms of having superior recall/precision over the 13 sets. To compute the “winner”, we do the following. For each of the 13 sets and each pair of detection methods, the recall/precision of the two methods is averaged over the 10 different training/testing partitionings of the data set. Let *method*₁ and *method*₂ be the two anomaly detection methods considered in a cell. *method*₁ is said to “win” the test if it has a higher average recall/precision in 7 or more of the 13 data sets. Likewise, *method*₂ is declared the winner if it has a higher recall/precision in 7 or more of the data sets. In addition, we also give the number of the 13 data sets for which the winning method performed better, and we give a *p*-value that indicates the significance of the results.

Finally, the second column in the table gives the average recall/precision over all experiments for each method. This is the average percentage of perturbed points that have probability density less than the median (for GMM and CAD) or have a score/distance greater than the median (LOF and 5th-NN) Though these numbers are informative, we caution that the number of head-to-head wins is probably a better way to compare two methods. The reason for this is that the recall/precision can vary significantly from data set to data set, and thus the average may tend to over-weight those data sets that are particularly difficult.

This *p*-value addresses the following question. Imagine that the whole experiment was repeated by choosing a new, arbitrary set of 13 datasets at random, and performing 10 different training/testing partitionings of each data set. The *p*-value gives the probability that we would obtain a different winner than was observed in our experiments due to choosing different data sets and partitionings. A low *p*-value means that it is unlikely that the observed winner was due mostly to chance, and so the results are significant. A high *p*-value means that if the experiment was repeated again with a new set of data sets, the result may be different. The *p*-value was

²Recall in this context is the fraction of perturbed points that are flagged as anomalies.

³Precision in this context is the fraction of anomalies that are actually perturbed points.

computed using a bootstrap re-sampling procedure [17].

The second table is similar to the first, but rather than comparing the recall/precision with which the methods can identify the perturbed points, it summarizes the recall/precision in identifying as non-anomalous points in the set *outliers*. Recall that the *outliers* set are points with exceptional values for their environmental attributes, but which are non-perturbed and so they are actually obtained from the training distribution. Thus, this table gives an indication of how successful the various methods are in ignoring anomalous environmental attributes when the points were in fact sampled from the baseline (or training) distribution. Just as in the first table, each cell gives the winner, the number of trials won, and the associated *p*-value of the experiment. The second column gives the average precision/recall. This is the percentage of non-perturbed points with exceptional environmental attributes that were considered to be non-anomalous.

E. Discussion

Our experiments were designed to answer the following question: If a human expert wishes to identify a set of environmental attributes that should not be directly indicative of an anomaly, can any of the methods successfully ignore exceptional values for those attributes while at the same time taking them into account when performing anomaly detection?

The first part of the question is addressed by the results given in Table II. This Table shows that if the goal is to ignore anomalous values among the environmental attributes, then the CAD-GMM-Full algorithm is likely the best choice. This method declared the lowest percentage of data points with exceptional environmental attribute values to be anomalous. In every head-to-head comparison, it did a better job for at least 77% of the data sets (or 10 out of 13).

The second part of the question is addressed by the results given in Table I. This table shows that if the goal is to not only ignore exceptional environmental attributes, but also to take the normal correlation between environmental attributes and the indicator attributes into account, then the CAD-GMM-Full algorithm is again the best choice. Table I shows that compared to all of the other options, this method was best able to spot cases where the indicator attributes were not in-keeping with the environmental attributes because they had been swapped.

Finally, it is also important to point out that no method was uniformly preferable to any other method for each and every data set tested. It is probably unrealistic to expect that any method *would* be uniformly preferable. For example, we found that the CAD-based methods generally did better compared to the two distance-based methods (and the GMM method) on higher-dimensional data. The results were most striking for the thousand-dimensional physics paper data set, where the precision/recall for LOF and 5th-NN when identifying perturbed data was worse than a randomized labeling would have done (less than 50%). In this case, each of the CAD methods had greater than 90% precision/recall. For lower-dimensional data, the results were more mixed. Out of all of the head-to-head comparisons reported in Tables I and II, there

TABLE I

HEAD-TO-HEAD WINNER WHEN COMPARING RECALL/PRECISION FOR IDENTIFYING PERTURBED DATA AS ANOMALOUS (EACH CELL HAS WINNER, WINS OUT OF 13 DATA SETS, BOOTSTRAPPED p -VALUE)

