Semi-supervised Random Forest for Intrusion Detection Network

Ningxin Shi, Xiaohong Yuan, William Nick
North Carolina Agricultural and Technical State University
ningxss@hotmail.com, xhyuan@ncat.edu, wmnick@aggies.ncat.edu

Abstract

In order to protect valuable computer systems, network data needs to be analyzed and classified so that possible network intrusions can be detected. Machine learning techniques have been used to classify network data. For supervised machine learning methods, they can achieve high accuracy at classifying network data as normal or malicious, but they require the availability of fully labeled data. Semi-supervised machine learning methods, however, can use a small number of labeled examples and train a large number of examples without label.

In this research, we explore the use of semi-supervised Random Forest in classifying network data and intrusion detection. It was used to classify the Third International Knowledge Discovery and Data Mining Tools Competition dataset (KDD 1999) and the result were compared with the results of using the supervised methods of Random Forest. The results were also compared with those using ladder network, an approach which combines unsupervised neural networks, in classifying KDD 1999.

KEYWORDS

Random Forest, semi-supervised machine learning, KDD 1999.

1 INTRODUCTION

In the last decade, the development of computer technology and the invention of the internet have brought many benefits to society. Some of these benefits include faster communication, better convenience and more productivity. Today, more than three billion people have access to the Internet, equating to 42 percent of the global population in 2014. Thus, network security has become very important. Intrusion Detection System (IDS) is a device or software application that monitors a network or system for malicious activity or policy violation. Current IDSs are either signature based or anomaly based. The signature based IDS detects the attacks which have been documented. Thus new types of attacks cannot be detected by the signature based detection. The anomaly based detection focuses on finding unknown or unusual activity patterns in the observed data. It can detect new attacks but it also needs a domain expert to distinguish normal or abnormal activity patterns.

In this research, machine learning methods has been used to detect attacks. Some of the machine learning classifiers have been developed under supervised learning. Supervised learning is to use labeled data to train the classifier and then test the classifier in the test set. The disadvantage of supervised learning is that it needs a large amount of labeled data, which is not practical since it's labor intensive. Different from supervised learning, the unsupervised learning is to use unlabeled data to train the classifier and to infer a function described hidden structure. However unsupervised learning is very sensitive to noise data and the results can be influenced a lot. Hasan and his team developed the Support Vector Machine (SVM) and Random Forest supervised learning methods[4]. Salama and his team have used a hybrid intrusion detection scheme utilizing both SVM and Deep Belief Networks (DBN) [5].

Semi-supervised learning falls between supervised learning and unsupervised learning. Different from supervised learning, semi-supervised learning uses a small amount of labeled data and a large amount of unlabeled data to train and test the classifiers. Researchers have found that unlabeled data, when used in conjunction with a small amount of labeled data, can produce considerable improvement in learning accuracy. There are two kinds of semi-supervised learning, transductive and inductive learning. Transductive learning is to use labeled data to train the classifier and use the unlabeled data to test the classifier. The inductive learning is to use both labeled and unlabeled data to train the classifier. Zehra Cataltepe *et al* had a research on semi-supervised decision trees for network intrusion detection based on KDD99 dataset [6]. Leistner *et al* developed a semi-supervised algorithm of Random Forest [3] based on Breiman's research on Random Forest which developed the Out-of-bag theory [1]. Liu *et al* discussed about the node splitting for Random Forest Construction and developed the new algorithm [2].

This research implemented semi-supervised Random Forest for network intrusion detection. KDD99 dataset was used for training the classifier. The experiments used the same number of classes, and the same number of samples in each class as in Mtahir *et al* 's research [7] in order to compare the results with supervised Random Forest and semi-supervised ladder nework.

The paper is organized as follows. Section 2 discusses the background of Random Forest based on Dr. Leo Breiman's research [1]. Section 3 discusses about the experiment results and compares to results on supervised Random Forest and semi-supervised ladder network [7]. Section 4 concludes the paper.

2 Background

2.1 Random Forest

Random Forest is an ensemble of classification or regression trees. The Random Forest classifier works by partitioning the training set of the data into k subsets and constructing a decision tree out of each subset. All of the subsets are randomly selected. Each decision tree is made by randomly selecting m variables out of all the variables and finding the best split on the selected variables. This is done at each node and continued until a node cannot be split further, leading to the leaf nodes. Each tree votes on a classification after running the test set on each of them. The final classification of the forest is determined by the majority of the decision trees [7].

The algorithm of Random Forest proposed by Dr. Leo Breiman described as below [1].

Given a set of classifiers $t_1(x), t_2(x), \ldots, t_k(x)$, and with the training set drawn at random from the distribution of the random vector $Y, \boldsymbol{X}$, we can get the margin function as

$$mg(\boldsymbol{X}, Y) = av_k I(t_k(\boldsymbol{X}) = Y) \\ -max_{j \neq Y} av_k I(t_k(\boldsymbol{X}) = j) \quad (1)$$

where I is the indicator function. The margin measures the average number of votes at $\boldsymbol{X}$, Y for the right class exceeds the average votes for any other class. The larger the margin, the more accuracy in testing. The generalization error is given by

$$PE* = P_{\boldsymbol{X},Y}(mg(\boldsymbol{X}, Y) < 0) \quad (2)$$

where the subscripts $\boldsymbol{X}$, Y indicate that the probability is over the $\boldsymbol{X}$, Y space.

