Machine Learning in Network Intrusion Detection

Liru Long, Xilei Wang and Xiaoxi Zhu

I. Introduction

Network security is of great importance to individuals and organizations. Advanced technologies have been developed to protect both incoming and outgoing traffic, e.g. encryption of sensitive information, firewalls to block risky traffic. However, traditional firewalls and Intrusion Detection System (IDS) identify and block suspicious traffic based on pre-configured rules, traffic signatures as well as patterns of known attacks stored in its database. Given the exponential growth in the size and complexity of network, this legacy approach turns out to be inefficient. On one hand, the increasing amount of attack patterns requires large database and long processing time. On the other hand, numerous unknown attack patterns pop up everyday, which makes it next to impossible to keep the firewall/IDS up-to-date.

In response of the challenges, an advanced firewall and/or IDS should possess the following characteristics. Firstly, it should be able to detect network attacks efficiently. Secondly, it should be able to assess unknown traffic patterns. Both academia and industry have been developing IDS implemented with Machine Learning algorithm. A properly trained IDS learns the general pattern of normal against malicious traffic, thus able to process incoming traffic much faster than those iterating through static rules. In addition, the pattern generalization also allows IDS to make a better judgement on unforeseen traffic patterns.

This project aims to build an effective IDS using Machine Learning algorithms. The pre-processing engine in IDS extracts relevant features (such as transport protocol, application service, connection duration, payload size, etc.) from each network connection. The core engine uses the extracted features as learning input and outputs the binary classification result, i.e., normal VS attack.

The rest of the paper is organized as follows: Section 2 summarizes a few related works. Section 3 discusses the data set and input features in more details. Section 4 establishes the performance baseline using different single-stage classifier. Section 5 proposes an extension with more sophisticated multi-stage classification algorithms. Section 6 focuses on optimization with feature reduction. Section 7 concludes the project with a comparison of various implementations and proposes directions of future improvement.

Liru Long lirul@stanford.edu), Xilei Wang (xileiw@stanford.edu) and Xiaoxi Zhu (zxx313@stanford.edu) are with the Department of Electrical Engineering, Stanford University, Palo Alto CA94305

II. Related Work

Given the fact that data in production network is hard to obtain, the KDDCup'99 [1] data set has been widely studied and applied in related field of network security.

As clean data set is the first step in developing good machine learning algorithms, study in [2] focuses on techniques of raw data processing and normalization. Sabhnani proposes an effective feature reduction scheme that results in significant improvement of the classification results.

Siddiqui et. al. [3] analyzed the dataset with K-means clustering, targeting to characterize the correlation between various types of attacks and the transport layer protocol.

Researchers have also attempted to develop more advanced learning algorithm for IDS. The algorithm proposed in [4] optimizes SVM with GA techniques so that the most relavent feature sets and optimal parameters of SVM could be identified quickly. Chandrasekhar et. al. [5] proposes a cascaded technique using K-means clustering, Fuzzy-neural networks and SVM.

III. Data Set and Features

Training and test data sets used in this project are downloaded from KDDCup99 website - an annual Data Mining and Knowledge Discovery competition organized by ACM Special Interest Group. Raw TCP dump data in a simulated Local Area Network (LAN) are processed to generate network connection records, each constitutes of 41 feature fields and a class field. The full training set is generated from seven weeks of network traffic, which results in five million records. The full test set, which is generated from two weeks of traffic, leads to two million connection records. A 10% subset of training and test data was used in this project. The package includes a training set of 494,021 samples and a test set of 311,029 samples.

Each data sample consists of 41 features, which can be further categorized as basic features, content features, time-based features. The network connection features of downloaded data set are of inconsistent format. Some features are thus converted (from text to numerical value) and/or normalized to the same order. Conversion approaches are listed as below:

- Text to numerical: feature fields labeled with text message (e.g. protocol, service, etc) are converted to integer values between 0 to 10.
- Medium numerical values: feature fields where the range is of medium magnitude ~10^2 (e.g. num_file_creations) are normalized to the range [0,10] w.r.t. its max value

- Large numerical values: feature fields where the numerical range is large ~10^5 (e.g. dst_bytes) are converted to the range [0,10] using base-10 logarithm.