	Avg rcl/prc	CAD-Full	CAD-Split	Direct CAD	5 th -NN	LOF
GMM	.720	Full, 11, 0.01	Split, 8, 0.25	CAD, 8, 0.35	GMM, 8, 0.25	GMM, 8, 0.16
CAD-Full	.793	–	Full, 10, 0.04	Full, 10, 0.03	Full, 10, 0.01	Full, 10, 0.01
CAD-Split	.737	–	–	Split, 7, 0.47	Split, 7, 0.41	Split, 8, 0.23
Direct CAD	.730	–	–	–	CAD, 8, 0.28	CAD, 9, 0.13
5 th -NN	.721	–	–	–	–	5 th -NN, 7, 0.41
LOF	.721	–	–	–	–	–

TABLE II

HEAD-TO-HEAD WINNER WHEN COMPARING RECALL/PRECISION FOR IDENTIFYING NON-PERTURBED OUTLIER DATA AS NON-ANOMALOUS (EACH CELL HAS WINNER, WINS OUT OF 13 DATA SETS, BOOTSTRAPPED p -VALUE)

	Avg rcl/prc	CAD-Full	CAD-Split	Direct CAD	5 th -NN	LOF
GMM	.500	Full, 12, 0.00	Split, 10, 0.06	CAD, 9, 0.12	GMM, 9, 0.23	GMM, 7, 0.34
CAD-Full	.749	–	Full, 11, 0.01	Full, 10, 0.01	Full, 13, 0.00	Full, 10, 0.01
CAD-Split	.687	–	–	CAD, 7, 0.49	Split, 10, 0.01	Split, 9, 0.09
Direct CAD	.681	–	–	–	CAD, 11, 0.00	CAD, 9, 0.12
5 th -NN	.500	–	–	–	–	LOF, 7, 0.37
LOF	.549	–	–	–	–	–

is only one case where any method won 13 out of 13 times (CAD-GMM-Full vs. LOF in terms of the ability to ignore anomalous environmental attributes). Thus, our results do not imply that any method will do better than any other method on an arbitrary data set. What the results *do* imply is that if the goal is to choose the method that is *most likely* to do better on an arbitrary data set, then the CAD-GMM-Full algorithm is the best choice.

V. RELATED WORK

Anomaly and outlier detection have been widely studied, and the breadth and depth of the existing research in the area precludes a detailed summary here. An excellent survey of so-called “statistical” approaches to anomaly detection can be found in Markou and Singh’s 2003 survey [1]. In this context, “statistical” refers to methods which try to identify the underlying generative distribution for the data, and then identify points that do not seem to belong to that distribution. Markou and Singh give references to around a dozen papers, that, like the CAD methodology, rely in some capacity on Gaussian Mixture Modeling to represent the underlying data distribution. However, unlike the CAD methodology, nearly all of those papers implicitly assume that all attributes should be treated equally during the detection process.

The specific application area of intrusion detection is the topic of literally hundreds of papers that are related to anomaly detection. Unfortunately, space precludes a detailed survey of methods for intrusion detection here. One key difference between much of this work and our own proposal is the specificity: most methods for intrusion detection are specifically tailored to that problem, and are not targeted towards the generic problem of anomaly detection, as the CAD methodology is. An interesting 2003 study by Lazarevic et al. [18] considers the application of generic anomaly detection methods to the specific problem of intrusion detection. The

authors test several methods and conclude that LOF [8] (tested in this paper) does a particularly good job in that domain.

While most general-purpose methods assume that all attributes are equally important, there are some exceptions to this. Aggarwal and Yu [19] describe a method for identifying outliers in subspaces of the entire data space. Their method identifies combinations of attributes where outliers are particularly obvious, the idea being that a large number of dimensions can obscure outliers by introducing noise (the so-called “curse of dimensionality” [20]). By considering subspaces, this effect can be mitigated. However, Aggarwal and Yu make no *a priori* distinctions among the types of attribute values using domain knowledge, as is done in the CAD methodology.

One body of work on anomaly detection originating in statistics that does make an *a priori* distinction among different attribute types in much the same way as the CAD methodology is so-called *spatial scan statistics* [21][22][23]. In this work, data are aggregated based upon their spatial location, and then the spatial distribution is scanned for sparse or dense areas. Spatial attributes are treated in a fashion analogous to environmental attributes in the CAD methodology, in the sense that they are not taken as direct evidence of an anomaly; rather, the expected sparsity or density is conditioned on the spatial attributes. However, such statistics are limited in that they cannot easily be extended to multiple indicator attributes (the indicator attribute is assumed to be a single count), nor are they meant to handle the sparsity and computational complexity that accompany large numbers of environmental attributes.

