Anomaly Detection Enhanced Classification in Computer Intrusion Detection

Mike Fugate and James R. Gattiker
{ fugate,gatt}@lanl.gov

Los Alamos National Laboratory

Abstract. This paper describes experiences and results applying Support Vector Machine (SVM) to a Computer Intrusion Detection (CID) dataset. This is the second stage of work with this dataset, emphasizing incorporation of anomaly detection in the modeling and prediction of cyber–attacks. The SVM method for classification is used as a benchmark method (from previous study [1]), and the anomaly detection approaches compare so–called "one class" SVMs with a thresholded Mahalanobis distance to define support regions. Results compare the performance of the methods, and investigate joint performance of classification and anomaly detection. The dataset used is the DARPA/KDD-99 publicly available dataset of features from network packets classified into non–attack and four attack categories.

1 Introduction

This paper describes work with the goal of enhancing capabilities in computer intrusion detection. The work builds upon a study of classification performance, that compared various methods of classifying information derived from computer network packets into attack versus normal categories, based on a labeled training dataset[1]. This previous work examines the well-studied dataset and the classification task, described various approaches to modeling the data emphasizing the application of SVMs which had not been applied previously to this data, and validates our classification methods compared to other studies with detailed presentation of performance. The previous work clears the way through exploratory data analysis and model validation, toward studying whether and how anomaly detection can be used to enhance performance. The DARPA project that initiated the dataset used here concluded that anomaly detection

should be examined to boost the performance of machine learning in the computer intrusion detection task[2].

In this discussion, the term *anomaly detection* will mean making a model from unlabeled data, and using this model to make some inference about future (or hold–out) data. Our data is a feature vector derived from network packets, which we will call an "example" or "sample". On the other hand, *classification* will mean building a model from labeled data, and using that model to classify future (or hold–out) examples.

One technique to meld these approaches is to stage the two techniques, using anomaly detection to segment data into two sets for classification. In our previous work, we observe that the data has substantial nonstationarity[1] between the training set and the temporally and procedurally distinct test set. With classification methods that can be thought of as learning a decision surface between two statistical distributions, performance is expected to degrade significantly when classifying examples that are from regions not well represented in the training set. Anomaly detection can be seen as a problem of learning the *density* (landscape) or the *support* (boundary) of a data distribution. Nonstationarity can then be thought of as data that departs from the support of the distribution. Since we can judge that these "anomalous" examples will be classified poorly because they are not representative of the classifier training set, we can treat them differently (or not at all). A second technique examined uses anomaly detection with an assumption that any examples that are different are suspicious, which is an assumption that may or may not be true depending on the application.

As in our previous work, this paper does not attempt to address issues in dataset generation or feature selection. The details of the network and data collection process as well as the way in which this "raw data" is transformed into well-defined feature vectors is a very important problem, unfortunately beyond the scope of this study.

2 Dataset Description

The data is described in more detail in [1][2]. Briefly, we are using data derived from a DARPA project which set up a real network and logged normal and attack network traffic. This experiment yielded a

training set, and a *test set*. The test set was recorded after the training set, and is known to reflect somewhat different activity, which is a significant feature of our analysis. The data from this experiment were transformed into a "clean" dataset for the 1999 KDD-Cup, a competition associated with the Knowledge Discovery and Datamining conference. This dataset has 41 features for every example, with a training and test set size of approximately 500,000 and 300,000 examples, respectively. The data are labeled as attack or normal, and furthermore are labeled with an attack type that, although too fine-grained to allow experimentation, can be grouped into four broad categories of attacks: denial of service (DoS), probe, user to root (u2r), and remote to local (r2l). This is of particular interest since performance was shown previously to be very different for these categories, plausibly because they exhibit distinct nonstationarity.

