Machine Learning for Network Intrusion Detection

Final Report for CS 229, Fall 2014

Martina Troesch (mtroesch@stanford.edu) and Ian Walsh (iwalsh@stanford.edu)

Abstract

Cyber security is an important and growing area of data mining and machine learning applications. We address the problem of distinguishing benign network traffic from malicious network-based attacks. Given a labeled dataset of some 5M network connection traces, we have implemented both supervised (Decision Trees, Random Forests) and unsupervised (Local Outlier Factor) learning algorithms to solve the binary classification problem of whether a given connection is normal or abnormal (malicious). Our results for LOF are mixed and hard to interpret, but with Decision Trees we are able to achieve prediction accuracies of over 90% for both normal and abnormal connections. Posterior analysis of the best-performing trees gives us new insight into the relative importance of different features for attack classification and suggests future avenues to explore.

1 Background

As networked systems become more and more pervasive and businesses continue to move more and more of their sensitive data online, the number and sophistication of cyber attacks and network security breaches has risen dramatically [5]. As FBI Director James Comey stated earlier this year, "There are two kinds of big companies in the United States. There are those who've been hacked... and those who don't yet know they've been hacked." [6] In order to secure their infrastructure and protect sensitive assets, organizations are increasingly relying on network intrusion detection systems (NIDS) to automatically monitor their network traffic and report suspicious or anomalous behavior.

Historically, most NIDS operate in one of two styles: misuse detection and anomaly detection. Misuse detection searches for precise signatures of known malicious behavior, while anomaly detection tries to build a model for what constitutes "normal" network traffic patterns and then flag deviations from those patterns. For all the same reasons that signature-based antivirus software is becoming obsolete (the ease of spoofing signatures and the increasing diversity and sophistication of new attacks), misuse-detection is struggling to remain relevant in today's threat landscape. Anomaly-based intrusion detection offers the enticing prospect of being able to detect novel attacks even before they've been studied and characterized by security analysts, as well as being able to detect variations on existing attack methods. In our project we focus on classifying anomalies using both supervised and unsupervised learning techniques.

2 Data and Features

Our dataset comes from The Third International Knowledge Discovery and Data Mining Tools Competition (KDD Cup '99) [4]. The goal of the competition was to build a model capable of classifying network connections as either normal or anomalous, exactly the task we want to accomplish. The KDD Cup '99 data is itself a subset of an original Intrusion Detection Evaluation dataset compiled by MIT's Lincoln Laboratory at the behest of DARPA in 1998 [11]. The data consists of simulated network connection traces representing a variety of network-based attacks against a background of normal network activity over a seven-week period at a medium-sized Air Force base, and there is a smaller two-week section of test data with identical features. For the KDD Cup '99 subset, there are about 5M total network connections' worth of training data, and a test set of about 319K connections.

The data consist of network connections captured by a UNIX tcpdump-like utility and analyzed by a tool similar to tcptrace [8]. Each connection is described by 41 features and is labeled as either "normal" network traffic or with specific type of network-based attack. Some of the features include duration, protocol type, the number of source bytes, and the number of destination bytes. The full list of features can be found at [4]. The types of attacks present in the dataset can also be found at [4] and include malicious activities such as buffer overflow and smurf attacks. Rather than try to predict the exact type of attack, which may be very difficult if not impossible to do accurately from a single connection, we focus on the somewhat easier binary classification problem of simply labeling connections as "normal" or "abnormal".

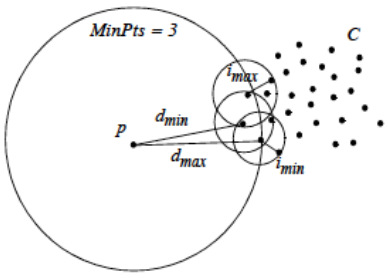
Obtaining public datasets for network intrusion detection is very difficult, for both privacy reasons and the costly, error-prone task of hand-labeling connections (and given FBI Director Comey's warning, the accuracy of such labels must be suspect!). As one of the few available public datasets in this area, the KDD Cup '99 data has been widely studied and cited by the intrusion detection community [12]. While there has been criticism raised against the dataset, and it is no longer an accurate representation of network activity in most environments, it still serves a valuable role as a benchmark for training and comparison of new detection algorithms, as well as a minimal "sanity check" that any new scheme must pass to be considered credible. [1]

3 Approaches

We have implemented three different machine learning algorithms to classify the KDD Cup '99 data, with the goal of optimizing their performance and comparing their strengths and weaknesses. Local outlier factors (LOF) is an unsupervised learning method that assigns every data point a numeric score representing its likelihood of being an outlier, based on its local density of nearby points relative to its neighbors. Decision trees are a supervised approach that models the decision boundary as leaves of a binary tree, with the interior nodes representing different features of the input connections ordered by their relative importance for classification. Random forests are a variation of decision trees wherein instead of training a single tree on the full feature set, we train an ensemble of smaller trees each on a random subset of the features, and aggregate the predictions of each small tree into our final prediction for a sample. We discuss the details of each in turn.