The existing work that is probably closest in spirit to our own is likely the paper on anomaly detection for finding disease outbreaks by Wong, Moore, Cooper, and Wagner [24]. Their work makes use of a Bayes Net as a baseline to describe possible environmental structures for a large number of input variables. However, because their method relies on a Bayes Net, there are certain limitations inherent to this approach. The most significant limitation of relying on a

Bayes Net is that continuous phenomena are hard to describe in this framework. Among other issues, this makes it very difficult to describe the temporal patterns such as time lags and periodicity. Furthermore, a Bayes Net implicitly assumes that even complex phenomena can be described in terms of interaction between a few environmental attributes, and that there are no hidden variables unseen in the data, which is not true in real situations where many hidden attributes and missing values are commonplace. A Bayes Net cannot learn two different explanations for observed phenomena and learn to recognize both, for it cannot recognize unobserved variables. Such hidden variables are precisely the type of information captured by the use of multiple Gaussian clusters in our model; each Gaussian corresponds to a different hidden state.

Finally, we mention that traditional prediction models, such as linear or nonlinear models of regression [25][26] can also be potentially used for deriving the cause-effect interrelationships that the CAD method uses. For example, for a given set of environmental attribute values, a trained regression model will predict the indicator attribute value, and by comparing it with the actual indicator attribute value, we can determine if the test point is anomalous or not. However, there are important limitations on such regression methods as compared to the CAD model. First, in regression there is typically one response variable (indicator variable) and multiple predictor variables (environmental variables), while the CAD method allows an arbitrary number of indicator attributes. Second, it is unclear how a regression-based methodology could be used to rank the interest of data points based on their indicator attributes. Distance from the predicted value could be used, but this seems somewhat arbitrary, since the natural variation may change depending on indicator attribute values.

VI. CONCLUSIONS AND FUTURE WORK

In this paper we have described a new approach to anomaly detection. We take a rigorous, statistical approach where the various data attributes are partitioned into *environmental* and *indicator* attributes, and a new observation is considered anomalous if its indicator attributes cannot be explained in the context of the environmental attributes. There are many avenues for future work. For example, we have considered only one way to take into account a very simple type of domain knowledge to boost the accuracy of anomaly detection. How might more complex domain knowledge boost accuracy even more? For another example, an important issue that we have not addressed is scalability during the construction of the model. While the model can always quickly and efficiently be used to check for new anomalies, training the model on a multi-gigabyte database may be slow without a modification of our algorithms. One way to address this may be to use a variation of the techniques of Bradley et al. [27] for scaling standard, Gaussian EM to clustering of very large databases.

APPENDIX I

EXPERIMENT ON KDD CUP 1999 DATASET

One of the anonymous referees for the paper suggested that we apply our algorithms to the KDD Cup 1999 data set.

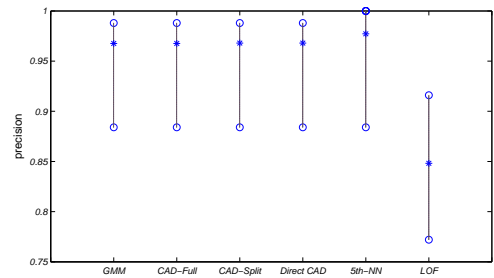


Fig. 3. Precision rate on KDD Cup 1999 data set by 10-fold cross-validation. The minimum and maximum precision rates over 10 runs are depicted as circles and the overall precision is depicted as a star sign

Unlike the thirteen data sets described in the paper, this data set is labeled, and so we know if a particular test record is actually anomalous. After partitioning into causal and result attributes, we used a 10-fold cross-validation test to compare the precision of the different detection methods on this data set; error bars showing the low and high accuracy of each method as well as the mean are provided in Figure 3. The results show that this particular task is rather easy, and so it is difficult to draw any additional conclusions from the results. All of the methods except for LOF perform very well, with almost perfect overlap among the error bars.

APPENDIX II EM DERIVATIONS

A. MLE Goal.

The goal is to maximize the objective function:

$$\Lambda = \log \prod_{k=1}^n f_{CAD}(y_k|x_k) = \sum_{k=1}^n \log \left\{ \sum_{i=1}^{n_U} p(x_k \in U_i) \sum_{j=1}^{n_V} f_G(y_k|V_j) p(V_j|U_i) \right\} \quad (1)$$

where:

$$f_{CAD}(y_k|x_k) = \sum_{i=1}^{n_U} p(x_k \in U_i) \sum_{j=1}^{n_V} f_G(y_k|V_j) p(V_j|U_i) \quad (2)$$