We have found it useful to further segment the dataset. The training set from KDD was broken into three parts to investigate modeling on a stationary dataset: 10% was sampled for model training, 5% for model tuning (adjusting modeling parameters), and the remainder is used for validation (assessment of performance on the stationary data). Although this makes the model training set a small part of the available data it is sufficient. To reach this conclusion the performance was observed as the data size is increased, and through a large range above this dataset size no significant improvement in generalization performance was seen. Since the object of this study was the investigation of methods and approaches, rather than exhaustive parametric optimization for a final product, the trade-off between marginal performance gains and convenience (mostly in terms of SVM training times) for exploratory investigation is appropriate. The test set remains intact as a separate dataset so that the impact of nonstationarity can be explored – the test and training data are *not* drawn from a uniformly mixed dataset. Our experience has been that data nonstationarity in on-line classification systems is a significant application issue.

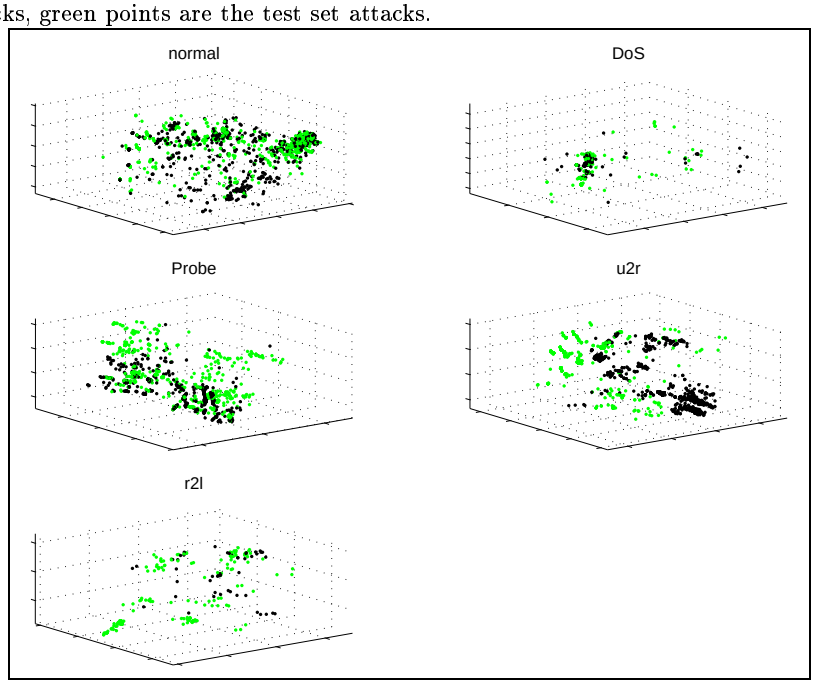
The methods chosen assume ordered numeric data. Therefore, ANOVA transformation is applied to all variables, both categorical (by individual values) and real (by intervals). Each discrete subset (value or interval) is modeled by the observed probability of attack in the training data given the category or range, for each variable

independently. This results in a transformed dataset of the same size but with a monotonic and consistently scaled metric basis. This dataspace mapping will have a significant effect on the results of the automated learning, and in previous work has shown itself to be a valuable technique in managing learning methods on large datasets.

2.1 Dataset Nonstationarity

Some data summaries will indicate the nonstationarity present. Figure 1's plots of the first three principal components show the distinction between the training set and the test set by attack type.

Fig. 1. Plots of the four attack types. In each plot Black points are the training set attacks, green points are the test set attacks.



Another view of nonstationarity can be observed through prediction performance shown in Table 1. We will use methods to draw a boundary around a dataset, and then check whether new data falls within that boundary. From the presentation in Fig. 1 we expect there to be a distinct difference in the test examples compared to the

Table 1. Test data performance comparison of the SVM-RBF and Mahalanobis outlier detection methods.

Attack type	% inlier by both	%outlier by SVM	%outlier by Mahal	%outlier by both
Normal	95.24	1.15	1.99	1.62
DoS	98.17	0.45	0.67	0.71
Probe	55.11	2.02	32.12	10.75
R2L	91.10	0.12	7.73	1.04
U2R	40.35	0.44	37.28	21.93

Table 2. Validation data performance comparison of the SVM-RBF and Mahalanobis outlier detection methods.

