A Sparse Bayesian Framework for Anomaly Detection in Heterogeneous Networks

Jian Zhang and Rajgopal Kannan

Dept. Computer Science
Louisiana State University
Baton Rouge, LA 70803
{zhang,rkannan}@csc.lsu.edu

Abstract. The capability to detect anomalous states in a network is important for both the smooth operation of the network and the security of the network. Modern networks are often heterogeneous. This raises a new challenge for anomaly detection, as there may be a wide variety of anomalous activities across the heterogeneous components of a network. We often seek a detection system that not only performs accurate anomaly detection but also provides mechanisms for human expert to understand the decision making process inside the system. In this paper, we investigate the application of sparse Bayesian methods for anomaly detection in such scenario. By taking advantage of the sparse Bayesian framework's capability to conduct automatic relevance discovery, we construct a detection system whose decision making is mostly based on a few representative examples from the training set. This provides human interpretability as expert can analyze the representative examples to understand the detection mechanism. Our experiment results show the potential of this approach.

Keywords: Anomaly Detection, Sparse Bayesian Classification.

1 Introduction

In a complex system, it is crucial to monitor the system activities for the normal operation of the system. Techniques that can identify anomalous state in a system have been the subject of many researches [7,1,11,3]. Machine learning methods are a particularly promising approach to the problem. However, in this case, it is often not enough to just classify a system event or activity to be normal or abnormal. We also want to know why the event is abnormal. And when there are many different types of abnormal events, we want to know what the types are. This is important because detecting the anomaly is just a start. To deal with the problem, e.g., to recover from the abnormal state or to fight against security breach, one need to know more.

Many off-the-shelf machine learning methods lack the capability to explain the decision made by the methods. For example, given a particular trained neural network for anomaly detection, it is not easy to derive a human-understandable

X. Zhang and D. Qiao (Eds.): QShine 2010, LNICST 74, pp. 75–87, 2011.

model on how the neural network judges some event to be normal and the others to be abnormal. On the other hand, in many cases, the detection problem can be complex. There may be many different types of anomalies. And we may need to combine many features of an event to specify the anomaly. In these scenarios, human analysis is often needed to explain the anomalous event and identify the type of the anomaly. However, automated data analysis program can still help. In particular, we can use automated programs to select a few "representative" cases and present them to the human experts. The expert then analyze the "representative" cases and build models for the (different types of) anomalies. In this way, automated analysis facilitates human analysis by reducing the amount of cases that the experts need to look through.

In this paper, we propose a sparse Bayes framework for anomaly detection. Given a set of normal and abnormal examples, we build a classifier that detects future anomalies. The classification is based on a very small set of examples and these examples can be viewed as the "representative" cases of the abnormal activities. By analyzing this small set of examples, a human expert may identify the mechanism that causes the anomaly.

One way to build a small set of "representative" cases from a collection is to perform clustering. Clustering groups the data into clusters and the cluster centers can serve as the "representative" cases. However, several factors make clustering less applicable in our scenario: 1) We may not know how many number of clusters exist in our collection of events. 2) In many cases, the "center" of a cluster is not well-defined when the attributes describing the event involves nominal attributes. For example, if a cluster contains some events that involves TCP as the connection protocol and some other involves UDP. There is no center that "averages" over the two protocols. 3) We seek "representative" cases that not only represent the data in the collection, but also can help in making the distinction between the abnormal and the normal. i.e., The "representative" cases should play an important role in the classification of the events.

In the sparse Bayes framework, the search for the representative cases (events) is performed while the classifier for anomaly detection is being constructed. The representative cases (events) are also the ones that are important for the classification. In fact, to classify an unknown event, one only needs to compare the event to the representative events and the result of such comparison decides whether the unknown event is normal or abnormal. When the representative events can lead to accurate classification, they represent well the data in the collection of examples.

We test our sparse Bayes framework for anomaly detection using the KDD intrusion detection dataset. Experiment results show that a classifier constructed following the sparse Bayes framework gives accurate classification between the normal and the abnormal events. The performance of the classifier is at the same level as that of a classifier based on support vector machine. Our classification framework also generates a very small set of representative cases, which human expert can analyze to understand the anomaly or determine the types of the anomaly.