B. Expectation Step.

The expectation of the objective function is:

$$\begin{aligned} & Q(\Theta, \bar{\Theta}) \\ &= E[\Lambda(X, Y, \alpha, \beta|X, \bar{\Theta})|X, Y, \Theta] \\ &= E\left[\sum_{k=1}^n \log f_{CAD}(x_k, y_k, \alpha_k, \beta_k|x_k, \bar{\Theta})|X, Y, \Theta\right] \\ &= \sum_{k=1}^n \sum_{\alpha_k} \sum_{\beta_k} \{\log f_{CAD}(y_k, \alpha_k, \beta_k|x_k, \bar{\Theta}) f(\alpha_k, \beta_k|X, Y, \Theta)\} \\ &= \sum_{k=1}^n \sum_{i=1}^{n_U} \sum_{j=1}^{n_V} \{\log f_{CAD}(y_k, \alpha_k^{(i)}, \beta_k^{(j)}|x_k, \bar{\Theta}) \\ &\quad \times f(\alpha_k^{(i)}, \beta_k^{(j)}|x_k, y_k, \Theta)\} \end{aligned} \quad (3)$$

Now we derive the two sub-expressions in (3):

$$\begin{aligned}
& f(\alpha_k^{(i)}, \beta_k^{(j)} | x_k, y_k, \Theta) \\
&= \frac{f(x_k, \alpha_k^{(i)}, y_k, \beta_k^{(j)} | \Theta)}{f(x_k, y_k | \Theta)} \\
&= \frac{f(x_k, \alpha_k^{(i)} | \Theta) f(y_k, \beta_k^{(j)} | x_k, \alpha_k^{(i)}, \Theta)}{f(x_k | \Theta) f(y_k | x_k, \Theta)} \\
&= \frac{f(x_k, \alpha_k^{(i)} | \Theta) f(y_k, \beta_k^{(j)} | x_k, \alpha_k^{(i)}, \Theta)}{f(x_k | \Theta) \sum_{t=1}^{n_U} \left\{ p(x_k \in U_t, \Theta) \sum_{h=1}^{n_V} [f_G(y_k | V_h) p(V_h | U_t, \Theta)] \right\}} \quad (4)
\end{aligned}$$

Since:

$$f(x_k, \alpha_k^{(i)} | \Theta) = f(x_k, U_i | \Theta) = f_G(x_k | U_i) p(U_i) \quad (5)$$

Through (2), we can get:

$$f(y_k, \beta_k^{(j)} | x_k, \alpha_k^{(i)}, \Theta) = f_G(y_k | V_j) p(V_j | U_i, \Theta) \quad (6)$$

$\sum_{i=1}^{n_U} p(x_k \in U_i) = 1$, $p(x_k \in U_t)$ is the membership of x_k in cluster U_t , so:

$$p(x_k \in U_t, \Theta) = \frac{f_G(x_k | U_t) p(U_t)}{\sum_{i=1}^{n_U} f_G(x_k | U_i) p(U_i)} = \frac{f_G(x_k | U_t) p(U_t)}{f(x_k | \Theta)} \quad (7)$$

Inserting (5), (6) and (7) into (4), we get:

$$\begin{aligned}
& \frac{f(\alpha_k^{(i)}, \beta_k^{(j)} | x_k, y_k, \Theta)}{f_G(x_k | U_i) p(U_i) f_G(y_k | V_j) p(V_j | U_i, \Theta)} \\
&= \frac{\sum_{t=1}^{n_U} \sum_{h=1}^{n_V} \{ f_G(x_k | U_t) p(U_t) f_G(y_k | V_h) p(V_h | U_t, \Theta) \}}{\sum_{t=1}^{n_U} \sum_{h=1}^{n_V} \{ f_G(x_k | U_t) p(U_t) f_G(y_k | V_h) p(V_h | U_t, \Theta) \}} \quad (8)
\end{aligned}$$

For simplicity, we introduce b_{kij} :

$$b_{kij} = f(\alpha_k^{(i)}, \beta_k^{(j)} | x_k, y_k, \Theta) \quad (9)$$

With the new denotation b_{kij} , the Q function in (3) becomes:

$$Q(\Theta, \bar{\Theta}) = \sum_{k=1}^n \sum_{i=1}^{n_U} \sum_{j=1}^{n_V} \left\{ \log f_{CAD}(y_k, \alpha_k^{(i)}, \beta_k^{(j)} | x_k, \bar{\Theta}) \times b_{kij} \right\} \quad (10)$$

where:

$$b_{kij} = \frac{f_G(x_k | U_i) p(U_i) f_G(y_k | V_j) p(V_j | U_i, \Theta)}{\sum_{t=1}^{n_U} \sum_{h=1}^{n_V} \{ f_G(x_k | U_t) p(U_t) f_G(y_k | V_h) p(V_h | U_t, \Theta) \}} \quad (11)$$

