Detecting Network Intrusions

Naveen Krishnamurthi, Kevin Miller
Stanford University, Computer Science
{naveenk1, kmiller4}@stanford.edu

Abstract

The purpose of this project is to create a predictive model for detecting network intrusions. It makes use of a dataset, which consists of nine weeks of raw TCP dump data for a local area network (LAN) that was designed to simulate a typical U.S. Air Force LAN. We implemented a variety of different classification algorithms based on Logistic Regression and the Naïve Bayesian model to determine if individual connections were malicious. False positive and negative error rates were used as criteria to evaluate the relative performance of these approaches. We determined that a multi-dimensional Naïve Bayesian model is an effective approach for filtering network connections that minimizes impedance of normal network activity.

Introduction

Detecting malicious use of network connections is an important challenge that many security professionals face. Network attack detection is an active field of research, and a number of different approaches have been employed. Since reactive systems risk damage to the network, a pressing need exists for a predictive model that can accurately identify network attacks before they occur. With prior research in mind, we therefore attempt to develop a model that can accurately detect network attacks. We are interested in applying our model as a filter that can be incorporated into existing network security systems, which means that it must minimize misclassification of normal network connections in order to limit harms to network performance.

Data

The dataset that was used for training and testing our network attack identification model was prepared by MIT Lincoln Labs.[1] Researchers simulated a typical U.S. Air Force LAN for nine weeks, and exposed it to a number of different network attacks. The network's raw TCP dump data was then processed into a training set consisting of approximately five million connections, and a test set consisting of approximately four hundred thousand connections. Each data entry in both sets contains seven discrete features (service, protocol, log-in status, etc.) and thirty-two continuous features (duration of connection, bytes downloaded, log-in attempts, etc.). The data points also contain a label classifying each connection as either "normal" or an attack. It is important to note that eighty percent of the connections in both the training set and the test set are network attacks. In contrast, research suggests that approximately only five percent of real-word network traffic is malicious in nature.[2] This necessitates that our model have a low false-positive classification rate (rate of misclassification of benign connections). It is also important to note that the training set contains only twenty-eight different attack types, whereas the test set consists of forty-two different attack types. This is designed to test our model's efficacy at classifying attacks that have not been seen previously. Processing the data involved discretizing discrete data that was represented

[1] Data collected from http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[2] Based on a research collected by the private security firm Incapsula. Source: http://www.incapsula.com/the-incapsula-blog/item/225-what-google-doesnt-show-you-31-of-website-traffic-can-harm-your-business

symbolically, and breaking up both the training set and test set into different files that contain the discrete and continuous features.

I. Naïve-Bayseian Model Using Discrete Features

The first algorithm that was implemented was a Naïve-Bayseian classifier that incorporated a multinomial approach and Laplace smoothing. The classifier was trained on the discrete features in the training set and tested on both the test set and a 10% sample of the training set. This resulted in an error of only 6.3% on the 10% sample of the training set, and 9.36% on the test set. However a close inspection of the results revealed that the 9.36% error on the test set was composed of a 3% false-negative (missed attack) rate, and an unacceptably high 28% false-positive rate. From these results it is clear that a simple Naïve-Bayseian Classifier that relies purely on the discrete network data is not a practical model for a network connection filter in this use case, and another approach must therefore be used instead.

II. Logistic Regression

Our next approach was to design a model that could take advantage of the continuous features in the training and test data sets. Because the algorithm simply needs to determine whether or not a network connection is an attack, logistic regression seemed to be the best candidate. To train theta, logistic regression was implemented using Newton's Method for approximately 500,000 iterations. The resultant theta classified the 10% subset of the training set extremely well with only 1.22% error. Unfortunately, logistic regression had an extremely high variance as the error increased to 20% when classifying the test set. Close examination of the error also revealed that the algorithm had a false-positive rate of approximately 91%, which is completely unacceptable for our use case.