The rest of the paper is organized as follows: In Section 2, we discuss related work. In Section 3, we give a detailed description of our sparse Bayesian framework for anomaly detection. In Section 4, we present and discuss our experiment results. We conclude the paper in Section 5.

2 Related Work

Anomaly detection has been the subject of many researches. There are excellent surveys on the problem, the approaches considered and the applications. We refer to [7,1,11,3] for comprehensive elaboration on the topic. Intrusion detection is a major application area of anomaly detection and many systems have been proposed, for example, [13,12,2]. Statistical models and machine learning methods are heavily employed in anomaly and intrusion detection systems.

For example, one popular approach in intrusion detection is to use rule-based systems. A set of rules specifying abnormal events (intrusions) is extracted and new intrusion events are identified by matching them to these rules. The rules can be extracted either manually, or through data mining or machine learning techniques. Decision trees [10] and fuzzy logic [5] have been used for intrusions detection. Rule learning algorithms such as RIPPER have also been tested for such purpose [14]. The advantage of these techniques is that the rules they use to make decisions can be easily interpreted by humans. Pure machine learning techniques such as neural networks [8] and support vector machines [4] are also employed in intrusion and anomaly detection.

Sparse Bayesian learning is a learning technique developed by Tipping [15,6]. The goal of this framework is to construct a parsimonious model that can give good prediction. The framework achieves this by select features using an automatic relevance detection technique. A classifier constructed in this framework is called a relevance vector machine. The relevance vector machine has been applied to help dynamically tracking faces in video sequences [16].

3 Sparse Bayesian Framework for Anomaly Detection

We specify a network event as a vector of attributes. Some are numerical and the others are nominal. We have a collection of N examples of normal and abnormal events, which we denote as $\{x_i, t_i\}^N$, where $x_i = \{x_i^1, x_i^2, \ldots, x_i^k\}$ is the vector of k attributes describing the i-th example event. And t_i is the class label (normal or abnormal) of the event.

Our goal is to construct a classifier to classify future unknown events. Normally, the classification would be based on a function of attributes. For example, one may learn a function in the form of $\sum_j w_j x^j + b$ and classifies the unknown event x by the sign of the function values. Our framework takes a different approach. The classification function is based on the similarity measure between the unknown event and the events in the training collection. Our approach bears some similarity to the nearest-neighbor classification. However, instead of compare the unknown event to the nearest neighbors, we compare it to a small set of examples that are selected according to a statistical process that automatically discovers the relevance between the events.

3.1 Similarity Based on Normalized Features

The first step in our sparse Bayesian framework is to calculate similarity between network events. Given two network event x_i and x_j, we measure the similarity between the two event by the "extended" dot product of the two corresponding feature vectors. i.e., $S(x_i, x_j) = \sum_k x_i^k \circ x_j^k$. We call it the "extended" dot product because some features are nominal and the others are numerical. For a numerical feature (x_i^k and x_j^k), $x_i^k \circ x_j^k$ is the product of x_i^k and x_j^k. When the feature is nominal, we define

$$x_i^k \circ x_j^k = \begin{cases} 1 & \text{if } x_i^k = x_j^k; \\ 0 & \text{otherwise;} \end{cases} \tag{1}$$

One problem in calculating similarity in such fashion is that the result may be dominated by one or a few features that has very large values compare to the other features. (In fact, this is a common problem for similarity calculation, not just for our extended dot product.) To deal with this situation, we normalize the features before performing the extended dot product. Clearly, only the numerical features require normalization. Consider the vector of values for the k-th feature $k = \{x_1^k, x_2^k, \ldots, x_N^k\}$. We view that the values follow a normal distribution. We then transform each value x_i^k to be

$$\phi(x_i^k) = \frac{x_i^k - \mu_k}{\sigma_k}. \tag{2}$$

where $\mu_k = \frac{1}{N} \sum_i x_i^k$ is the mean of the values for the k-th feature and σ_k is the standard deviation of the values. Once the feature values are normalized, the similarity calculation is done with the transformed values. Let O be the set of nominal features and U be the set of numerical features, we have

$$S(i, j) = \sum_{k \in U} \phi(x_i^k)\phi(x_j^k) + \sum_{k \in O} x_i^k \circ x_j^k. \tag{3}$$

