Intrusion Detection based on Network Traffic Data

CSCI 5622-Machine Learning Project

May 5, 2017

Byron Becker

Kevin Holligan

Saim Khan

Victoria Slattum

Ishita Srivastava

University of Colorado Boulder

# **INTRODUCTION**

# The rise in popularity of the internet and the advent of the Internet of Things has increased the risk of network attacks. Intrusion detection is an important component of network security and is used for the purpose of safeguarding the internal network against outside attacks. Traditional intrusion detection mechanisms like firewalls and access control systems are not able to detect and prevent increasingly sophisticated network attacks. Using machine learning techniques for the purpose of intrusion detection can result in more adaptable solutions with higher accuracy and fewer false positives.

In this project, we used a C4.5 decision tree classifier to detect various kind of intrusions on network data.

# **RELATED WORKS**

Over the last 10 years, there has been a flurry of research into machine learning algorithms’ effectiveness at network traffic classification. This was due to a combination of static network classification techniques, which became outdated due to more dynamic IP addressing and the continued expansion of internet users and internet growth. We surveyed a number of papers and the various techniques employed.

Many of the papers examine the effectiveness between varying machine learning techniques, comparing Naïve Bayes, bayes net, neural networks, C4.5, and others [4,5]. Singh, et. al., found that bayes net, combined with a correlation based feature selection algorithm, was the highest performing classifier. However, this classifier struggled with long training times, but was still sufficient for near real-time classification [5]. Limthong, et. al., meanwhile found that k-nearest neighbors had the best performing accuracy, but were comparing it against a more limited set of algorithms, namely the Naïve Bayes model [4]. In both cases, the researchers collected their own data off a set of controlled ‘clean’ machines or through open access packet sniffing programs. We expand on these works by taking a deeper dive into the C4.5 algorithm and by using a dataset that contains a high volume of ‘attack’ data as opposed to a heavy percentage of normal traffic.

Li, et. al., and Casas, et. al., took an approach focusing on decision tree classifiers, distributed machine learning algorithms, and random forest classifiers as compared to traditional intrusion detection techniques, such as part of application based [2, 3].

# **DATASET**

The dataset we are using for this project was prepared by MIT Lincoln Labs by simulating a typical U.S. Air Force local area network. The dataset was generated by setting up a representative U.S. Air Force LAN, then peppering it with attacks for nine weeks. The dataset itself contains millions of tuples and each tuple is labeled as either normal or a kind of intrusion (attack). The four major types of attack categories are:

1. DOS (denial of service)
2. R2L (unauthorized access from a remote machine)
3. U2R (unauthorized access to root)
4. Probing (Surveillance)

Each record has a total of 41 features. The features are of the following three types:

1. Basic features of individual TCP connections
2. Content features suggested by domain knowledge
3. Traffic features computed using a two second time window

For more information on the dataset refer to <http://kdd.ics.uci.edu/databases/kddcup99/task.html>

The training data contains 494,021 training examples, and the test dataset contains 311,029 data points.

The training dataset contains data that result in 23 different classifications of attacks, whereas the test dataset contains 38 different classifications of attacks. Looking more into the data, we can see that 98.2% of the training data has one of the three classifications: ‘smurf’, ‘neptune’, and ‘normal’; and 90.9% of the test data falls into these classifications. ‘Normal’ naturally represents ordinary, safe traffic data, whereas ‘smurf’ and ‘neptune’ are both forms of denial of service attacks. In both the test and train datasets, the ‘normal’ data represents between 19-20% of the data, while the other 79-80% of the data represents some form of attack. Unlike regular network traffic, this dataset has a much higher volume of network attacks. This imbalance in the data may introduce biases in the model, which could result in a model that has problems generalizing to unseen data (since normal network traffic is not nearly as malicious).

# **DATA PREPROCESSING**

# The following processing steps were taken to filter the data for the classifier:

# REMOVING DUPLICATES

After examining the data, we found multiple cases of recurring training examples in the dataset. This resulted in two primary problems, namely: it made the dataset too large to analyze efficiently, and secondly, it created a bias towards more frequently occuring records. Therefore, as a first step to preprocessing, all the duplicate records were removed.