Each training sample is labeled with a text indicating either normal or attack, with an exact description for type of attack. The specific attack types can be further categorized into four generic categories. The label is thus converged to integer values with the correspondence map – **[normal:0, Probing:1, DoS:2, U2R:3, R2L:4]**.

IV. Single-stage Classifier

In this section, three single-stage classifiers [1] - Naive Bayes, Decision Tree, and K-means clustering - are trained and tested. Hold-out cross validation is used to select model parameters where applicable. Multi-class classification is implemented for each model. However, the result is only evaluated based on the confusion matrix of normal(positive) VS attack(negative) defined in Table I. In other words, misclassification across different types of attacks are ignored as it does not raise any concern in real application.

True Positive Rate (TPR)	$\frac{True normal predicted as normal}{True Normal}$
False Positive Rate (FPR)	$\frac{True attack predicted as normal}{True Attack}$
False Negative Rate (FNR)	$\frac{True normal predicted as attack}{True Normal}$
True Negative Rate (TNR)	$\frac{True attack predicted as attack}{True attack}$

TABLE I: Confusion Matrix

A. Naive Bayes

Classification with Naive Bayers algorithm is simple and efficient, yet gives satisfactory results for most classification problems. Therefore, it is implemented as the baseline for other more advanced algorithms.

A trained Naive Bayes classifier uses Maximum A Posterior estimation to predict the class of test data P(Y) with known features $x_1, x_2, ..., x_n$ by maximizing the conditional probability of $P(y|x_1, x_2, ..., x_n)$. Bayes Theorem defines that:

$$P(y|x_1, x_2, ..., x_n) = \frac{P(y)P(x_1, x_2, ..., x_n|y)}{P(x_1, x_2, ..., x_n)} \quad (1)$$

Under the assumption that all features are independent, Eq. 1 can be further reduced to:

$$P(y|x_1, x_2, ..., x_n) = \frac{P(y)\prod P(x_i|y)}{P(x_1, x_2, ..., x_n)} \quad (2)$$

$$P(y|x_1, x_2, ..., x_n) \propto P(y)\prod P(x_i|y) \quad (3)$$

Therefore, the MAP can be expressed as:

$$\hat{y} = \arg\max_y P(y)\prod P(x_i|y) \quad (4)$$

In addition, it is assumed that each feature follows a Gaussian distribution as in Eq. 5 where the parameters μ_y and σ_y are estimated using maximum likelihood.

[1]All machine learning algorithms are implemented with scikit-learn library downloaded from http://scikit-learn.org/stable/

$$P(x_i|y) = \frac{1}{\sqrt{2\pi}\sigma_y} exp\Big(-\frac{(x_i-\mu_i)^2}{2\sigma_y^2}\Big) \quad (5)$$

The confusion matrix of classification result using the above multi-class Gaussian Naive Bayes classifier is presented in Table II

TPR = 0.9410	FPR = 0.0901
FNR = 0.0590	TNR = 0.9099

TABLE II: Confusion Matrix for Naive Bayes

B. Decision Tree

Decision Tree is another supervised learning algorithm commonly used for classification. The classification process can be represented as a tree structure, where root node and each intermediate node contains an attribute classification criteria. A given test sample starts from the root node and traverses through a selected path to a leaf node based on the decision made at each node. Each leaf node is labeled during training process and the test sample is assigned the same label as the leaf node.

The classification criterion at each node is formulated during training process by maximizing information Gain. $IG(Y|X)$ for a given node with attribute X is defined as Entropy difference before and after partition [7].

$$IG(Y|X) = H(Y) - H(Y|X) \quad (6)$$

$H(Y)$ is Entropy before partition, which measures the homogeneity of a given set of data as follows:

$$H(Y) = -\sum_i p_i log p_i \quad (7)$$

$H(Y|X)$ is the weighted sum of Entropy for all subsets after partition based on Attribute X, defined as below:

$$H(Y|X) = -\sum_i P(X = x_i)H(Y|X = x_i) \quad (8)$$

Thus each node partitions the data set using the most distinguishable attribute by maximizing the Information Gain.