3.2 Sparse Bayesian Classification

We follow a logistic regression model to define the probability of an event being normal or abnormal. Given the similarity measure S and a network event x, let t be the type ("normal" and "abnormal") of the event, we define the probability of the event being normal or abnormal to be

$$p(t = \text{“abnormal”}|x, w) = \frac{1}{1 + \exp(\sum_i w_i S(x, x_i) + w_0)}$$

and

$$p(t = \text{“normal”}|x, w) = 1 - p(t = \text{“normal”}|x, w) = \frac{\exp(\sum_i w_i S(x, x_i) + w_0)}{1 + \exp(\sum_i w_i S(x, x_i) + w_0)}.$$

$w = \{w_0, w_1, \ldots, w_N\}$ is a vector of parameters for the probability model. In a general logistic regression model, the probability is defined as a function of features, i.e., it involves a quantity in the form of $\sum_k \hat{w}_k x^k$. Our model differs from the standard logistic regression in that the probability is defined as a function of training instances. Therefore, each w_i, $i \neq 0$ corresponds to an instance (event) in the training set (rather than a feature as is in a standard logistic regression model). If we encode the "abnormal" case as $t = 1$ and the "normal" case as $t = 0$, we have the following likelihood of the event given the parameter vector:

$$p(t|x, w) = \left(\frac{1}{1 + \exp(\sum_i w_i S(x, x_i) + w_0)}\right)^t \left(1 - \frac{1}{1 + \exp(\sum_i w_i S(x, x_i) + w_0)}\right)^{(1-t)} \tag{4}$$

Let $y(x) = \frac{1}{1+\exp(\sum_i w_i S(x,x_i)+w_0)}$, the likelihood can be simplified as $y(x)^t(1 - y(x))^{(1-t)}$.

One may estimate the parameters using a maximum likelihood approach. In Bayesian framework, rather than selecting a particular vector of values for w, we often model the parameter vector as a random variable draw from a prior distribution $p(w)$. Note that in many cases, as is in this paper, it is not necessarily to use a distribution that matches exactly the actual prior distribution of the problem. We often treat the model prior as a way to encode our bias in choosing the model parameters.

Following the sparse Bayesian framework of [15], we consider a particular prior distribution for w. We model that the i-th parameter w_i is draw from a normal distribution with zero mean and variance a_i, i.e., $p(w_i) \sim N(0, a_i)$. The distribution for the whole parameter vector w is a multivariate normal distribution with zero mean and a covariance matrix A, i.e.,

$$p(w|A) = N(w|\mathbf{0}, A). \tag{5}$$

where $\mathbf{0}$ is a $(N + 1)$-dimensional all zero vector and A is a diagonal matrix with the diagonal $A_{ii} = a_i$.

The normal and abnormal probability of a new network event x now can be calculated as

$$p(t|x, t_{\text{train}}, X_{\text{train}}, A) = \int p(t|x, w)p(w|t_{\text{train}}, X_{\text{train}}, A)dw \tag{6}$$

with $p(t|x, w)$ defined by Eq. 4 and $p(w|t_{\text{train}}, X_{\text{train}}, A)$ being the posterior distribution of w given the prior distribution $p(w|A)$ and the set of training example $\{t_i, x_i\}^N$. ($t_{\text{train}} = \{t_1, t_2, \ldots, t_N\}^t$ is the N-dimensional vector of class labels for the training examples. $X_{\text{train}} = \{S_{\text{train}}, \mathbf{1}\}$ is constructed by appending an N-dimensional all ones column vector to the right of the similarity matrix S_{train} of the training examples.) The probability of the new network event being normal or abnormal is determined by both the prior distribution (specified by A) and the training data.

3.3 Parameter Estimation

We follow the approach in [15] to estimate the parameters a_i. We give here a brief introduction to the parameter estimation process. Detailed description can be found in [15]. We first consider the posterior distribution of w, supposing that the values in A is given. Following the Bayes rule, we can write the posterior distribution as

$$p(w|t_{\text{train}}, X_{\text{train}}, A) = \frac{p(t_{\text{train}}|X_{\text{train}}, w)p(w|A)}{p(t_{\text{train}}|X_{\text{train}}, A)}.$$

Instead of considering the exact posterior distribution, we apply Laplace approximation to the distribution, i.e., we construct a normal distribution centered at the mode of the posterior distribution and use the normal distribution to approximate the posterior distribution.