* 1. REMOVING FEATURES BASED ON VARIANCE

# When working with a dataset with a robust number of features (41 in this instance), it is essential to remove features that carry no predictive value. Often times, these features are the ones with nearly identical values across all classification types. To identify the features with this quality, we used Scikit-learn’s feature selection module. The feature selection module removes all features with a variance below a certain threshold. In other words, if all the values of a particular feature are within the threshold of each other, that feature will be removed. For our dataset, we removed features with variance less than a threshold of 0.01.

* 1. NORMALIZATION OF FEATURES

Machine learning algorithms operate with higher accuracy if the data is normalized. This prevents certain features from dominating the others because of their scale. As such, we normalized the feature values to ensure that all features were weighted proportionally in their representation. This was implemented using scikit-learn’s preprocessing module. According to scikit documentation, the normalize method of this module scales individual samples to have a unit norm in a quick and easy way. The input to this method is a single array like dataset and performs the above operation using the L1 or L2 norms.

# **DATA STRUCTURING FOR CLASSIFIER**

The features in our dataset could be broadly divided into the following two categories:

1. CONTINUOUS FEATURES: Those features whose values can be any possible numerical value and are not restricted by set bounds or a discrete set.
2. CATEGORICAL FEATURES: Those features that can take only a particular set of discrete values.

|  |  |
| --- | --- |
|  |  |

The dataset has a combination of both types of features that make it challenging to feed to the classifier as the classifier will be unable to handle continuous and discrete features simultaneously. In order to use a regressor for continuous features and a classifier for categorical features, the features were divided into these two separate categories. The classifier was then applied on the categorical features and a regressor was applied to the continuous features. The results were then pipelined to give the final output.

# **CHOICE OF ALGORITHM: C4.5 DECISION TREE CLASSIFIER**

* 1. ALGORITHM EXPLANATION

C4.5 is a supervised learning algorithm based on Ross Quinlan’s ID3 (Iterative Dichotomizer 3) algorithm for generating decision trees from a dataset. It requires a set of feature values with classification labels as input. The algorithm then learns from these training examples and predicts an output value for any unseen input object. C4.5 improves upon ID3 in that it can be used for both continuous and discrete data, it can handle training data with missing feature values, and it prunes the trees after they are constructed.

The C4.5 classifier follows the Occam’s Razor philosophy of choosing the simpler of the two correct solutions offered by the decision tree.

The algorithm works as follows:

1. Check for the following base cases:
   1. All training examples are part of one class.
      1. C4.5 creates a leaf node telling the tree to choose that particular class.
   2. Encounter an instance of a previously unseen class.
      1. C4.5 adds a decision node higher up the tree using its expected value.
   3. There is no data in the training set
      1. C4.5 returns a “failure” tree leaf
2. Calculate information gain (difference in entropy) for splitting on each of the features and determine which feature maximizes information gain. Entropy and information gain are computed as follows:
3. **Entropy:  **
4. **Gain:  **

where:

E(S) – information entropy of S

G(S,A) – gain of S after a split on feature A

n – number of classes in S

Pr(Ci) – ratio of class Ciin S

m – number of possible values feature A can take

Pr(Ai) – ratio of cases that have feature Ai  in S

E(SAi) –subset of S that has feature Ai [1].

1. Create a decision node that splits on the feature that maximizes the information gain.
2. Partition the dataset examples into subsets based on the split, then do steps 1-3 on the new partition, adding these nodes as children of the decision node.
   1. WHY THIS ALGORITHM WORKS

Upon reviewing the literature, we found that the C4.5 algorithm is one of the most popular methods for classification of network traffic [3]. The algorithm’s popularity is likely due to its many positive attributes. For example, a C4.5 model is easy to implement and interpret, capable of incorporating both continuous and categorical data, and is good at mitigating noise in a dataset.

The following are specific benefits of using C4.5 Decision Trees for prediction and classification:

1. Feature selection and variable screening is performed implicitly within the algorithm. After fitting the C4.5 model to a training set, the top few nodes of the decision tree will be the most significant.

2.   Minimal data processing is required because the decision tree classifier does not require any type of normalizing or scaling of data.