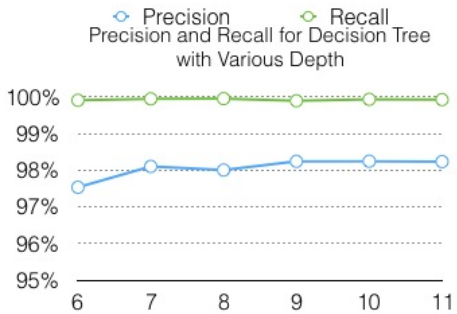

Fig. 1: Decision Tree Performance VS Tree Depth

Decision Tree has several advantages over Naive Bayes. First, it selects only one attribute as classification criteria at each node. Thus the process prioritizes the feature set,

which achieves similar outcome as a weighted cost function. Second, by limiting the tree depth, it filters out attributes that are less indicative, hence prevent overfitting and reducing unwanted noise of high-dimensional feature set.

The key parameter in Decision Tree algorithm is the maximum tree depth. Test runs are conducted to select the optimal value. Initial runs are conducted over the depths range of 5 to 40 with a step of 5. Second iteration of runs uses finer step, with the depth ranging from 6 to 11.

Fig. 1 presents the trend of precision and recall for normal class with varying tree depth. A Decision Tree classifier with maximum depth of 9 is selected based on the results, which gives the optimal combination of precision and recall.

The confusion matrix of classification results using the trained Decision Tree is given in Table III below.

TPR = 0.9878	FPR = 0.0918
FNR = 0.0122	TFR = 0.9082

TABLE III: Confusion Matrix for Decision Tree

C. K-means clustering

K-means clustering, as indicated by its name, groups given data set into a fixed number of clusters according to their geometric locations in the space spanned by the features. The algorithm follows the steps below:

1. K cluster centroids are initialized (typically randomly)
2. Assign each sample to the a cluster whose centroid is closest to the sample using Eq. 9

$$c^{(i)} := \arg\min_i ||x^{(i)} - \mu_j||^2 \tag{9}$$

where $x^{(i)}$ is the geometric location of the i^{th} sample, μ_j is the geometric location of the j^{th} centroid and $c^{(i)}$ is the assigned centroid.

3. Reset the centroids of each cluster to the center of all samples in the same cluster using Eq. 10

$$\mu_j := \frac{\sum_{i=1}^{m} 1\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^{m} 1\{c^{(i)} = j\}} \tag{10}$$

Classification of test data is simply achieved by selecting the nearest centroids based on Euclidean distance. It is noted that generated clusters are not labeled, while the test data needs to be classified as either normal or attack. Therefore, an additional step is added, where each centroid is assigned to one of class 0 to 4 based on majority vote of all samples in this cluster. Test data is then labeled the same class as its nearest centroid.

Viewing the data set in a high-dimensional space, they should form five clusters in the ideal situation. However, considering the fact that each class can be further divided into different sub-classes, it is likely to have many more clusters. Even worse, two clusters belonging to the same class may not be located next to each other, indicating that the data set is not linearly separable. To address this issue, number of cluster centroids needs to be selected wisely. As a common practice, number of centroids is set to a large number in order to deal with data skew as well as non-linear separable data set. However, an arbitrarily large K value may result in over-fitting.

Test runs are conducted with K value varying from 100 to 1000 to identify the best performing classifier. Each model, with a given K value, is also trained multiple times as the centroids can be different due to randomly assigned initial value. Fig. 2 shows the performance comparison of different models based on precision and recall computed from cross validation result.