The mode of the posterior distribution can be found by maximizing the log probability of the distribution:

$$\begin{aligned} &\ln p(w|t_{\text{train}}, X_{\text{train}}, A) \\ &= \ln p(t_{\text{train}}|X_{\text{train}}, w)p(w|A) - \ln p(t_{\text{train}}|X_{\text{train}}, A) \\ &= \sum_{i=1}^{N}\{t_i \ln y_i + (1-t_i)\ln(1-y_i)\} - \frac{1}{2}w^t A w + C \end{aligned}$$

where $y_i = y(x_i) = \frac{1}{1+\exp(\sum_j w_j S(x_i,x_j)+w_0)}$ and C is some quantity not dependent on w.

Suppose A is known. The log probability is a function of w. To find the w that maximize the log probability, we set the gradient of the log probability to zero, i.e.,

$$\bigtriangledown \ln p(w|t_{\text{train}}, X_{\text{train}}, A) = X_{\text{train}}^t(t_{\text{train}} - y) - Aw = 0$$

where $y = \{y_1, y_2, \ldots, y_N\}^t$. This is a system of N nonlinear equations. One can obtain the solution (roots) of the equations numerically using Newton-Raphson iterative method, which leads to a type of iterative reweighted least square (IRLS) problem. In particular, the Hessan of the log likelihood is $\bigtriangledown\bigtriangledown \ln p(w|t_{\text{train}}, X_{\text{train}}, A) = -(X_{\text{train}}^t B X_{\text{train}} + A)$ where B is a diagonal matrix with $B_{ii} = y_i(1-y_i)$. The update rule for the Newton-Raphson method would then be:

$$w_{\text{new}} = w_{\text{old}} - (X_{\text{train}}^t B X_{\text{train}} + A)^{-1}(X_{\text{train}}^t(y - t_{\text{train}}) + Aw_{mathrmold})$$

with y evaluated using w_{old} in each step.

Once we obtain $w^* \approx \arg\max_w \ln p(w|t_{\text{train}}, X_{\text{train}}, A)$, we approximate $p(w|t_{\text{train}}, X_{\text{train}}, A)$ by a normal distribution $N(w^*, (X_{\text{train}}^t B^* X_{\text{train}} + A)^{-1})$, where B^* is B with y evaluated using w^*.

The above steps assume that we know the values of a_i in A. The parameters a_i can be estimated following an empirical Bayes approach. That is, we set A to be $A = \arg\max_A \int p(t_{\text{train}}|X_{\text{train}}, w)p(w|A)dw$ where $p(t_{\text{train}}|X_{\text{train}}, w)$ follows Eq. 4 and $p(w|A)$ follows Eq. 5. Unfortunately, there is no close-form solution to this integration. We again apply Laplace approximation. Note that the w^* that maximize the function $f(w) = p(t_{\text{train}}|X_{\text{train}}, w)p(w|A)$ is exactly the w^* we obtain in the approximation of the posterior distribution. With this, the probability of the training examples can be written as:

$$\begin{aligned} & p(t_{\text{train}}|X_{\text{train}}, A) \\ = & \int p(t_{\text{train}}|X_{\text{train}}, w)p(w|A)dw \\ \approx & \, p(t_{\text{train}}|X_{\text{train}}, w^*)p(w^*|A)(2\pi)^{(N+1)/2}|\Sigma^*|^{1/2}. \end{aligned}$$

When we know w^* (and Σ^*), we can obtain A by maximizing the above probability. Set the derivative of this probability to zero, one can obtain the update rule for the parameters a_i as:

$$a_i^{(\text{new})} = \frac{1 - a_i^{(\text{old})}\Sigma^*_{ii}}{(w_i^*)^2} \tag{7}$$