The remaining sub-expression in Q function is:

$$\begin{aligned}
& f_{CAD}(y_k, \alpha_k^{(i)}, \beta_k^{(j)} | x_k, \bar{\Theta}) \\
&= f(y_k, \beta_k^{(j)} | x_k, \alpha_k^{(i)}, \bar{\Theta}) \times f(\alpha_k^{(i)} | x_k, \bar{\Theta}) \\
&= f_G(y_k | \bar{V}_j) \times p(V_j | U_i, \bar{\Theta}) \times \frac{f_G(x_k | \bar{U}_i) p(\bar{U}_i)}{f(x_k | \bar{\Theta})} \quad (12)
\end{aligned}$$

Inserting (12) into (10), the Q function becomes:

$$\begin{aligned}
& Q(\Theta, \bar{\Theta}) \\
&= \sum_{k=1}^n \sum_{i=1}^{n_U} \sum_{j=1}^{n_V} \left\{ \left[\begin{aligned} & \log f_G(y_k | \bar{V}_j) + \log p(V_j | U_i, \bar{\Theta}) \\ & + \log f_G(x_k | \bar{U}_i) + \log p(\bar{U}_i) \\ & - \log f(x_k | \bar{\Theta}) \end{aligned} \right] \times b_{kij} \right\} \quad (13)
\end{aligned}$$

In the Q function from (13), we have $\log f(x_k | \bar{\Theta}) = \log \sum_{t=1}^{n_U} f_G(x_k | \bar{U}_t) p(\bar{U}_t)$, which is hard to differentiate. We can play an approximation ‘‘trick’’ by using $f(x_k | \Theta)$ to approximate $f(x_k | \bar{\Theta})$. To make the maximization target clear, we can rewrite the (13) as:

$$\begin{aligned}
& Q'(\Theta, \bar{\Theta}) \\
&= \sum_{k=1}^n \sum_{i=1}^{n_U} \sum_{j=1}^{n_V} b_{kij} \log f_G(y_k | \bar{V}_j) + \sum_{k=1}^n \sum_{i=1}^{n_U} \sum_{j=1}^{n_V} b_{kij} \log \overline{p(V_j | U_i)} \\
&+ \sum_{k=1}^n \sum_{i=1}^{n_U} \sum_{j=1}^{n_V} b_{kij} \log f_G(x_k | \bar{U}_i) + \sum_{k=1}^n \sum_{i=1}^{n_U} \sum_{j=1}^{n_V} b_{kij} \log \overline{p(\bar{U}_i)} \\
&- \sum_{k=1}^n \sum_{i=1}^{n_U} \sum_{j=1}^{n_V} b_{kij} \log f(x_k | \Theta) \quad (14)
\end{aligned}$$

We will perform the maximization of Q' by maximizing each of the four terms in (14).

C. Maximization Step.

1) : First, we want to maximize:

$$\sum_{k=1}^n \sum_{i=1}^{n_U} \sum_{j=1}^{n_V} b_{kij} \log \overline{p(\bar{U}_i)} \quad (15)$$

with respect to $\overline{p(\bar{U}_i)}$. Note that the $\overline{p(\bar{U}_i)}$ is constrained by:

$$\sum_{i=1}^{n_U} \overline{p(\bar{U}_i)} = 1 \quad (16)$$

Here, we do not count $\sum_{k=1}^n \sum_{i=1}^{n_U} \sum_{j=1}^{n_V} b_{kij} \log \overline{p(V_j | U_i)}$ when maximizing (15). Through the method of Lagrange multipliers, maximizing (15) is the same as maximizing:

$$h = \sum_{k=1}^n \sum_{i=1}^{n_U} \sum_{j=1}^{n_V} b_{kij} \log \overline{p(\bar{U}_i)} + \lambda \left[\sum_{i=1}^{n_U} \overline{p(\bar{U}_i)} - 1 \right] \quad (17)$$

(11) Taking the derivative of (17) with respect to $\overline{p(\bar{U}_i)}$ and setting it to zero:

$$\begin{aligned}
\frac{\partial h}{\partial \overline{p(\bar{U}_i)}} &= \sum_{k=1}^n \sum_{j=1}^{n_V} \frac{b_{kij}}{\overline{p(\bar{U}_i)}} + \lambda = 0 \\
\sum_{k=1}^n \sum_{j=1}^{n_V} b_{kij} + \lambda \overline{p(\bar{U}_i)} &= 0 \quad (18)
\end{aligned}$$

We now sum (18) over all i and solve for λ :

$$\sum_{h=1}^{n_U} \left[\sum_{k=1}^n \sum_{j=1}^{n_V} b_{khj} + \lambda \overline{p(U_h)} \right] = 0$$

$$\lambda = -\frac{\sum_{k=1}^n \sum_{h=1}^{n_U} \sum_{j=1}^{n_V} b_{khj}}{\sum_{h=1}^{n_U} \overline{p(U_h)}} = -\sum_{k=1}^n \sum_{h=1}^{n_U} \sum_{j=1}^{n_V} b_{khj} \quad (19)$$