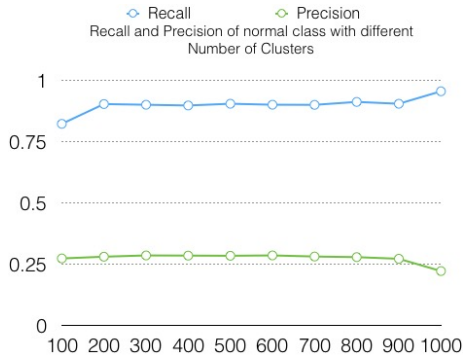

Fig. 2: K-means Performance VS Number of Clusters

The 400-cluster model is selected since it generates most satisfactory results in cross validation. It is also observed that the performance of this model is most stable across multiple runs. However, no cluster of class 3 (R2L attack) can be formed even for such a large K value.

The confusion matrix resulting from a modified 400-cluster K-means algorithm is given in Table IV

TPR = 0.9837	FPR = 0.1005
FNR = 0.0163	TFR = 0.8995

TABLE IV: Confusion Matrix for K-means

D. Discussion

The confusion matrices for all three single stage classifiers show similar results. It is noted that although True Positive Rate(TPR) is high (>90%), which minimize the occurrence of false alarm. However the False Positive Rate(FPR) is relatively high. which implies potential security risks arising from undetected attacks.

A possible root cause for the inferior performance is that some attack traffic have distinctive feature signature compared to normal class while others may have similar feature patterns. A simple classifier trained with the complete set is thus biased towards the most distinguishable features. In the next section, multi-stage classifier models are proposed to overcome this limitation.

V. Multi-stage Classifier

The three single-stage classifiers give satisfactory recall value for normal class. However, the relatively low precision value indicates a large number of undetected attacks. One of the possible reasons is that those misclassified traffic

has similar feature signature compared to normal class. The proposed solution is to implement multiple classifier that performs finer classification. Two different approaches are discussed in this section. Random forest constructs parallel classifiers while the Decision Tree with GMM model builds a cascaded classifier model.

A. Random Forest

Random Forest, in its simplest form, is a collection of Decision Trees. Each decision tree is trained separately with only a subset of training samples. As a result, trees are constructed with different classification criteria at each level. The randomness helps to reduce the variance. A given test sample is thus classified using all different trees and the label is assigned based on majority vote [8].

There is a natural trade-off in determining the number of decision trees in random forest algorithm. Increased number of trees may improve the classification performance, however resulting in longer runtime. Test runs are conducted with number of DTs varying from 10 to 15. The performance difference is not significant across different runs. Therefore, the number of trees is limited to 11, which generates more stable results across multiple test runs.

The confusion matrix of classification result using the Random Forest algorithm is given in Table V.

TPR = 0.9949	FPR = 0.0929
FNR = 0.0051	TFR = 0.9071

TABLE V: Confusion Matrix for Random Forest

B. Decision Tree with GMM

The major performance concern is precision value of normal class. Thus a cascaded classifier is proposed where the second stage classifier only acts upon the normal class labeled by first classifier. The second stage classifier is thus trained to identify finer difference between normal and attack class. Decision Tree is selected as the first classifier as it gives the best performance out of the three algorithms in Section 3. The second stage classifier is implemented with Gaussian Mixture Model (GMM).

GMM is a probabilistic model with the underlying assumption that all data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. Test sample is assigned to the class that it most likely belongs to. It is clear that GMM is a unsupervised learning algorithm. Unsupervised learning is preferred in the second stage as it is misclassification mostly come from subcategories that are not present in training set. Similar as K-means clustering, GMM is able to handle non-linear boundary in the data sets.

GMM implements four sub-models, each corresponding to a different covariance matrix. As with other learning algorithms, test runs are conducted to select optimal parameter setting. Tied covariance matrix generates best result in cross-validation. The confusion matrix of classification result with the cascaded classifier is presented in Table VI.