Combining all the above, we have that, given an initial set of values of a_i, we can estimate w^*. With the values of w^*, we can then obtain an updated estimate of a_i. One can repeat this estimation process until a good estimate of the a_i is obtained. The algorithm for parameter estimation is summarized in Algorithm 1.

```
Input  : A set of $N$ examples $\{x_i, t_i\}$; Convergence criteria $\delta_a$, $\delta_w$ and $M$
Output: The parameters $\{a_i\}_{i=1}^N$
Process the training examples to obtain the similarity matrix.
$S(i,j) \leftarrow \sum_{k \in U} \phi(x_{ik})\phi(x_{jk}) + \sum_{k \in O} x_{ik} \circ x_{jk}$, where $\phi$ and $\circ$ are defined in Eq. 1 and Eq. 2 respectively. $O$ is the set of nominal attributes and $U$ is the set of numerical attributes.
$X \leftarrow \{S, \mathbf{1}\}$ ;
while $max(a_i) < M$ and change of $a_i$ larger than $\delta_a$ do
    $w \leftarrow \mathbf{0}$;
    while change of $w_i$ large than $\delta_w$ do
        $y_i \leftarrow \frac{1}{1+\exp(\sum_{j \neq i} w_j S(x_i, x_j) + w_0)}$;
        $B \leftarrow \text{diag}(y)$, $\text{diag}(y)$ constructs a diagonal matrix whose diagonal is $y$;
        $w \leftarrow w - (X^t B X + A)^{-1}(X^t(y - t) + Aw)$;
    end
    $\Sigma \leftarrow (X^t B X + A)^{-1}$;
    $a_i \leftarrow \frac{1 - a_i \Sigma_{ii}}{(w_i)^2}$, where $\Sigma_{ii}$ is the $i$-th element on the diagonal of the matrix $\Sigma$;
end
```

Algorithm 1. Parameter Estimate

3.4 Simplified Classification

In a full Bayesian framework, the probability of a new network event being normal or abnormal is calculated using Eq. 6. Since there is no close-form solution to the integration, one either employs numerical methods or applies an approximation (e.g., the one we used for parameter estimation). This will either increase computational cost or produce less accurate result.

To classify new network events (after we have obtained the values for a_i), instead of following Eq. 6, we can calculate the probability following Eq. 4 using a $w^* = \arg\max_w p(w|t_{\text{train}}, X_{\text{train}}, A)$. This is similar to making prediction by a MAP (maximum a posterior) approach. We now show that if the posterior distribution of w is approximated using a Normal distribution, classifying new network events following the Bayesian approach and the MAP approach generates the same result.

Theorem 1. *Assume that the posterior distribution $p(w|t_{\text{train}}, X_{\text{train}}, A)$ is approximated by a normal distribution $N(w^*, \Sigma^*)$. Classification using $p^{\text{MAP}}(t) = p(t|x, w^*) = y(x)^t(1 - y(x))^{(1-t)}$ and $p^{\text{Bayes}}(t) = p(t|x, t_{\text{train}}, X_{\text{train}}, A) = \int p(t|x, w)p(w|t_{\text{train}}, X_{\text{train}}, A)dw \approx \int p(t|x, w)N(w|w^*, \Sigma^*)dw$ yields the same result.*

Proof. Consider classification using $p^{\text{Bayes}}(t)$. We say a network event x is abnormal if $p^{\text{Bayes}}(1) > p^{\text{Bayes}}(0)$ and normal otherwise. We calculate

$$\begin{aligned}
&\int p(1|x, w)N(w|w^*, \Sigma^*)dw - \int p(0|x, w)N(w|w^*, \Sigma^*)dw \\
&= \int [p(1|x, w) - p(0|x, w)]N(w|w^*, \Sigma^*)dw \\
&= \int [\frac{2}{1 + exp(-u)} - 1]N(u|\mu, \sigma)du
\end{aligned}$$

In the last step, we set $u = w^t x$. For a fixed x, because w follows a normal distribution $N(w^*, \Sigma^*)$, $u = w^t x$ also follows a normal distribution with the mean $\mu = w^{*t} x$.