3.1 Local Outlier Factors

The Local Outlier Factor (LOF) algorithm is an unsupervised algorithm developed by [3] that assigns, to every data point, a numeric score representing its likelihood of being an outlier. Higher scores correspond to a greater "outlier factor". The intuition behind LOF is that points whose 'local density' is much less than the local density of their nearest neighbors are highly likely to be outliers. The following figure from [3] illustrates this idea by showing the density around point p compared to the density around its nearest neighbors.



Concretely, the LOF score of every example i is calculated using the equation

$$LOF(i) = \frac{\sum_{n \in N_k(i)} \frac{lrd(n)}{lrd(i)}}{|N_k(i)|} \tag{1}$$

$N_k(i)$ is the set of closest examples, or neighbors, that are within $kDist(i)$, where $kDist(i)$ is the distance of the k^{th} nearest example to i. Thus $|N_k(i)| = k$, usually, but it may be greater than k if multiple points are tied as the k^{th} nearest neighbor to i.

In (1), the quantity $lrd(i)$ is known as the *local reachability density* of point i, and it represents the density of points in a local region around i. It is given mathematically by

$$lrd(i) = \left(\frac{|N_k(i)|}{\sum_{n \in N_k(i)} reachD_k(i,n)} \right) \tag{2}$$

where the *reachability distance* is defined as $reachD_k(i,n) = \max\{kDist(n), dist(i,n)\}$.
The scores generated by LOF tend to 1.0 for points that are clearly not outliers, and scores higher than 1.0 indicate an increasing likelihood of being an outlier.

LOF has three parameters that need to be chosen by the user: the value of k, which controls how many neighbors we consider "local" to a point, the distance metric for comparing points, and the threshold score for declaring a point as either "normal" or "abnormal".

3.2 Binary Decision Trees

Decision Trees (DTs) are a supervised learning algorithm that can learn complex decision boundaries for handling both classification and regression problems. The algorithm works by constructing a tree from the training data in which interior nodes correspond to one of the input features and the leaf nodes contain a prediction of the output value or category. Each interior node also contains a cutoff value, and in a binary DT like we have implemented, a left and a right subtree. To make a prediction using a DT, we walk down the tree from the root with our feature vector, branching left at each node if our feature is <= the cutoff value at the node, and right otherwise, until we reach a leaf.

In constructing our DT, we intuitively want to put those features most strongly correlated with the classification near the *top* of the tree, so that we make our most important decisions first. Non-informative features should be placed lower, near the leaves, or as an optimization they may be removed entirely. We quantify these notions through the concept of *information gain* between the feature and the label, which is formulated in terms of the related quantities *entropy* and *conditional entropy*:

Entropy:

$$H(X) = -\sum_{j=1}^{m} p_j \log p_j \tag{3}$$

Conditional Entropy:

$$H(Y|X) = \sum_j P(x = v_j) H(Y|X = v_j) \tag{4}$$

Information Gain:

$$IG(Y|X) = H(Y) - H(Y|X) \tag{5}$$

The quantity $IG(Y|X)$ represents how much knowing X tells us about the value of Y. With these definitions, constructing the tree is relatively straightforward: beginning at the root, for every interior node we

select the feature having the greatest information gain with the class label, and we choose the cutoff value as that value having the highest specific conditional entropy for that feature. Once a feature has been selected for a node, it cannot be selected again for any of its children. The left and right subtrees are trained recursively on the appropriate subset of the training data, based on the feature and cutoff selected. The process stops when all features have been used or when the number of training samples in a node becomes too small. In our implementation, we stop building the tree when there are $|S| <= 10$ training samples at a node, or when all samples at a node are identical. In predicting the label for a leaf, we use the following heuristic: if *any* of the training examples that made it to the leaf are abnormal, we predict "abnormal", else we predict "normal". We made this decision to increase the number of malicious connections we correctly classified (fewer false negatives) at the expense of misclassifying some benign connections as abnormal (more false positives).