Combining (18) and (19) and solving for $\overline{p(U_i)}$,

$$\overline{p(U_i)} = \frac{\sum_{k=1}^n \sum_{j=1}^{n_V} b_{kij}}{\sum_{k=1}^n \sum_{h=1}^{n_U} \sum_{j=1}^{n_V} b_{khj}} \quad i \in \{1, 2, \dots, n_U\} \quad (20)$$

2) : Now we want to maximize:

$$\sum_{k=1}^n \sum_{i=1}^{n_U} \sum_{j=1}^{n_V} b_{kij} \log f_G(y_k | \bar{V}_j) \text{ with respect to } \bar{\mu}_{V_j} \text{ and } \bar{\Sigma}_{V_j}.$$

Recall that by linear algebra, if A is symmetric matrix:

$$\begin{aligned} \nabla_b (b^T A b) &= 2Ab, \\ \nabla_A (b^T A b) &= b b^T, \\ \nabla_A |A| &= A^{-1} |A|. \end{aligned}$$

$$\begin{aligned} &\sum_{k=1}^n \sum_{i=1}^{n_U} \sum_{j=1}^{n_V} b_{kij} \log f_G(y_k | \bar{V}_j) \\ &= \sum_{k=1}^n \sum_{i=1}^{n_U} \sum_{j=1}^{n_V} b_{kij} \log f_G(y_k | \bar{\mu}_{V_j}, \bar{\Sigma}_{V_j}) \\ &= \sum_{k=1}^n \sum_{i=1}^{n_U} \sum_{j=1}^{n_V} b_{kij} \log \frac{\exp[-\frac{1}{2}(y_k - \bar{\mu}_{V_j})^T \bar{\Sigma}_{V_j}^{-1} (y_k - \bar{\mu}_{V_j})]}{(2\pi)^{\frac{d}{2}} |\bar{\Sigma}_{V_j}|^{\frac{1}{2}}} \end{aligned} \quad (21)$$

Taking the derivative of (21) with respect to $\bar{\mu}_{V_j}$ and setting it to zero:

$$\sum_{k=1}^n \sum_{i=1}^{n_U} b_{kij} \left[\frac{1}{2} \times 2 \times \bar{\Sigma}_{V_j}^{-1} \times (y_k - \bar{\mu}_{V_j}) \right] = 0 \quad (22)$$

Solving (22) for $\bar{\mu}_{V_j}, j \in \{1, 2, \dots, n_V\}$:

$$\bar{\mu}_{V_j} = \frac{\sum_{k=1}^n \sum_{i=1}^{n_U} b_{kij} y_k}{\sum_{k=1}^n \sum_{i=1}^{n_U} b_{kij}} \quad (23)$$

Now taking the derivative of (21) w.r.t. $\bar{\Sigma}_{V_j}^{-1}$ and setting it to zero:

$$\sum_{k=1}^n \sum_{i=1}^{n_U} b_{kij} [\bar{\Sigma}_{V_j} - (y_k - \bar{\mu}_{V_j})(y_k - \bar{\mu}_{V_j})^T] = 0 \quad (24)$$

Solving (24) for $\bar{\Sigma}_{V_j}, j \in \{1, 2, \dots, n_V\}$:

$$\bar{\Sigma}_{V_j} = \frac{\sum_{k=1}^n \sum_{i=1}^{n_U} b_{kij} (y_k - \bar{\mu}_{V_j})(y_k - \bar{\mu}_{V_j})^T}{\sum_{k=1}^n \sum_{i=1}^{n_U} b_{kij}} \quad (25)$$

We can maximize $\sum_{k=1}^n \sum_{i=1}^{n_U} \sum_{j=1}^{n_V} b_{kij} \log f_G(x_k | \bar{U}_i)$ in the similar way and get the update rule for $\bar{U}_i = (\bar{\mu}_{U_i}, \bar{\Sigma}_{U_i}), i \in \{1, 2, \dots, n_U\}$:

$$\bar{\mu}_{U_i} = \frac{\sum_{k=1}^n \sum_{j=1}^{n_V} b_{kij} x_k}{\sum_{k=1}^n \sum_{j=1}^{n_V} b_{kij}} \quad (26)$$

$$\bar{\Sigma}_{U_i} = \frac{\sum_{k=1}^n \sum_{j=1}^{n_V} b_{kij} (x_k - \bar{\mu}_{U_i})(x_k - \bar{\mu}_{U_i})^T}{\sum_{k=1}^n \sum_{j=1}^{n_V} b_{kij}} \quad (27)$$

3) : Now we want to maximize:

$$\sum_{k=1}^n \sum_{i=1}^{n_U} \sum_{j=1}^{n_V} b_{kij} \log \overline{p(V_j | U_i)} \quad (28)$$

w.r.t. $\overline{p(V_j | U_i)}$.