TPR = 0.9336	FPR = 0.0873
FNR = 0.0664	TFR = 0.9127

TABLE VI: Confusion Matrix for Cascaded

C. Discussion

It is observed that both multi-stage classifiers gives marginal performance improvement. The results trigger more in-depth analysis into the data set. It is identified that most mis-classification arises from class 3 (U2R attack) and class 4 (R2L attack). A detailed study reveals the following insights.

- Class 3 attack has extremely small sample size in both training set and test set compared to other class. Thus, the unique feature signature/pattern is not properly learned by the classifiers.
- In addition, training set and test set does not follow similar distribution across different classes. Only 0.23% of training samples are of Class 4 type. However, there are 5.2% of them in the test set. The uneven distribution results in bias of trained classifier, thus poor performance in identifying this shadowed class.
- Lastly, class 4 can be further split into different sub-classes of attacks. Examining the original feature data shows that class 4 samples in training set and class 4 samples in test set belong to different sub-classes. It is highly likely that those sub-classes have different feature signature/patterns. Thus the trained classifier is not able to identify the unknown pattern in test set.

In conclusion, a cleaner training set should be re-generated in order to build classifier with high performance.

VI. Feature Reduction

The optimal depth for best-performing Decision Tree based classifier is 9, indicating that there are redundant or unimportant features out of the 41-dimension feature space. Therefore, feature reduction is exercised as a further optimization for the IDS with machine learning algorithm.

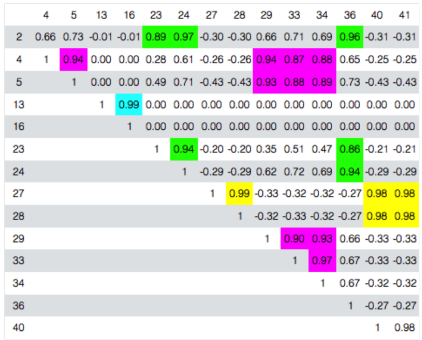

Fig. 3: Partial Correlation Matrix

The approach for feature reduction is highlighted below:

1. Generate correlation matrix for complete feature set
2. Identify feature groups with high pair-wise correlation
3. Apply Principal Component Analysis to reduce the dimension of each feature groups

$$r_{ij} = \frac{\sum(y_{ki} - \mu_i)(y_{kj} - \mu_j)}{\sqrt{\sum(y_{ki} - \mu_i)\sum(y_{kj} - \mu_j)}} \tag{11}$$

A partial correlation matrix is given in Fig. 3, where the correlation is computed using widely adopted Pearson r coefficient in Eq. 11. Highlighted fields are selected groups of features with high correlation, given as follows:

- Group1: 13 & 16
- Group2: 2 & 23 & 24 & 26
- Group3: 27 & 28 & 40 & 41
- Group4: 4 & 5 & 29 & 33 & 34

Each group is then reduced to its principal component with orthogonal linear transformation defined in Eq. 12 . The feature space dimension is reduced from 41 to 30.

$$\mathbf{w} = \underset{||\mathbf{w}||=1}{\arg\max} \sum (x_{(i)} * \mathbf{w})^2 \tag{12}$$

The base-line single-stage classifiers are then re-trained with the new feature set. However, the performance improvement is marginal. Detailed performance comparison for all models in this paper is presented in the conclusion.

A further analysis on feature correlation is conducted to further investigate this counter-intuitive result. In the analysis, feature reduction is applied to individual classes of traffic. Re-applying steps 1) & 2) on a single class of samples, the result is summarized in Table VII.

Class	Correlated Features
Probe	2 & 5 & 32 & 34 & 35 27 & 28 & 40 & 41 23 & 29 & 30
DoS	4 & 5 & 25 & 26 & 38 & 39 27 & 28 & 40 & 41 2 & 24 & 29 & 33 & 34 & 36
U2R	27 & 28 & 40 & 41
R2L	27 & 28 & 40 & 41 32 & 34 & 36 13 & 16, 26 & 26, 29 & 30

TABLE VII: Feature Correlation for Individual Class

It is shown that each class has different correlated feature groups. On one hand, it is a strong proof of the assumption that each class has a unique feature signature/pattern. On the other hand, it reveals that the generic feature reduction approach is too aggressive. It does help to compress redundant features. However, it blends some of the distinctive features from different classes as well. Therefore, a one-vs-all classifier with class-specific feature reduction may give better results. Due to time constraints, this approach is not implemented. The concluding remarks highlights this approach as a future extension of this project.