3.3 Random Forests

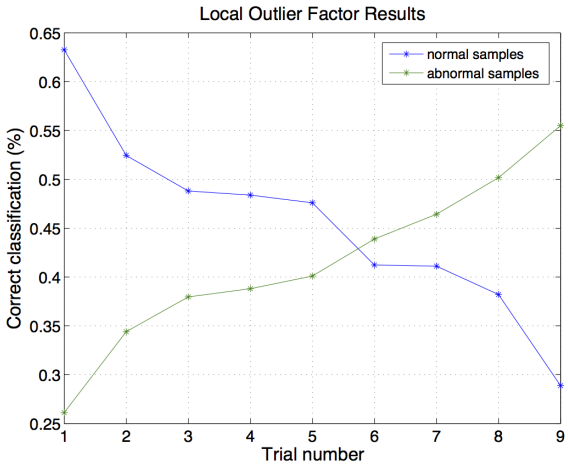
In practice, single decision trees often tend to overfit the training data, and may generalize poorly as a consequence [10]. *Random forests* is a technique for reducing DT test error due to overfitting. Instead of training a single DT on the entire set of features, we instead train an ensemble of n trees, each considering only a random subset of m of the features. In making a prediction for a test example, we generate the n predictions from each tree and we output our final classification as the mode of these predictions.

Our implementation uses ensembles of $n = 20$ trees, each trained on a random subset of $m = 11$ features. We chose $m = 11$ because, after expanding symbolic features in binary sub-features, each connection contains 122 distinct features, and there is evidence in the literature that $m \approx \sqrt{|F|}$ is usually a good choice. We had to alter our heuristic for leaf predictions from the single DT case: for forests, we choose the prediction for each leaf to be the *mode* of all training samples it saw. Without this change, our random forest were classifying *every* sample they saw as "abnormal", so a more conservative prediction policy was necessary.

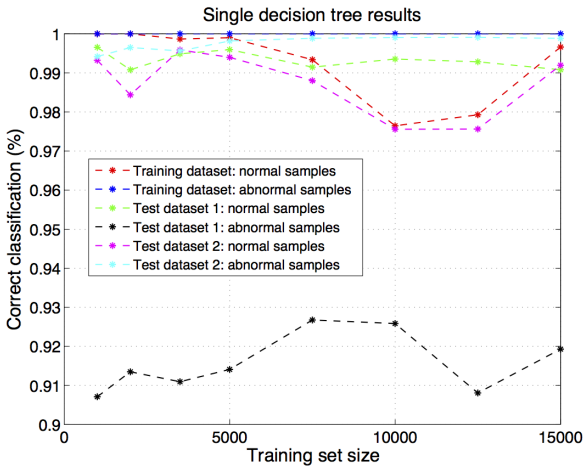
4 Experimental Results

The results for the *LOF* algorithm are shown in the following plot. The y axis shows the percent of the examples that were classified correctly. The x axis shows the trial number, where a trial number uses a particular percentage of the examples as the k value and corresponding threshold value. The threshold values were chosen for each k value using a method proposed by [7] where the *LOF* score of five thousand random "normal" samples was calculated, and then the threshold was chosen such that 2% of the samples would be classified as "abnormal" if they were being tested. The distance metric used for comparing feature vectors was chosen to be the cosine distance after some preliminary experiments.

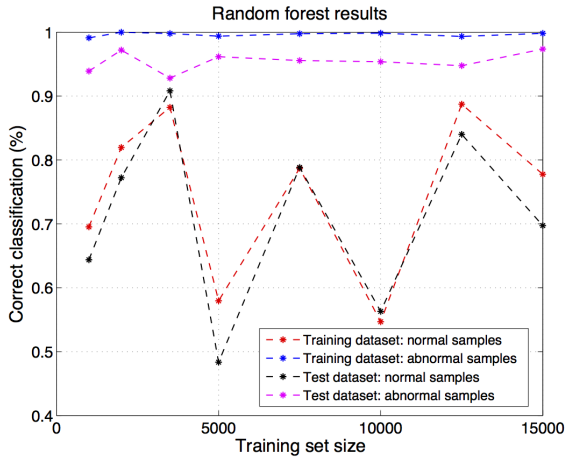
trial #	1	2	3	4	5
% k	10	15	20	25	30
threshold	1.65	1.92	2.46	2.87	3.00
trial #	6	7	8	9	
% k	35	40	45	50	
threshold	3.16	3.39	3.50	3.85	