Let $f(u) = \frac{2}{1+exp(-u)} - 1$. Note that $f(u)$ is symmetric with respect to the origin, i.e., $f(-u) = -f(u)$ and $f(0) = 0$. Also, $f(u)$ is a monotonically increasing function. The classification depends on $\int f(u)N(u|\mu, \sigma)du$. Let x_0 be an event such that $\mu(x_0) = w^{*t}x_0 > 0$. For every $u_0 = \mu(x_0) - \epsilon$, there is $u_0' = \mu(x_0) + \epsilon$, such that $p(u_0) = p(u_0')$ and $|f(u_0)| < |f(u_0')|$ where $|\cdot|$ denotes the absolute value. Therefore, $\int f(u)N(u|\mu(x_0), \sigma)du >> 0$ and the event x_0 will be classified as abnormal using p^{Bayes}. When using p^{MAP}, classification depends on $f(u)$ with $u = w^{*t}x$. Because $w^{*t}x_0 > 0$, $f(w^{*t}x_0) > 0$. Therefore, x_0 will be classified as abnormal too. The same argument holds for an event x_0 such that $w^{*t}x_0 \leq 0$. In this case, both methods will classify the event as normal. □

4 Experiments

We test our framework using the KDD Cup anomaly detection dataset [9]. The dataset is the most widely used data set for evaluation in anomaly detection research. It is based on the data captured in DARPA98 IDS evaluation program. The dataset is a collection of connection vectors. Each vector has 41 features and is labeled either as normal or an attack. The dataset we use is a subsample of the original KDD Cup dataset called the 10-percent KDD Cup dataset. There are 22 attack types in the 10-percent subset.

We first conduct experiments to test the detection performance of our framework and compare it to that of an SVM-based detection system. We then analyze the representative instances selected by our framework.

4.1 Detection Performance

From the 10-percent KDD Cup dataset, we randomly sample a subset of normal and attack instances as training examples. From the remaining of the instances, we then randomly sample another subset to use as test data. We train a SVM-based classifier and a sparse Bayes classifier using the training examples. The performance of the classifiers are then tested on the test data. For each experiment condition, we repeat the experiment 20 times. Each data point reported in the figures is the average of the results from the 20 repetitions.

We measure the performance of the classifiers using the standard F-score. There are only two classes in our data: normal and attack. If a classifier label a true attack as attack, we count it as a true positive. Otherwise if it labels a true attack as normal, we count the case as a false negative. Similarly, we have true negative (label a normal instance as normal) and false positive (label a normal instance as attack). We denote by TP the number of true positives, FN the number of false negatives, TN true negatives and FP false positive. The precision of a classification is define as $TP/(TP + FP)$ and the recall as $TP/(TP+FN)$. The harmonic mean of the precision and the recall is the F-score of the classification performance, i.e., $2\frac{\cdot\text{precision}\times\text{recall}}{\text{precision}+\text{recall}}$. If a classifier gives perfect classification, the F-score should take the value 1 and the worst classification will give an F-score zero.

Fig. 1 shows the performance of the SVM-based classifier and the sparse Bayes classifier when given different numbers of training examples. 4 experiments are conducted using training set of different sizes. In experiment 1, the collection of training examples contain 9 normal connection vectors and 11 attack vectors. In experiment 2, 19 normal and 23 attack vectors. Experiment 3, 48 normal and 59 attacks and experiment 4, 97 normal and 118 attacks. The result shows that for both classifiers, the performance becomes better if more training examples are used. Overall, the SVM-based classifier and the sparse Bayes classifier have similar level of performance.

In the above experiment, attack instances of all types are grouped into one collection. (We omitted some attack types in the original data set that only have few instances.) The training and the testing instances for the attack class

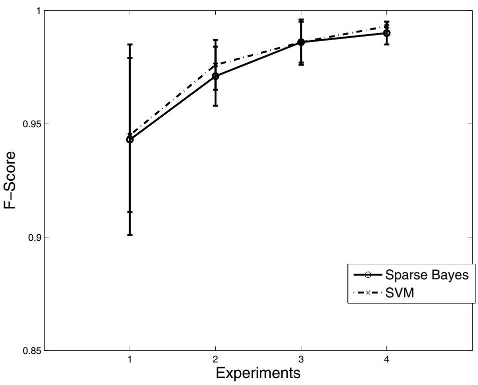

Fig. 1. Detection Performance Given Training Set of Different Size

are sampled from this collection. Intuitively, one may think that the detection task would be more difficult when there are many different types of attacks. The classifier need to recognize many types rather than dealing with one type of attack (anomaly). The number of attack types in the dataset then reflects, to some extent, the complexity of the detection task. To test the performance of the two classifiers on tasks of different complexity, we constructed datasets in which the attack class contains limited types of attacks. We then conducted experiments using these datasets.