In an attempt to decrease the massive variance, we considered modifying our selection of the continuous features. Since most of the continuous features have a value of zero for each connection, it seemed likely that running logistic regression on the features that were predominantly non-zero would result in a model with decreased variance and a lower false-positive rate. The results of the new regression are shown in Figure 1 below:

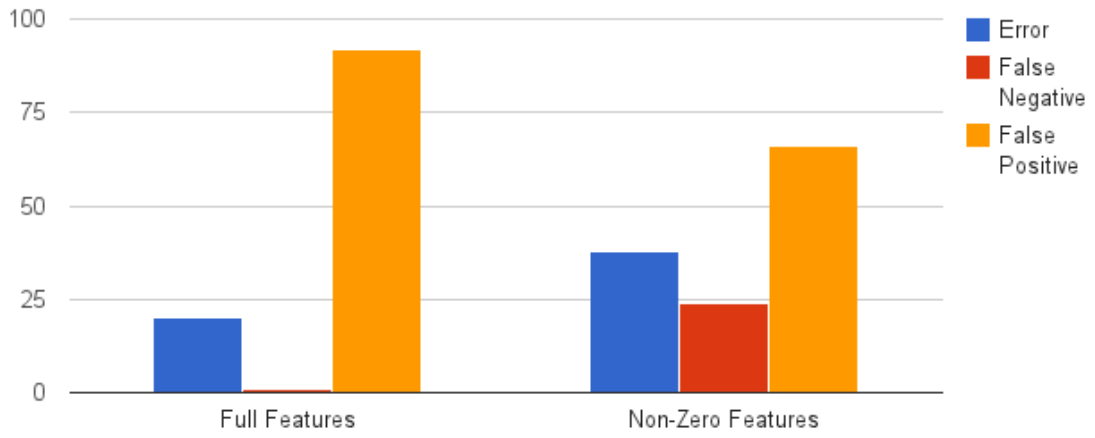

Figure 1: Difference in logistic regression error rates with refinement of feature selection

By refining our feature selection we were able to successfully decrease the false-positive rate of the logistic regression model, and only increase its error as result of Simpson's paradox. Both the error and the false-positive rate, however, were still unacceptably high and therefore logistic regression

alone appeared to be insufficient for our use case.

III. Naïve-Bayseian Model Using All Features

Since logistic regression had such poor results, a decision was made to try incorporating the continuous features into the Naïve Bayes Model. To do this, the continuous features of each file were discretized by assigning each of them a discrete value based on their continuous value, and then combined with the discrete features of each file for use by the Naïve-Bayesian Classifier. The combination approach did decrease the training error by a significant amount, but had no visible impact on the testing error. As a result, it seemed unlikely that the addition of discretized continuous data would result in decreased bias, and would instead only increase the variance of the model. We therefore decided to try and come up with another approach that would involve modifying our Naïve Bayes algorithm to achieve better results.

IV. Multi-Dimensional Naïve-Bayseian Model Using Discrete Features

The final algorithm that we examined was developed by taking a closer look at the implementation of our Naïve-Bayseian Classifier. The training stage of the Naïve-Bayseian Classifier returns a vector of probabilities in which the ith entry represent the probability that a connection is an attack given the presence of the ith feature. In this dataset, however, an increased value of a feature does not necessarily mean an increased chance of attack. For example, a connection with a service of "aol" (which is represented using the discrete value 54) does not necessarily have a higher chance of being an attack than a connection with service "http" (which is represented using the discrete value 16). Using this intuition, a modified version of the Naïve-Bayesian Model was implemented using an mxn array A of probabilities where m is the number of features (7), n is the max value taken on by any of the features (72) and the value at A^{ji} is the probability that a connection is an attack given that the ith feature has value j. Using this Multi-Dimensional Naïve-Bayseian Classifier (MDNBC) resulted in an overall error of 8.68%. A comparison of its overall error rate against the error rates our other approaches is provided in Figure 2 below.