Attack type	% inlier by both	%outlier by SVM	%outlier by Mahal	%outlier by both
Normal	90.29	3.78	3.00	2.93
DoS	98.77	0.70	0.49	0.04
Probe	67.95	0.25	30.34	1.46
R2L	57.30	0.00	39.61	3.09
U2R	30.00	0.00	48.00	22.00

boundary generated on the training data. We used the training set, including both normals and attack examples, to derive such a boundary (within which lies 98% of the training set). Then, we examine the test set attack data, as to whether it lies within this boundary or not. We use two methods, Mahalanobis distance (MHD) and a Support Vector Machine with radial basis kernel (RBF), to construct the boundary. Table 1 shows the percent of each attack type in the test set that were called inliers[1] by both methods, that were called outliers by only one method, or that were called outliers by both methods. Table 2 is similar for the validation set (stationary with respect to the training data).

We can make two broad observations from this table. First, some attack types have apparently changed significantly in the test set. Second, the methods do not perform identically, since in some cases there are significant portions of the attack that were classified as outlier by one method, and inliers by the other.

Table 3 shows that the proportion of normals is similar between the training and test sets, but the attacks are not. This is a constructed feature of the datasets, and they are not only nonstationary

[1] we have adopted the term "inliers" to mean those points that are inside the boundary

in frequency, but also in type. This is a representative performance of the method.

Table 3. Distribution of categories in the train and test datasets.

Attack	Train(%)	Test (%)
Normal	19.69	19.48
DoS	79.24	73.90
PROBE	0.83	1.34
R2L	0.23	5.20
U2R	0.01	0.07

3 Description of Learning Methods

It is important to note that significant effort was spent investigating alternative parameter settings and learning settings, the results here show only the most successfully optimized results.

3.1 Anomaly Detection Methods

Mahalanobis Distance Let y be a $p \times 1$ random vector in the Euclidean space R^p. Assume that the mean vector of y is μ and the covariance matrix is Σ. The (squared) Mahalanobis distance from y to μ is defined to be $D^2 = (y-\mu)'\Sigma^{-1}(y-\mu)$. The Mahalanobis distance is often used to measure how far a random vector is from the center of its distribution, see [4] and [5]. μ and Σ are unknown and are estimated from data with the sample mean, $\bar{y}$ and the sample covariance matrix, $\hat{\Sigma}$, respectively.

If a future observation has a Mahalanobis distance greater than $d(99)$, the distance that yields the 99th percentile on the training data, then this new observation is considered an outlier, otherwise it is considered an inlier.

The equation $d_N = (y-\bar{y})'\hat{\Sigma}^{-1}(y-\bar{y})$ defines an ellipsoid in R^p. Geometrically, the above procedure for identifying outliers amounts to calling any point outside this ellipsoid an outlier and any point inside the ellipsoid is an inlier. This is a very constrained model compared to the flexibility of the SVM methods.

One-class Support Vector Machines Schölkopf et al.[6] adapted the to estimate the support of a distribution. A fraction, ν, of the observed data to be outliers is specified, and a "small", region, say S, in feature space that contains at least $(1-\nu) \times 100\%$ of the observed data. Any point outside of S is an outlier. In general S need not be an ellipsoid. Assuming that the training data is a random sample from an unknown distribution P, Schölkopf et al. provide a bound on the probability that a new observation drawn from P will be outside of S. Technical details that we do not address can be found in [6].

A considerable amount of effort was spent exploring the relative performance of different SVM kernel and parameter settings. Our explorations led us to a choice of the RBF kernel, considering also linear and polynomial kernels of degrees up to seven.

3.2 Method of Categorization

Since we previously explored in detail the optimized performance of alternative methods for classification [1], in this study we have chosen a single method in order to limit the number of options. The method chosen is Support Vector Machines using the radial basis function kernel[3]. An examination of the performance of this classifier is shown in Table 4. Note that this one performance point does not represent the entire spectrum of performance of the method across different detection rates. This provides indicative performance, and more detail is available in the report [1].