Fig. 2 plots the results of the experiments. In experiment 1, the attack data only contain one type of attacks (portsweep). In experiment 2, the data contain two types of attacks (portsweep, satan). In the experiment 3, 4 types of attacks are used (portsweep, satan, ipsweep and warezclient). The results show that both classifiers perform slightly but not significantly better when the task is less complex (involving less types of attacks). The results again show that the performance of the two classifiers are comparable across detection tasks of different complexity.

4.2 The Representative Instances

As discussed before, our goal of applying sparse Bayes framework to anomaly detection is not simply to construct a detection system. We also want the system to help human analysis of the anomaly. In particular, we use the sparse Bayes framework to identify a small set of representative anomaly cases such that by examining this small set, human expert can gain better understanding of the anomalous activities and better understanding of the subtypes inside the abnormal class, if there is any.

We constructed a collection of normal and attack vectors in which the attack class contains 4 types of attacks (portsweep, satan, warezclient and ipsweep). We then train a sparse Bayes classifier from the collection. Recall that the probability of an unknown vector x belonging to normal or abnormal (attack) class

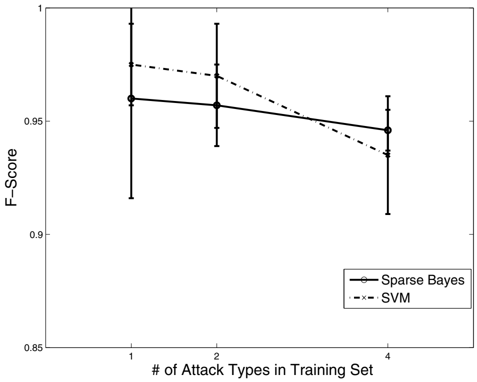

Fig. 2. Detection Performance with Different Number of Attack Types in Training Set

is essentially determined by $\sum_{i \in \text{TraingSet}} w_i S(x, x_i)$. Therefore, the training examples x_i whose corresponding weight w_i has a large magnitude decides the outcome of the classification. We thus view these examples as representative instances. Fig. 3 plot the w vector obtained after training. The dash-doted vertical lines on the figure separates different classes. For example, the first section (from instance 1 to instance 97) contains the weights for the training instances in the normal class. The second section contains the weights for the attacks of the type portsweep. (third section, satan; 4th section, warezclient and the last section ipsweep.) The plot shows that, except for the normal class, most w_i has a value zero or close to zero. There are only few w_i that has large magnitude. The corresponding instances are the representative instances.

We select a few representative instances and plot the similarity measure S between the representative instances and the other instances in the training data. Fig. 4 shows the similarity matrix. Each row of the similarity matrix corresponds to an instance in the training data and each column corresponds to a representative instance. The (i, j)-entry of the matrix is the similarity measure between the i-th training example and the j-th representative instance. We plot the matrix

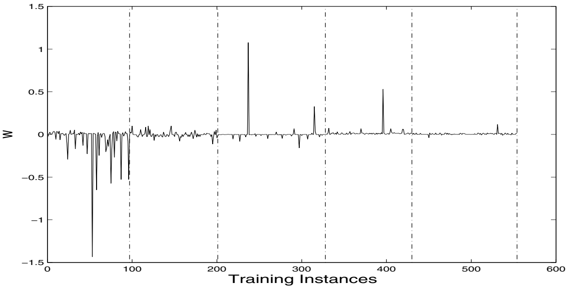

Fig. 3. Weight of the Training Instances

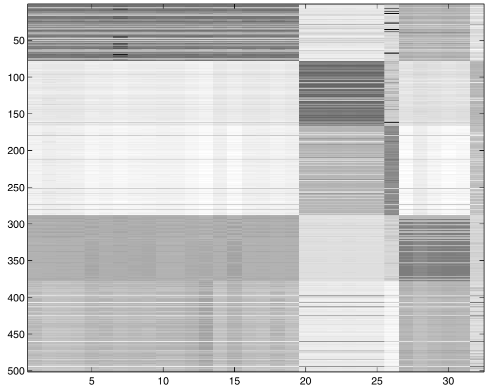

Fig. 4. Similarity between the Representative Instances and the Other Training Examples