We can set a constraint on the $\overline{p(V_j | U_i)}$:

$$\sum_{j=1}^{n_V} \overline{p(V_j | U_i)} = 1 \quad i \in \{1, 2, \dots, n_U\} \quad (29)$$

Again by the method of Lagrange multiplier, maximizing (28) is the same as maximizing:

$$\begin{aligned} h &= \sum_{k=1}^n \sum_{i=1}^{n_U} \sum_{j=1}^{n_V} b_{kij} \log \overline{p(V_j | U_i)} \\ &\quad + \lambda_i \left(\sum_{j=1}^{n_V} \overline{p(V_j | U_i)} - 1 \right), i \in \{1, 2, \dots, n_U\} \end{aligned} \quad (30)$$

Taking the derivative of (30) w.r.t. $\overline{p(V_j | U_i)}$ and setting it to zero:

$$\begin{aligned} \frac{\partial h}{\partial \overline{p(V_j | U_i)}} &= \sum_{k=1}^n \frac{b_{kij}}{\overline{p(V_j | U_i)}} + \lambda_i = 0 \\ \sum_{k=1}^n b_{kij} + \lambda_i \overline{p(V_j | U_i)} &= 0 \end{aligned} \quad (31)$$

We now sum (31) over all j and solve for λ_i :

$$\sum_{h=1}^{n_V} \left[\sum_{k=1}^n b_{kih} + \lambda_i \overline{p(V_h | U_i)} \right] = 0 \quad (32)$$

$$\lambda_i = -\frac{\sum_{k=1}^n \sum_{h=1}^{n_V} b_{kih}}{\sum_{h=1}^{n_V} \overline{p(V_h | U_i)}} = -\sum_{k=1}^n \sum_{h=1}^{n_V} b_{kih} \quad (33)$$

Finally, combining (31) and (33) and solving for $\overline{p(V_j | U_i)}, i \in \{1, 2, \dots, n_U\}, j \in \{1, 2, \dots, n_V\}$ we have:

$$\overline{p(V_j | U_i)} = \frac{\sum_{k=1}^n b_{kij}}{\sum_{k=1}^n \sum_{h=1}^{n_V} b_{kih}} \quad (34)$$