Table 4. The reference SVM radial basis function classifier performance.

	Overall			DoS	Probe	R2L	U2R
	error%	det%	fp%	det%	det%	det%	det%
% attacks, test data				91.8	1.7	6.5	0.09
Validation	0.07	99.94	0.11	99.99	99.06	90.02	20.00
Test	6.86	91.83	1.43	97.30	79.26	18.29	25.88

3.3 Anomaly Detection to Preprocess for Classification

Performance in a region of the dataspace well populated in the training data, i.e., the training set support, is expected to be better than

overall performance, and therefore also better than the examples outside this support. However, how to treat the performance of the anomalous examples is an open issue. Should they be considered as "normals", lowering the detection rate, or as "attacks", raising the false positive rate, should they not be considered at all, or should they be classified using a different methodology or at least a different model? The performance results documented allow the impact of various system assumptions to be assessed.

4 Results

Table 5. Predicted outliers by known class for the validation and test sets. The %O column shows the percentage of outliers represented by each category.

	RBF						MHD					
	validation			test			validation			test		
	in	out	%O	in	out	%O	in	out	%O	in	out	%O
Norm	79596	5726	68.4	58916	1677	32.7	80257	5065	59.5	58408	2185	25.1
DoS	341002	2543	30.4	227172	2681	52.3	341708	1837	21.6	226680	3173	36.5
Probe	3567	62	0.7	3634	532	10.4	2475	1154	13.6	2380	1786	20.5
R2L	942	30	0.4	16000	189	3.7	557	415	4.9	14768	1421	16.3
U2R	39	11	0.1	177	51	0.9	15	35	0.4	93	135	1.6
Total	425146	8372		305899	5130		425012	8506		302329	8700	

In Table 5 we show the results from defining a region of feature space that contains 98% of the training data. Two methods were used to define a region: support vector machines with a radial basis kernel (SVM-RBF) and Mahalanobis distance (MHD). For each type of attack we present the number of observations that are considered inliers and outliers. In addition we also show the distribution of attacks conditional on being an outlier; these are the entries in the column labeled %O. Tables 1 and 2 highlight the degree of (dis)agreement between the two methods.

The overwhelming number of examples for both the validation and test data correspond to DoS attacks: 79% for the validation data and 74% for the test data. There seems to be a bias on the part of both SVM-RBF and MHD to learn the region of feature space populated by DoS attacks. In the validation set, 68.4% of the

outliers identified by RBF are normals and for MHD nearly 60% of the outliers are normals.

The outlier selection rate of SVM-RBF on the test set is peculiar. Both SVM-RBF and MHD were trained so that approximately 2% of the observations would be beyond the support. In the test set, SVM-RBF identifies only 1.65% observations as outliers; MHD identifies 2.8% of the test data as outliers. Recall that the test data is nonstationary while the validation set is not. Not only was the distribution of attacks different from the training data, but the types of attacks were also different.

Examining performance on the test set we find that for both SVM-RBF and MHD a lower percentage of the outliers are normals. MHD is identifying a much higher percentage of probe, R2L, and U2R attacks as outliers than is SVM-RBF. In fact, these three categories is where the nonstationarity of the test data is concentrated. In table 6 for both the validation and test data we show the compo-

Table 6. Predicted class by known category for the validation and test sets, using the support vector machine supervised classifier. The %A column shows the percentage of attacks represented by each category.

	validation				test			
	Normal	Attack	% Attack	%A	Normal	Attack	% Attack	%A
Norm	85222	100	0.12	0.03	60272	321	0.53	0.14
DoS	69	343476	99.98	98.70	6961	222892	96.97	98.43
Probe	50	3579	98.62	1.03	1043	3123	74.96	1.38
R2L	155	817	84.05	0.23	16143	46	0.28	0.02
U2R	50	0	0.00	0.00	172	56	24.56	0.02
Total	85546	347972		100.00	84591	226438		100.00

sition of the predicted classes by attack type. The classifier here is the supervised SVM discriminator described in Section 3.2.

