MACHINE LEARNING CLASSIFICATION OF MALICIOUS NETWORK TRAFFIC

KEITH LEHNERT AND ERIC FRIEDRICH

1. Introduction

1.1. **Intrusion Detection Systems.** In our society, information systems are everywhere. They are used by corporations to store proprietary and other sensitive data, by families to store financial and personal information, by universities to keep research data and ideas, and by governments to store defense and security information. It is very important that the information systems that house this vitally sensitive information be secure. In order for information systems to be secure, it is paramount that they utilize robust security mechanisms. Commonly found security mechanisms are passwords on accounts, encryption of sensitive data, virus protection, and intrusion detection.

An Intrusion Detection System (IDS) monitors activity at an access point and can log or prevent activities that are marked as intrusions. Intrusions occur when malicious activity gains access to or affects the usability of a computer resource.

1.2. **Industry Standards.** The main methods used by IDSs are rule creation, signature data comparison, and anomaly detection.

- With rule creation, the definitions of an unacceptable access must be predefined. These rules are tuned considerably in order to allow normal accesses through while stopping attacks. The process of manually updating these rules makes it very difficult to stay up-to-date.
- Signature data comparison uses a set of known detection data and compares it to current activity. Processes with a high matching probability are marked as intrusions.
- Anomaly detection defines a set of allowable and normal usage behavior. These patterns are then compared to current activity. Accesses that do not fall in line with the pattern definition are marked as malicious.

Many of the most popular intrusion detection methods require a large amount of support in order to be effective.

The industry standard IDS is Snort. This software uses a rule-driven architecture in order to view received packets and compare them against signature data, standard protocol, and anomaly characterization. Rules are updated on a weekly basis, and are maintained manually. Other tools such as Untangle Intrusion Protection provides thousands of built in signatures for comparison and provides updates automatically at set intervals. The company is constantly tuning settings in order to provide a more accurate system.

1.3. **Research.** Machine learning has large implications for intrusion detection, because intrusions are becoming more sophisticated and computer information systems are consistently become more complicated. By using machine learning techniques to analyze incoming network data, we can decide to block malicious attacks before they compromise an information system. Research in the field of intrusion detection seems to concentrate on a variety of support vector machine method, neural networks and cluster algorithms. These algorithms can use unsupervised learning methods to automatically update the algorithm based on new data.

1.4. **KDD Data.** One of the main challenges of intrusion detection is gathering appropriate data for training and testing of an algorithm. Although it is possible to set up a machine to collect intrusion data, ground-truthing the data could be very difficult, and our algorithm effectiveness would hinge on the accuracy of the Ground-Truth procedure. Creating a system where we provide attacks in order to allow collection of data may not be indicative of reality.

Because of this, we decided to utilize previous work done the field to allow us to concentrate more fully on our algorithm, implementation, and validation. In 1999, the Fifth International Conference on Knowledge Discovery and Data Mining was held. As part of this conference, the KDD-99 data set was created in order to facilitate a competition for building a network intrusion detector. In particular, the KDD-99 data set includes a standard set of data and a wide variety of intrusions simulated on a military network. By utilizing this data, we can have more confidence that our input truth is correct, and that the data itself is more indicative of reality.

2. Project Approach

The goal of this research is two-fold. First, we attempt to find the most effective machine learning model for identifying network attacks. Although scalability and performance are major considerations in every commericial product, our results are targeted at minimizing false positives and negatives. The approach to this work will be done in steps, with additional complexity being added to the model at each level.

As a preface to developing any models, the data must first be put into a usable format. We are using the KDDCup 99 dataset, described above, which consists of features that are either continous (numerically) valued or discrete. The discrete features in the provided dataset are in text format(i.e. tcp/udp) and must be converted to a numeric value for processing by the learning algorithm. This small amount of preprocessing is all that is required, as most of the data is already provided in a machine friendly format. The class labels provided indicate which class of attack (if any) the training example belongs to and are converted to labels of either "attack" or "safe" before use.

The Shogun [1] Machine Learning Toolbox is used in an Octave/Matlab environment for training and testing the learning algorithms. It provides several different SVM implementations along with multiple kernels. We examine two things, the relative importance of features in training the dataset and the choice of kernel algorithm. By understanding what features are most relevant, the dataset can be trimmed to include only the most useful data. Using a reduced size training increases the speed of training and classification, leading to a more efficient application.

Similarly, the choice of kernel results in different levels of errors when applied to the KDDCup dataset. Shogun offers four different kernels: Gaussian, Sigmoid, Linear, and Polynomial. Each kernel offers some parameters for tuning, for example variance of the Gaussian kernel can be specified. For the polynomial kernel, the degree of polynomial used is an argument to the SVM. This uses a kernel of the form $k(x, x') = (x * x')^d$, where d is the given degree. All kernel tests were done with the full 41 feature training set.