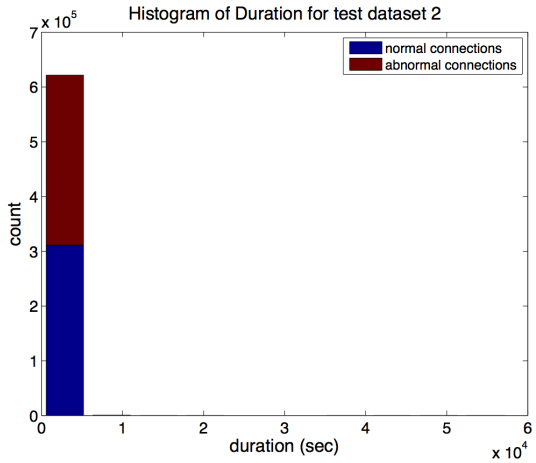
Classification accuracy as a function of training set size for single DTs is shown below. Test dataset 1 contained dfnsdlf samples, and test dataset 2 contained 311029 samples.



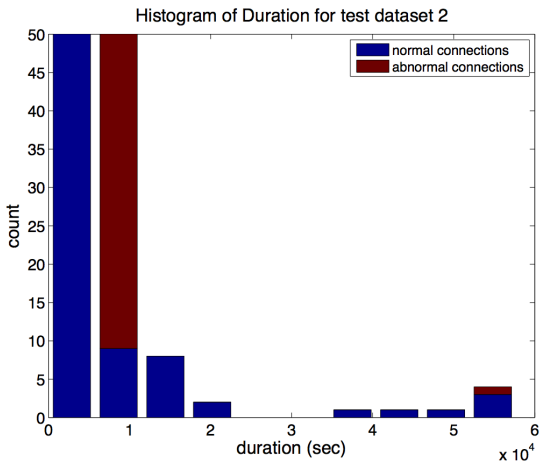
Classification accuracy as a function of training set size for random forests of 20 DTs, each trained on 11 features can be seen in the following plot. Only test dataset 2 was used.



Distribution of values for the "duration" feature in test dataset 2 is shown in the below histogram.



A zoomed-in view of the same distribution for test dataset 2 can be seen in the next plot.



5 Discussion

Our LOF results were poor: we were not able to achieve better than chance accuracy for both normal and abnormal connections simultaneously. We have two theories of why this was the case. First, the entire concept of anomaly-based intrusion detection is predicated on the assumption that intrusions, or malicious connections, are relatively *rare* or *anomalous*. Interestingly, we found this was not the case in the KDD '99 dataset: both the training and test data contain 80% "abnormal" connections! This means that LOF is fighting an uphill battle when trying to identify abnormal connections as outliers. Secondly, LOF is very sensitive to the distribution and size of the dataset relative to the parameter k. Too small a k, and points won't look very far when calculating their local densities: even small clusters of outliers may be self-reinforcing. Too large a k, on the other hand, and the local neighborhood becomes too big; points are compared against essentially the entire dataset, so everything looks abnormal.

Random forests performed reasonably well at classifying abnormal connections, but did worse than we expected at recognizing normal connections. We hypothesize two reasons for the poor performance: first, we implemented them as a safeguard against overfitting the training data, but our single DT results show no evidence of overfitting! If the test and training data are mostly homogeneous, much of the benefit of random forests is lost. Secondly, our single-DT analysis (below) indicates that very few features were significantly indicative of the final classification: most features were just noise. By restricting each tree in the forest to a random subset of the features, then, we were actually *hurting* our predictive power by weakening the influence of the best features.

Somewhat surprisingly, single decision trees were the best-performing algorithm we studied: they were able to achieve over 90% classification accuracy for every training sample size and test dataset we attempted, and near-perfect classification of abnormal connections. We'd expected that single trees would show evidence of overfitting by having higher testing errors; the fact that they did not suggests that the test data does not vary significantly from the training data.

An important advantage of DTs is that the decision boundary they produce is framed in terms of the *original feature set*, so their results are relatively natural to interpret. We took advantage of this by dissecting some of our best-performing DTs to see what insights we might glean about the dataset. We found that the "best" features, those most strongly predictive of the classification, were connection duration and the protocol/service (udp, icmp, smtp, etc.), while the "worst" predictive features were the time-averaged features for a destination host (number of connections in the last 2s, for example).