In the validation set, nearly all the DoS attacks are being classified as attacks and within the observations classified as attack, but in the test set, while the distribution of attack types is somewhat similar to the validation data, the composition of correctly predicted attacks is quite different. For example, in the validation set, nearly 99% of the probe attacks are identified as attacks but in the test data only 75% are identified as attacks. Given an observation is classified as an attack, there is a 1.03% chance that observation is a probe

attack for the validation set and a 1.38% chance if we look at the test set.

Table 7. Prediction: % classified as attack for outliers and inliers, by attack

	SVM-RBF				MHD			
	validation		test		validation		test	
	inlier	outlier	inlier	outlier	inlier	outlier	inlier	outlier
Normal	0.12	0.07	0.50	1.55	0.08	0.77	0.21	9.11
DoS	99.98	99.57	97.48	53.64	99.99	97.44	97.86	33.47
Probe	98.74	91.94	82.69	22.18	98.59	98.70	95.63	47.42
R2L	84.93	56.67	0.28	0.53	86.54	80.72	0.19	1.27
U2R	0.00	0.00	31.64	0.00	0.00	0.00	40.86	13.33

The results presented in table 7 contrast how the prediction method performs for data considered as inliers versus data identified as outliers. We compare performance on the validation and test set using both SVM-RBF and MHD to identify inliers and outliers. It is important to keep in mind that the prediction method was trained on the entire training set and not on just the observations that would be considered inliers.

The SVM-RBF method is less likely to call an example from normal, DoS, probe, and R2L an attack if it is classified as an outlier than if it is classified as an inlier. For non–attack examples in the validation data, the prediction model is less likely to call an example identified as an outlier by SVM-RBF an attack than it is if SVM-RBF calls that example an inlier. In contrast, if MHD identifies the example as an outlier the prediction model is more likely to classify that example as an attack than if it is considered an inlier. On the test set, the prediction model is more likely to call a normal example an attack if it is identified as a outlier than if it is identified as an inlier for both SVM-RBF and MHD. A non-attack example in the test set that is called an outlier by MHD is much more likely to be classified as an attack than a normal example called an outlier by SVM-RBF.

For DoS attacks in the validation set, the prediction model works about the same on inliers and outliers for both SVM-RBF and MHD; slightly fewer DoS attacks identified as outliers by MHD are classified as attacks than are DoS attacks identified as inliers. On the test set

there is a dramatic difference in performance between inliers and outliers. If SVM-RBF or MHD call a DoS attack an inlier the the prediction model classifies nearly 98% of these as attacks. However, if SVM-RBF calls a DoS an outlier, only 54% of these are classified as an attack; if MHD identifies the example as an outlier, only 34% these are predicted to be attacks.

Because SVM-RBF identifies so few probe, R2L, and U2R as outliers, as shown in Table 5 we should be cautious about any inferences we might want to make with respect to these attack types.

For probe attacks from the validation set, the prediction model is classifying approximately the same percent as attacks if MHD call the example an inlier or outlier; if SVM-RBF calls the example an outlier then it is less likely to be classified as an attack than if called an inlier. For probe attacks in the test set the prediction model is less likely to call an outlier an attack than it is an inlier, for both SVM-RBF and MHD. If MHD calls the example an outlier the model is more likely to classify it as an attack than if SVM-RBF calls the example an outlier.

For R2L attacks in the validation set, approximately inlier classification is approximately 85% correct for both methods. For examples identified as outliers by SVM-RBF, only 57% are classified correctly by the model. In contrast, of the outliers identified by MHD, the model correctly classifies about 81%. This non–parity is an interesting effect of the method, showing that anomaly detection can have very different performance. The prediction model applied to the test data works poorly with respect to R2L attacks, regardless of whether or not the example is called an inlier or an outlier.