using grayscale, with dark indicating high similarity and white low similarity. We observe that the similarity matrix divides into several blocks. Columns 1-19 correspond to the representative instances from the normal class and they have the strongest similarity to the normal examples (row 1-78). The representative instances for the second class (portsweep) lie on column 20-25. Again they have the strongest similarity to the training examples in the portsweep class (row 79-166). In general (except the last group) the representative instances have the highest similarity to the instances in the group to which they belong.

5 Conclusion

We investigate the application of sparse Bayes classification in anomaly detection where there are multiple types of anomalies. Our goal is to construct a detection system that not only identifies anomaly but also facilitates human analysis of the anomalous activities. It does so by selecting a small set of instances that are important for making the detection. These instances can be viewed as representative instances for the anomalies. By analyzing the representative instances, an expert may determine the type and the mechanism of the anomaly. When an unknown event is detected as anomaly due to its connection to a representative instance, corresponding response can be applied to deal with the new anomaly. We conduct experiments to test our framework. The results show that the system based on our framework has similar detection performance comparing to an SVM-based detection system. Furthermore, a few representative instances can be identified by our framework that can be potentially helpful for expert analysis.

Acknowledgment. This work was supported in part by NSF grant IIS-0905478 and Louisiana Board of Regents grant LEQSF(2009-11)-RD-A-09.

References

1. Agyemang, M., Barker, K., Alhajj, R.: A comprehensive survey of numeric and symbolic outlier mining techniques. Intell. Data Anal. 10(6), 521–538 (2006)
2. Beale, J., Caswell, B., Poor, M.: Snort 2.1 intrusion detection. Syngress Publishing (2004)
3. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: A survey. ACM Comput. Surv. 41(3) (2009)
4. Chen, W.-H., Hsu, S.-H., Shen, H.-P.: Application of SVM and ANN for intrusion detection. Computers & Oper. Res. 32, 2617–2634 (2005)
5. Dickerson, J.E., Dickerson, J.A.: Fuzzy network profiling for intrusion detection. In: 19th International Conference of the North American Fuzzy Information Processing Society (2000)
6. Faul, A.C., Tipping, M.E.: Analysis of sparse bayesian learning. In: Advances in Neural Information Processing Systems, pp. 383–389 (2001)
7. Hodge, V., Austin, J.: A survey of outlier detection methodologies. Artificial Intelligence Review 22(2), 85–126 (2004)
8. Kayacik, H.G., Zincir-Heywood, A.N., Heywood, M.I.: A hierarchical SOM-based intrusion detection system. Eng. Appl. of AI 20(4), 439–451 (2007)
9. KDD. Kdd cup intrusion detection dataset (1999), `http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`
10. Levin, I.: KDD-99 classifier learning contest: LLSoft's results overview. SIGKDD Explorations 1(2), 67–75 (2000)
11. Patcha, A., Park, J.-M.: An overview of anomaly detection techniques: Existing solutions and latest technological trends. Computer Networks 51(12), 3448–3470 (2007)
12. Paxson, V.: Bro: a system for detecting network intruders in real-time. In: Proceedings of the 7th USENIX Security Symposium (1998)
13. Porras, P.A., Neumann, P.G.: Emerlad. In: Proceedings of 20th National Information Systems Security Conference, pp. 353–365 (1997)
14. Stolfo, S., Prodromidis, A., Tselepsis, S., Lee, W., Fan, D., Chan, P.: JAM: Java agents for meta-learning over distributed databases. In: Workshop on Fraud Detection and Risk Management AAAI 1997 (1997)
15. Tipping, M.E.: Sparse bayesian learning and the relevance vector machine. Journal of Machine Learning Research 1, 211–244 (2001)
16. Williams, O., Blake, A., Cipolla, R.: Sparse bayesian learning for efficient visual tracking. IEEE Trans. Pattern Analysis and Machine Intelligence 27(8), 1292–1304 (2005)