REFERENCES

- [1] M. Markou and S. Singh, "Novelty detection: a review part 1: statistical approaches," *Signal Processing*, vol. 83, pp. 2481 – 2497, Decemeber 2003.
- [2] W.-K. Wong, A. Moore, G. Cooper, and M. Wagner, "Rule-based anomaly pattern detection for detecting disease outbreaks," *AAAI Conference Proceedings*, pp. 217–223, August 2002.
- [3] A. Adam, E. Rivlin, and I. Shimshoni, "Ror: Rejection of outliers by rotations in stereo matching," *Conference on Computer Vision and Pattern Recognition (CVPR-00)*, pp. 1002–1009, June 2000.
- [4] F. de la Torre and M. J. Black, "Robust principal component analysis for computer vision," in *Proceedings of the Eighth International Conference On Computer Vision (ICCV-01)*, 2001, pp. 362–369.
- [5] A. Lazarevic, L. Ertöz, V. Kumar, A. Ozgur, and J. Srivastava, "A comparative study of anomaly detection schemes in network intrusion detection," in *Proceedings of the Third SIAM International Conference on Data Mining*, 2003.
- [6] Y. Zhang and W. Lee, "Intrusion detection in wireless ad-hoc networks," in *MOBICOM*, 2000, pp. 275–283.
- [7] E. M. Knorr, R. T. Ng, and V. Tucakov, "Distance-Based Outliers: Algorithms and Applications," *VLDB Journal*, vol. 8, no. 3-4, pp. 237–253, February 2000.
- [8] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: identifying density-based local outliers," *ACM SIGMOD Conference Proceedings*, pp. 93–104, May 2000.
- [9] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," *ACM SIGMOD Conference Proceedings*, pp. 427–438, May 2000.
- [10] E. Eskin, "Anomaly Detection over Noisy Data using Learned Probability Distributions," *ICML Conference Proceedings*, pp. 255–262, Jun 2000.
- [11] K. Yamanishi, J. ichi Takeuchi, G. J. Williams, and P. Milne, "On-Line Unsupervised Outlier Detection Using Finite Mixtures with Discounting Learning Algorithms," *Data Mining and Knowledge Discovery*, vol. 8, no. 3, pp. 275 – 300, May 2004.
- [12] S. Roberts and L. Tarassenko, "A Probabilistic Resource Allocating Network for Novelty Detection," *NEURAL COMPUTATION*, vol. 6, pp. 270 – 284, March 1994.
- [13] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal Royal Stat. Soc., Series B*, vol. 39, no. 1, pp. 1–38, 1977.
- [14] J. Bilmes, "A gentle tutorial on the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models," *Technical Report, University of Berkeley, ICSI-TR-97-021*, 1997.
- [15] D. M. Chickering and D. Heckerman, "Efficient approximations for the marginal likelihood of bayesian networks with hidden variables," *Machine Learning*, vol. 29, no. 2-3, pp. 181–212, 1997.
- [16] M. A. T. Figueiredo and A. K. Jain, "Unsupervised learning of finite mixture models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 3, pp. 381–396, 2002.
- [17] B. Efron and R. J. Tibshirani, *An introduction to the Bootstrap*. New York: Chapman and Hall, 1993.
- [18] A. Lazarevic, L. Ertöz, V. Kumar, A. Ozgur, and J. Srivastava, "A comparative study of anomaly detection schemes in network intrusion detection," in *SDM*, 2003.
- [19] C. C. Aggarwal and P. S. Yu, "Outlier detection for high dimensional data," *ACM SIGMOD Conference Proceedings*, pp. 37–46, May 2001.
- [20] K. S. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, "When Is "Nearest Neighbor" Meaningful?" *International Conference on Database Theory Proceedings*, pp. 217–235, January 1999.
- [21] M. Kulldorff, "A spatial scan statistic," *Communications in Statistics: Theory and Methods*, vol. 26, no. 6, pp. 1481–1496, 1997.
- [22] D. B. Neill and A. W. Moore, "Rapid detection of significant spatial clusters," in *KDD '04: Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining*, 2004, pp. 256–265.
- [23] M. Kulldorff, "Spatial scan statistics: models, calculations, and applications," in *Scan Statistics and Applications*, edited by Glaz and Balakrishnan, pp. 303–322, 1999.
- [24] W.-K. Wong, A. Moore, G. Cooper, and M. Wagner, "Bayesian Network Anomaly Pattern Detection for Disease Outbreaks," *ICML Conference Proceedings*, pp. 808–815, August 2003.
- [25] L. Breiman, R. A. Olshen, J. H. Friedman, and C. J. Stone, "Classification and regression trees," 1984.
- [26] J. R. Quinlan, "Learning with Continuous Classes," *the 5th Australian Joint Conference on Artificial Intelligence Proceedings*, pp. 343–348, 1992.
- [27] P. S. Bradley, U. M. Fayyad, and C. Reina, "Scaling clustering algorithms to large databases," in *Knowledge Discovery and Data Mining*, 1998, pp. 9–15.



in large databases.



Xiuyao Song is working towards a PhD degree in the Computer and Information Sciences and Engineering Department at the University of Florida. She received the Bachelor's and Master's degree from Computer Science Department at the Huazhong University of Science and Technology, China in 1998 and 2001 respectively. After that, she spent a year as a project management specialist in Lucent Technologies (China) Co., Ltd. She began her PhD program in September 2002 and her research interest is data mining, more specifically, anomaly detection

Mingxi Wu is a Ph.D student in the Computer and Information Science and Engineering Department at University of Florida (UF). Prior to joining UF, he received his B.S. degree in Computer Science from Fudan University in 2000. After that, he worked in Microsoft for two years as a global technical support engineer. His current research includes outlier/anomaly detection in high dimensional data sets and top-k related database query processing.



Christopher Jermaine is an assistant professor in the CISE Department at the University of Florida, Gainesville. His research interests are in the area of databases, data mining, and applied statistics, with an emphasis on data analysis and randomized algorithms. He received his doctorate from the Georgia Institute of Technology.



Sanjay Ranka is a Professor of Computer Information Science and Engineering at University of Florida, Gainesville. His research interests are the areas of large-scale software systems, grid computing, high performance computing, data mining, biomedical computing and optimization. Most recently, he was the Chief Technology Officer at Paramark, where he developed real-time optimization software for optimizing marketing campaigns. He has also held positions as a tenured faculty positions at Syracuse University and as a researcher/visitor at IBM T.J. Watson Research Labs and Hitachi America Limited.

He has co-authored two books: Elements of Artificial Neural Networks (MIT Press) and Hypercube Algorithms (Springer Verlag). He has also coauthored 6 book chapters, 55+ journal articles and 100+ conference and workshop articles. He was one of the main architects of the Syracuse Fortran 90D/HPF compiler and served on the MPI Standards committee. He is a fellow of the IEEE and a member of IFIP Committee on System Modeling and Optimization.