In the validation set the prediction model incorrectly classifies all (50) of the U2R attacks as normal. In the test set, SVM-RBF identifies 51 out of 228 examples as outliers and MHD identifies 135. The prediction model correctly classifies 32% of the SVM-RBF identified inliers and none of the SVM-RBF identified outliers. For MHD inliers the model correctly classifies 41% of the inliers and 13% of the outliers.

Table 8 summarizes the performance of the overall system including anomaly detection. In this evaluation, the simpler MHD method out–performs the SVM-RBF method. As expected the inliers have better performance in both cases. In a real situation, the outliers

Table 8. Performance on test set, with different selections of the data by anomaly detection

	MHD		SVM-RBF	
	det%	fp%	det%	fp%
Overall	90.3	0.5	90.3	0.5
Inliers Only	91.9	0.2	90.9	0.5
Outliers as Normals	89.5	0.2	89.7	0.5
Outliers as Attacks	92.1	3.8	91.0	3.3

must be accounted for, and the results show what happens if we label by default all of the outliers as either attacks or normals. Labeling them as normals lowers the detection rate from the baseline (overall), with some improvement to the false positive rate (even though this is not significant for the SVM-RBF). Labeling outliers as attacks raises the detection rate, but also raises the false-positive rate significantly.

5 Discussion

The practical import of this analysis is not in terms of a finished algorithm product, since this study was on static and historical data. The primary contribution is the significance of considering network-based attack detection as distinct attack types, and the impact of anomaly detection on nonstationarity. The information presented shows clearly that different types of attacks have both very different signatures, as well as very different types of change. Also, simply the dominance in numbers of some categories will have a large effect on automated learners, as they try to minimize a criterion related to overall error minimization. The difference in performance between attack classes is important, as they each presumably have different associated misclassification costs. That these methods can treat the types differently, both as inliers and outliers, is an important consideration to the application engineer.

If these ideas were to be incorporated into a working system, the question of what to do with the outlier class arises. Choosing an arbitrary performance level on test set, the classifier along on all the data performance with a detection rate of 90.3% with a false-positive rate of 0.53%. On only the inliers the performance increases, but the outliers still need to be accounted for. Table 8 summarized these

results. Further exploration of a staged approach where inliers and outliers have different detection thresholds, or even different models altogether will be a solution to improving overall performance.

The anomaly detection segmentation increases the classification performance of inliers, as was hypothesized. The details of the performance, as discussed in Section 4, are sometimes puzzling and counterintuitive. For example, the percentage of outliers decreases from the validation to test sets for the SVM-RBF overall, and for some categories in the MHD, when natural expectation is that they would increase for a nonstationary dataset. Also, why the individual attack categories have their respective behavior with respect to nonstationarity in particular is not understood.

These algorithms are suitable for inclusion on a high speed network analysis tool, such as the programmable FPGA based *NIW Sensor* developed at Los Alamos[9]. This hardware package is capable of analyzing network traffic at gigabit speeds, and is the flip-side of this project in algorithm development.

Finally, we will comment on our experience in using these methods. SVMs with nonlinear kernels are challenging to use as a standalone tool for exploratory data analysis. Our experience has been that changes in parameters (e.g., kernel, regularization) can have significant effects in the performance of the algorithm, but yet these changes typically don't have clear causes. In an data analysis situation, it often isn't enough to simply tune for the best performance. In this case, how to tune for particular effects is not at all clear. One also wants to gain a better understanding of the data and problem. Kernel SVMs (and other nonlinear learners) are often deficient in this respect.

However, as these results show, in comparison to an intuitively understandable method such Mahalanobis distance, SVMs can be a valuable tool for gaining information regarding high-dimensional data, as well as good classification performance. If no comparative method is used, it would not be apparent whether the SVM is approximating Gaussian forms, or whether, as is the case here, the SVM is fitting a more wandering boundary. The analysis here clearly shows two things: the data is not approximately Gaussian (as is also suggested by the graphs), and the degree of flexibility in the model

of the support has a significant effect on the results both overall and by category.