3. Results

3.1. **Kernel Selection.** Running several Octave simulations with the Shogun toolbox and KDD dataset demonstrated that the kernel used has a significant impact on the accuracy of the results. Specifically, the Gaussian and Sigmoid kernels performed best, classifying only 2.79% of the 10000 test examples incorrectly. In all cases the SVM was trained on a sample size of 30000 training examples. The Linear and Polynomial kernels were much less accurate. Note however, that the polynomial kernel is only of degree 3, meaning that it cannot represent a high-dimensional inner-product in the SVM calculation. This result speaks to the high error rates of the Linear kernel as well. For the KDD training data, the Linear kernel is unable to compute the correct value of the inner product $x * x'$.

Table 1. Error Rates of Shogun Kernels

Kernel	Gaussian	Sigmoid	Linear	Polynomial(d=3)
Error[%]	2.79	2.79	24.11	78.9

By varying the degree of the polynomial kernel, we were able to show an effect on the accuracy of the Support Vector Machine classifier. Namely, as the degree of the polynomial increased, so did the accuracy of the SVM. There is an anomaly with a degree of 40, where the error spikes to 99% is currently unexplained. The maximum accuary of the polynomial kernel does not exceed that of the Gaussian kernel, even in high dimensions. This suggests that the test data contains feature vectors which are not present in the original training data and cannot be correctly classified.

Table 2. Errors of Polynomials with Varying Degrees

Degree	3	10	15	20	25	30	40	50	60	70	80
Error[%]	78.9	16.7	44.39	5.0	4.89	4.80	99.1	24.84	2.79	2.79	2.79

3.2. **Feature Selection.** There are several motivating factors behind limiting the feature set of the intrusion data for the SVM. A smaller feature set may result in significantly improved training and classification timing. Depending on the intrusion detection application, timing may be critical. Additionally, some features may not truly relate to the intrusion classification results and should be excluded.

Running two-feature permutations with the shogun toolbox on the KDD dataset demonstrated which features are more likely to be useful in the SVM training.

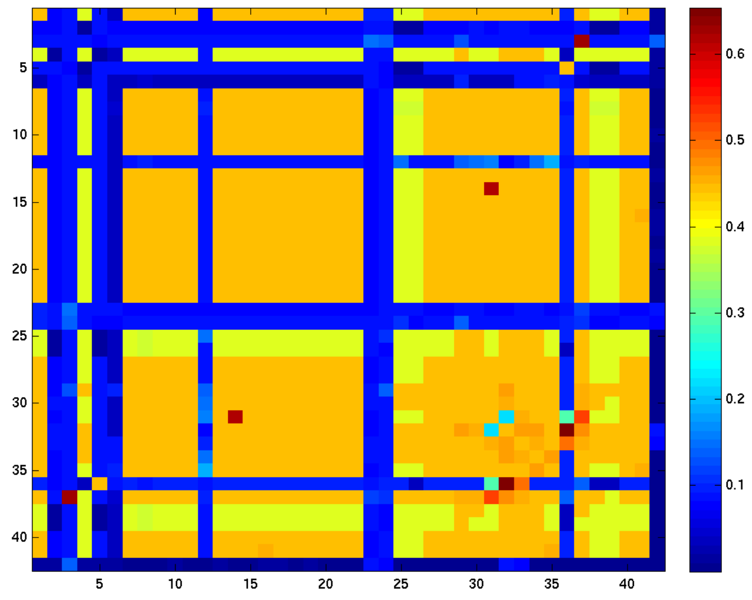

Figure 1. Errors of 2-Feature Permutations

An obvious threshold for choosing features to use is features that have errors consistently less than 20% of the time. This is a visual boundary on the output data and gives us a subset of about eight features. These features correspond to the protocol, service type, source bytes, destination bytes, logged in, count, server count, and destination host same source port rate.

4. Conclusion

Through the use of correct kernel choice and feature selection, we have shown that it is possible to improve the accuracy and efficiency of a Support Vector Machine applied to an Intrusion Detection scenario. The choice of kernel should be made to reflect the superior results provided by the Gaussian and Sigmoid kernels. Similarly, the best subset features to be trained on can be successfully identified using the parametric methods described above. By combining the kernel and feature selection, we arrive at an improved version of the algorithm. This more quickly and more accurately predicts the safety of network traffic.

References

[1] S.Sonnenburg, G.Raetsch, C.Schaefer and B.Schoelkopf, *Large Scale Multiple Kernel Learning.* Journal of Machine Learning Research,7:1531-1565, July 2006, K.Bennett and E.P.-Hernandez Editors.

[2] A.H. Sung, S. Mukkamala, *Identifying Important Features for Intrusion Detection Using Support Vector Machines and Neural Networks.* Proceedings of the 2003 Symposium on Applications and the Internet.

[3] J.T. Yao, S. Zhao, L. Fan, *Advanced Support Vector Machine Model for Intrusion Detection.* Lecture Notes in Computer Science, pg 538-543, 2006. Springer-Berlin.

[4] Pavel Laskov and Christin Schfer and Igor Kotenko, *Intrusion detection in unlabeled data with quarter-sphere support vector machines.* In Proc. DIMVA, 71–82, 2004.

[5] I. W. Tsang, J. T. Kwok, and P.-M. Cheung, "Core vector machines: Fast svm training on very large data sets," J. Mach. Learn. Res., vol. 6, pp. 363-392, 2005. [Online]. Available: http://portal.acm.org/citation.cfm?id=1046920.1058114