Armed with this knowledge, we investigated the distribution of these features in test dataset 2 (the distributions for the "duration" feature are presented above). We found that duration, for example, showed a very

clear correlation with classification: abnormal connections tended to be relatively short-lived, while virtually all lengthy connections represented benign network traffic, and the other top features showed similar associations. With this knowledge, we can begin to understand why our decision trees performed so well on this data: they excel at finding the most informative features to split on, and our analysis indicates that these same features were the most predictive in both the training and test data.

6 Conclusions & Future Work

While our single DT results are encouraging, the accuracy of our random forest and LOF algorithms could be improved. For random forests, we can think of two likely enhancements: first, we should aggressively remove training features that are shown to have little predictive power, either through their information gain scores or some other feature selection algorithm. This would reduce the probability of some tree being trained with only poor features. Secondly, instead of outputting the *mode* of the 20 trees as our final prediction, we could assign weights to each tree based on their performance on the training data, and take a weighted average of their predictions as our final result. This technique, known as "boosted" random forests, has proven to be extremely powerful in many applications.

In the case of LOF, as with random forests, we might be able to improve performance by restricting our analysis to fewer, more important features, so that the distances between normal and abnormal feature vectors would increase. Secondly, while the authors in [7] were able to achieve 74% classification accuracy, they were using certain time-based features not present in the KDD data itself. If, like those authors, we went back to the raw 1998 DARPA dataset and used *tcptrace* or a similar utility, we could extract these same features and more, which could indicate which subsets of data we should apply LOF scoring to for best results.

We note that our approaches are complementary: the importance of features learned in training single DTs can be fed back in to inform feature selection in random forests and LOF, and these models can generalize better than DTs to different distributions (i.e., new network attacks) once trained.

In conclusion, we have implemented three machine learning algorithms for detecting anomalies in network connection data. We have analyzed their performance and found that single decision trees give the best performance for this application, with surprisingly good classification accuracy. In addition, they can expose valuable properties of the underlying data, which may inform future analysis. Our results convince us that while network intrusion detection is a very hard problem, machine learning algorithms can help and will have an important role to play in future NID systems.

References

[1] *Testing intrusion detection systems: A critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory*, ACM Trans. Inf. Syst. Secur. **3** (2000), no. 4, 262–294.

[2] Dhruba Kumar Bhattacharyya and Jugal Kumar Kalita, *Network anomaly detection: A machine learning perspective*, CRC Press, 2013.

[3] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander, *Lof: identifying density-based local outliers*, ACM Sigmod Record, vol. 29, ACM, 2000, pp. 93–104.

[4] KDD Cup 1999 Data, *Kdd cup 1999 data*, `http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`, Web. Nov 2014.

[5] IBM. IBM Security Services 2014 Cyber Security Intelligence Index, *Ibm statistics on data breach epidemic*, `http://www-935.ibm.com/services/us/en/it-services/security-services/data-breach/`, April 2014, Web. 16, Nov 2014.

[6] CBSNews. CBS Interactive, *Fbi director james comey on threat of isis, cybercrime*, `http://www.cbsnews.com/news/fbi-director-james-comey-on-threat-of-isis-cybercrime/`, October 2014, Web. 16, Nov 2014.

[7] Aleksandar Lazarevic, Levent Ertöz, Vipin Kumar, Aysel Ozgur, and Jaideep Srivastava, *A comparative study of anomaly detection schemes in network intrusion detection.*, SDM, SIAM, 2003, pp. 25–36.

[8] Shawn Ostermann, *Tcptrace-official homepage*, `http://www.tcptrace.org/`, Web. Nov 2014.

[9] Robin Sommer and Vern Paxson, *Outside the closed world: On using machine learning for network intrusion detection*, Security and Privacy (SP), 2010 IEEE Symposium on, IEEE, 2010, pp. 305–316.

[10] Vladimir Svetnik, Andy Liaw, Christopher Tong, J Christopher Culberson, Robert P Sheridan, and Bradley P Feuston, *Random forest: a classification and regression tool for compound classification and qsar modeling*, Journal of chemical information and computer sciences **43** (2003), no. 6, 1947–1958.

[11] MIT Lincoln Laboratory: Communication Systems, Cyber Security: Cyber Systems, and Technology, *Darpa intrusion detection evaluation*, `http://www.ll.mit.edu/mission/communications/cyber/CSTcorpora/ideval/data/1998data.html`, Web. Nov 2014.

[12] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali-A Ghorbani, *A detailed analysis of the kdd cup 99 data set*, Proceedings of the Second IEEE Symposium on Computational Intelligence for Security and Defence Applications 2009, 2009.