We explored and used both the currently popular *libSVM* and *SVMLight* software for this work[7][8]. Currently, neither tool yields continuous values for outlier status, which, although arguably unsound, can be used for exploration of performance around the margin, and would provide an *ad-hoc* method for rank-selecting outliers.

6 Conclusion

Computer network attack detection is potentially tractable using automated learners and classifiers. Challenges remain for this methodology. One challenge is to develop an understanding of whether core attack types have a long-term signature; if not, tedious filtering data by hand to generate labeled datasets at short intervals is required. Anomaly detection methods have significant promise in this area, but they have not been demonstrated to have a performance with significant enough probability of detection at acceptable false-alarm rates.

Anomaly detection used as a method for filtering nonstationary example and ensure that classifiers operate in domains that were populated sufficiently in their training sets has been demonstrated to increase performance in this problem domain, as expected. The question remains of how to treat the outlier data robustly so that performance can be increased overall. One solution to this would be to relax the degree of discrimination of inliers, so that the training set will yield enough outliers to train an outlier-specific model. Another method could employ pure anomaly detection methods for the outliers. These are interesting directions for future work.

In this case, the SVM method did not lead to the boost in performance of the Mahalanobis distance method. There are several possible reasons for this. One is that there is perhaps not enough data to accurately assess the support of the distribution in all cases. The strong assumptions in the Mahalanobis distance measure, i.e. that the data can be represented by estimated mean and covariance, may provide required regularization. On the other hand, it is true that the SVM can be tuned to produce a more rigid classification surface, and can probably provide similar performance in this way

(although expanding the margin to include all data is costly). Another possible explanation is that the margin attention of the SVM emphasized different classes naturally, and so can provide a richer range of performance tradeoffs. The SVM method has provided insight into the data characteristics, and are an additional for data exploration and classification.

Additional areas for research suggest themselves. On-line adaptive anomaly detection is an intuitively interesting area, but whether an adaptive method can be biased with sufficient accuracy to distinguish attacks from non-attacks is an open question. Classification models of each category, with corresponding methods for distinguishing what is an inlier vs. outlier for each category seems like a compelling direction for improving performance. Studying how these machine learning methods complement rule-based systems is important in this application domain. This leads to the general topic of model ensembles: how capable families of models can be constructed, and how much performance increase can be realized.

References

1. Mike Fugate, James R. Gattiker, "Detecting Attacks in Computer Networks", Los Alamos National Laboratory Technical Report, LA-UR-02-1149.
2. Richard P. Lippmann *et al.*, "Evaluating Intrusion Detection Systems: The 1998 DARPA Off-line Intrusion Detection Evaluation", Proc of the DARPA Information Survivability Conf., vol. 2, pp. 12-26, 1999.
3. Trevor Hastie, Robert Tibshirani, Jerome Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer-Verlag, 2001.
4. Ronald Christensen (1996), *Plane Answers to Complex Questions: The Theory of Linear Models*, Second Edition. New York: Springer-Verlag.
5. Ronald Christensen (2001), *Advanced Linear Modeling*, Second Edition. New York: Springer-Verlag.
6. Bernhard Schölkopf, *et al.* (2000). "Estimating the Support of a High-Dimensional Distribution", Technical report MSR-TR-99-87, Microsoft Research, Microsoft Corporation.
7. C.Chang, C.Lin, "LIBSVM: a library for support vector machines", http://www.csie.ntu.edu.tw/ cjlin/papers/libsvm.ps.gz
8. T. Joachims, "Making large-Scale SVM Learning Practical", *Advances in Kernel Methods - Support Vector Learning*, B. Schlkopf and C. Burges and A. Smola (ed.), MIT-Press, 1999.
9. M.Gokhale,D. Dubois, A Dubois, M Boorman, "Gigabit Rate Network Intrusion Detection Technology", Los Alamos National Laboratory Technical Report, LA-UR-01-6185.