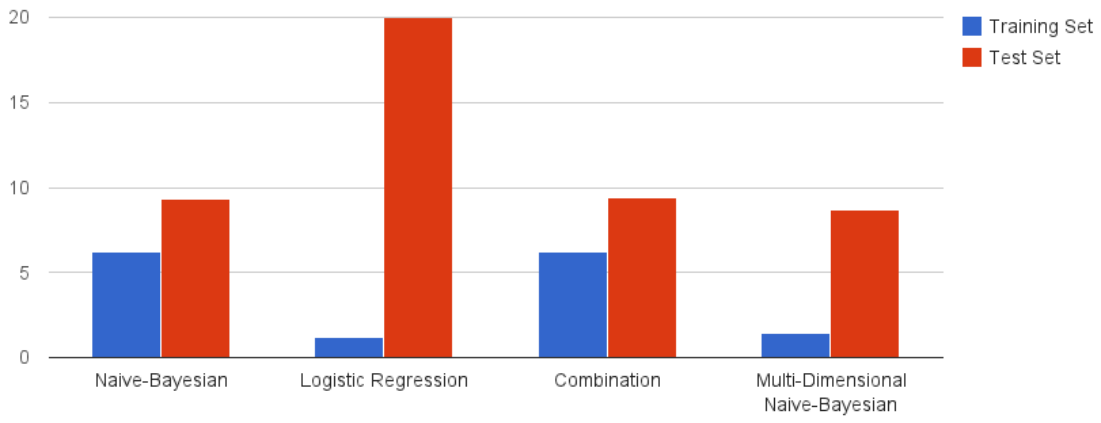

Figure 2: Comparison of overall error rates of different approaches

The MDNBC therefore outperforms our other approaches based on overall error. A closer examination of the MDNBC's false positive rate of 0.5% also shows significant improvement in comparison to other approaches. A summary of the false positive and negative rates of our approaches on the test set is provided in Figure 3 below:

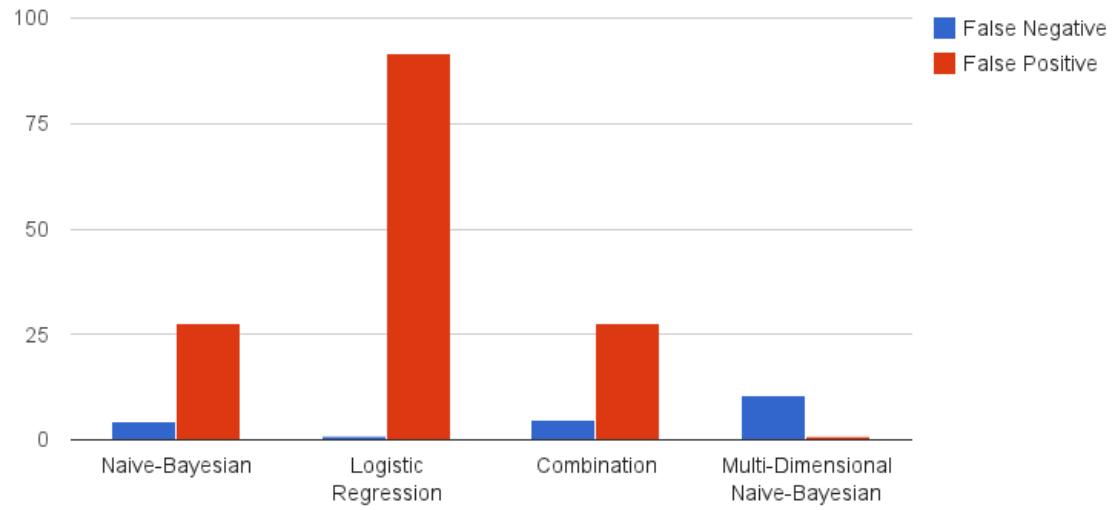

Figure 3: Comparison of False Positive and False Negative Rates of Different Approaches

It is important to acknowledge that the Simple Naïve Bayes Classifier's false negative rate of 10.65% does outperform the MDNBC's false negative rate of 3%, and Logistic Regression's false negative-rate of 0.9%. The significant impacts on network performance that accompany using these models as filters suggest that the MDNBC model is the ideal approach that can be integrated into existing network security defense systems. This is especially true when results are projected based on real world traffic, where only 5% of connections are malicious and 95% are benign. A comparison of the projected error rates based on real world traffic is provided in Figure 4 below:

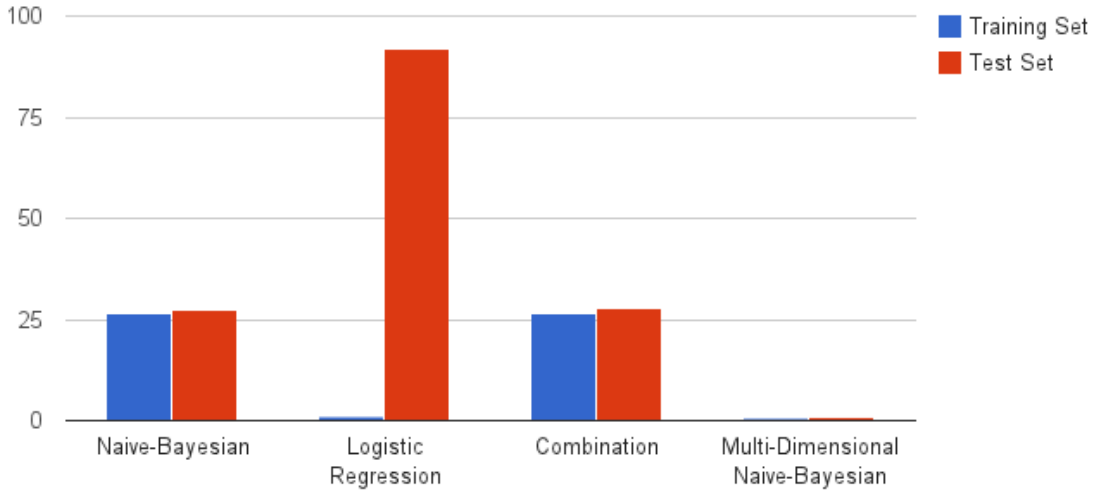

Figure 4: Comparison of Projected Error Rates Based on Real World Traffic

Conclusion

The MDNBC model therefore clearly outperforms our other models, given that it has a projected error rate of approximately 1% based on real world traffic. It is also important to note that the MDNBC's .5% false-positive rate is much lower than the current state of the art algorithm used to

classify this dataset, which uses a Hidden Naïve-Bayesian model.[3] The state of the art algorithm does have a slightly lower overall error for this data set, with an error rate of only 7.23%, as opposed to the MDNBC's rate of 8.68%. However its higher false-positive rate suggests that it will be a less practical filter for real-world network traffic conditions compared to the MDNBC in most use cases.

Areas of Further Research

There are three approaches that we plan to continue researching in order to improve our results and develop an even more accurate model for network connection filtering. The first would be to make another attempt at logistic regression using a feature-selection algorithm to determine which features to use. This algorithm would possibly select features that could accurately train our theta to work well on the test set as well as the training set. Another major source of error is our use of the Naïve Bayesian assumptions, which is by no means true in this context, since the features are not necessarily independent from each other. Dropping these assumptions and/or using a weighted Naïve Bayes approach would exponentially increase the time to build a classification model, to the point of impossibility for this project. However, such a model could be implemented in the future with enough time and computational resources. Finally, the third major improvement would be to obtain a data set that was refined to show a sequence of events (i.e. first a connection was established, then x bytes were downloaded, then a log-in attempt was made, etc.) rather than simply listing features. Data in this form could be used in a Markov Model based algorithm that could probably classify attacks more accurately.

[3] Levent Koc, Thomas A. Mazzuchi, Shahram Sarkani, A network intrusion detection system based on a Hidden Naïve Bayes multiclass classifier, Expert Systems with Applications, Volume 39, Issue 18, 15 December 2012, Pages 13492-13500, ISSN 0957-4174, http://dx.doi.org/10.1016/j.eswa.2012.07.009.
(http://www.sciencedirect.com/science/article/pii/S0957417412008640)