3. Classification performance is unaffected by nonlinear relationships between parameters.

4.  The classifier is easy to explain and interpret due to its intuitive and highly organized tree-like structure.

# **RESULTS AND ANALYSIS**

* 1. BASELINE

The baseline classifier we used for the data was a Naïve Bayes (NB) classifier. We used this model as it is a simple and efficient classifier for data. NB classifiers are subcategorized into Gaussian, Multinomial, and Bernoulli depending on the type of data you are working with. The Gaussian NB model is used for continuous data, while the Multinomial model can be used for categorical data, and the Bernoulli model can be used for boolean data. As our dataset contains both categorical and continuous data, we used a combination of Gaussian and Multinomial NB classifiers. To do this, we parsed the data into discrete and continuous subsets, formatted the respective subsets and ran the discrete feature data through Scikit-learn’s CountVectorizer to obtain counts that we could pass to the MultinomialNB model. We then took all of the class assignment probabilities from both models and fit a new Gaussian model on this training data. We then repeated the process with the test data on the trained Gaussian model to the following results.

This method of calculating the NB probabilities independently, then combining them and calculating an output works because of the fundamental assumption in Naïve Bayes that the features are all independent of each other. Network data classification involves considering many different features, many of which are dependent on other circumstances being present or not present, which makes the Naïve Bayes model a good baseline model for our data. With our combined feature set, we were able to obtain a training accuracy of 94.7% and a test accuracy of 77.3% as our baseline.

* 1. **C4.5 IMPLEMENTATION**

We parsed the data into discrete and continuous subsets, formatted the respective subsets, and ran the discrete feature data through Scikit-learn’s Decision Tree classifier and continuous features through Scikit-learn’s Decision Tree regressor. We then used Scikit-learn’s pipeline module to fit the results of both the classifier and regressor, and repeated this process with the test data to yield the following results:

**Train accuracy: 0.97740 – 98.7%**

**Test accuracy: 0.96363 – 96.3%**

The C4.5 decision tree classifier performed extremely well with preprocessing applied to the data.

# **CONCLUSION**

Using the C4.5 decision tree, we were able to significantly improve on our baseline Naive Bayes model: improving the test accuracy from 77.3% to 96.3%. This is a surprising result, exceeding our initial expectations for the decision tree classifier. With these results, we feel that a decision tree is an appropriate classifier for classifying incoming traffic and properly vetting potential attacks. For future work, we would consider converting our model into an online version of the classifier and analyzing the results. This would make the classifier more flexible for real world applications which use streaming data as opposed to retroactive analysis. We would also look into deployment methodologies, especially with the influx of embedded devices as part of the movement towards an Internet of Things.

# **DISTRIBUTION OF WORK**

BYRON BECKER

* Preliminary research
* Baseline model and Report
* Editing final report
* Video

KEVIN HOLLIGAN

* Preliminary research
* Data parsing and segregation
* Baseline model
* Editing and written style for write-up
* Project write-up
* Video

SAIM KHAN & ISHITA SRIVASTAVA

* Preliminary research
* Initial Data collection using Wireshark to intercept network traffic and analysis of whether the data is relevant for the task.
* Finalizing the dataset.
* Data Preprocessing.
* Project Write-up
* Reviewing and editing of all the deliverables

VICTORIA SLATTUM

* Preliminary research
* C4.5 model
* Editing write-up
* Slides for video
* Video

# **REFERENCES**

[1] Anon. 2011. Decision Trees – C4.5. (November 2011). Retrieved May 5, 2017 from <https://octaviansima.wordpress.com/2011/03/25/decision-trees-c4-5/>

[2] P. Casas, P. Fiadino, and A. D’Alconzo. Machine-Learning Based Approaches for Anomaly Detection and Classification in Cellular Networks. *Network Traffic Measurement and Analysis Conference 2016*.

[3] W. Li and A.W. Moore. 2007. A Machine Learning Approach for Efficient Traffic Classification. *2007 15th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems* (2007). DOI:<http://dx.doi.org/10.1109/mascots.2007.2>

[4] K. Limthong and T. Tawsook. 2012. Network traffic anomaly detection using machine learning approaches. *2012 IEEE Network Operations and Management Symposium* (2012). DOI:<http://dx.doi.org/10.1109/noms.2012.6211951>

[5] K. Singh, S. Agrawal, and B.S. Sohi. 2013. A Near Real-time IP Traffic Classification Using Machine Learning. *International Journal of Intelligent Systems and Applications* 5, 3 (March 2013), 83–93. DOI:http://dx.doi.org/10.5815/ijisa.2013.03.09