VII. Conclusion and Future Work

Fig. 4 below presents a complete view of different machine learning algorithms tested for a network Intrusion Detection System. The key performance metrics used for evaluation is precision (true positive against predicted positive) and recall (true positive against condition positive) as target is to classify network traffic into normal against attack.

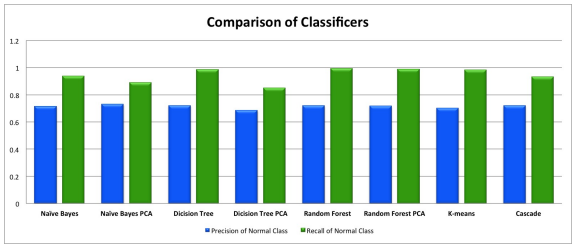

Fig. 4: Comparison of Classifier

The results show consistent performance across different approaches. Recall value ranges from 90% to 99%. The high percentage indicates a low rate of false alarm. However, the precision value varies in between 70% to 75%, leading to potential risks of undetected attacks.

Further analysis reveals that the root cause lies in the data sets. First of all, the raw features are of different format, the pre-processing and normalization step may shadow some unique patterns. Secondly, five classes are not evenly distributed in the training set. Data skew leads to bias in trained classifier and thus degraded accuracy while classifying test data. Thirdly, training data and test data in the same class does not necessarily possess similar feature signature/pattern as each class consists of multiple sub-classes. Lastly, features are correlated but the correlation pattern varies across different classes. Hence PCA across entire training set does not improve the performance.

In view of the above characteristics inherent to the data set, future work should focus on developing customized data processing techniques prior to implementing more sophisticated learning algorithms. A few prospective directions are briefly discussed here.

1. Analyze the value distribution of individual features to select more suitable normalization function
2. Re-generate training data set with a even distribution across different classes
3. Finer categorization of the general attack class
4. Conduct class-specific feature reduction, followed by one-vs-all classification with the reduced feature set

REFERENCES

[1] KDD Cup 1999 Data, http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, Web. Oct 2015.

[2] M. Sabhnani and G. Serpen, 'Application of Machine Learning Algorithms to KDD Intrusion Detection Dataset within Misuse Detection Context', *In Proceedings of the International Conference on Machine Learning: Models, Technologies, and Applications*, pp. 209-215, 2003.

[3] M. Siddiqui and S. Naahid, Analysis of KDD CUP 99 Dataset using Clustering based Data Mining,*IJDTA*, vol. 6, no. 5, pp. 23-34, 2013.

[4] D. S. Kim, H.-N. Nguyen, and J. S. Park, "Genetic algorithm to improve SVM based network intrusion detection system," in *Advanced Information Networking and Applications.* AINA. 19th International Conference, pp. 155-158, 2005.

[5] A. M. Chandrasekhar and K. Raghuveer, Intrusion detection technique by using k-means, fuzzy neural network and svm classifiers, in *Proceedings of IEEE International Conference on Networking, Sensing and Control*, 2013, pp. 1-7.

[6] Scikit-learn.org, 'scikit-learn: machine learning in Python - scikit-learn 0.17 documentation', 2015. [Online]. Available: http://scikit-learn.org/stable/. [Accessed: Nov-2015].

[7] Quinlan J. Induction of Decision Trees. *MACH LEARN*. 1986;1:81106.

[8] V. Svetnik, A. Liaw, C. Tong, J. Culberson, R. Sheridan and B. Feuston, 'Random Forest: A Classification and Regression Tool for Compound Classification and QSAR Modeling', *Journal of Chemical Information and Modeling*, vol. 43, no. 6, pp. 1947-1958, 2003.