In random forests, $t_k(X) = t(\boldsymbol{X}, \Theta_k)$. When the forest has a large number of trees, it follows the Strong Law of Large Numbers and the tree structure that:

$$P_{\boldsymbol{X},Y}(P_\Theta(t(\boldsymbol{X}, \Theta) = Y) \\ - max_{j \neq Y} P_\Theta(t(\boldsymbol{X}, \Theta) = j) < 0 \quad (3)$$

where Θ is random vector for splitting selection and Θ_k is independent identically distributed random vector.

2.2 Semi-supervised Random Forest

For Random Forest $F = \{t_1, t_2, \ldots, t_N\}$, when each tree is constructing, it learns a function $F: x \rightarrow y$. The training set is $\{x_i \in x\}_{i = 1 \ldots l}$ and the test set $\{y_i \in y\}_{i = 1 \ldots l.}$ Each internal node of Random Forest is binary split with a partition criterion.

When testing, given a test case x, Random Forest gives the probability estimation for each class as

$$p(k|x) = \frac{1}{N} \sum_{i=1}^{N} p_i(k|x) \quad (4)$$

where $p_i\ (k|x)$ is the probability estimation of class k given by the i^{th} tree [2].

It is estimated by calculating the ratio that class k gets votes from the leaves in the i^{th} tree

$$p_i(k|x) \frac{l_{i,k}}{\sum_{j=1}^{K} l_{i,j}} \quad (5)$$

where $l_{i,k}$ is the number of leaves in the i^{th} tree that vote for class k. Here the overall decision function of Random Forest is as follows.

$$F = argmaxp(k|x) \mid k \in y \quad (6)$$

Xiao Liu *et al* [2] developed the Semi-supervised splitting Random Forest based on the Out-Of-Bag (OOB) samples of the tree which was mentioned in Beriman's research.

The process of Random Forest constructing, an individual training set for each tree (random input) is generated from the original training set using bootstrap aggregation (for labeled set X_l^i and unlabeled setX_u^i). The samples which are not included in the training set is OOB samples and can be used to calculate OOB error. A OOBE is a good feature to do node splitting and semi-supervised training.

Xiao's algorithm for semi-supervised splitting is as follows [2].

Table 1 Algorithm of semi-supervised splitting

```
Algorithm  Semi-supervised Splitting
Input: A set of labeled data $X_l$ for training and a set of labeled $Y_l$ for testing.
Input: A set of unlabeled training data $X_u$.
Input: The size of the forest N.
Output: The Semi-supervised Random Forest F.
Step 1: Initialize an empty forest F.
Step 2: for the i^th decision tree in F
Step 3: Select a new labeled set $X_l^i$ and a new unlabeled set $X_u^i$ using the bootstrap aggregation from the original dataset.
Step 4: Train the tree with only labeled data: $t_l^i = trainTree(X_l^i)$.
Step 5: Calculate the OOBE: $e_l^i = oobe(t_l^i, X_l - X_l^i)$ using the labeled trained tree and the difference between original labeled data and labeled samples.
Step 6: Train the tree with both labeled and unlabeled data: $t_u^i = semiTree(X_l^i, X_u^i)$.
Step 7: Calculate the OOBE: $e_u^i = oobe(t_u^i, X_l - X_l^i)$ using unlabeled trained tree and the difference between original labeled dataset and labeled samples.
Step 8: Compare the OOBE of the above two different trained tree to select final classifiers.
            if $e_l^i > e_u^i$ then
Step 9:         $F = F \cup t_u^i$.
Step 10:    else
Step 11:        $F = F \cup t_l^i$.
Step 12:    end if
Step 13: end for
Step 14: Return F.
```

3 Experiment and Results

3.1 KDD 1999 dataset

KDD 1999 is the Third International Knowledge Discovery and Data Mining Tools Competition, which was held in conjunction with the Fifth International Conference on Knowledge Discovery and Data Mining [8]. The training data contains about 4GB of compressed packet capture data from 7 weeks of network traffic. These data have been processed into 4.9 million connection records, with a set of 41 features. Table 2 is the categories of malicious behavior included in the dataset [7].

Table 2 Categories of Malicious Behavior

Major Categories	Subcategores
Denial of Service (DoS)	Ping of Death, LAND, Neptune, Backscatter, Smurf, Teardrop
User to Root (U2R)	Buffer Overflow, Loadmodule Perl, Rootkit
Remote to Local (R2L)	FTP write, password guessing, IMAP attacks, Multi-hop, PHF, Spy, Warezclient, Warezmaster
Probing	Ipsweeping, Nmap, Portsweeping, Satan

3.2 Experiment

This research used Berman's Random Forest model [1] and implemented the algorithm of Liu *et al* [2] to split the tree nodes. The program was developed based on Kevin-kerandren's example program [9].

In order to compare with supervised Random Forest method, the experiments used the samples as 10, 1000, 3000 and 5000 examples per class and classes as 16, 9, 6 and 6 respectively which were same as those in Mtahir *et al* 's research [7]. Each of the class contained 50% labeled data and 50% unlabeled data.

3.3 Semi-Supervised Random Forest compared with Supervised Random Forest

To compare the supervised machine learning and semi-supervised machine learning, we used Mutahir Nadeem *et al* [7] ‘s results on supervised machine learning. Table 3 and Table 4 are the result comparison for semi-supervised Random Forest and supervised Random Forest.

Table 3 Accuracy for Supervised Random Forest and Semi-supervised Random Forest [7].

Examples		Accuracy %	
# per class	# of classes	Semi-supervised RF	Supervised RF
10	16	N/A	87%
1000	9	95%	99%
3000	6	99%	99%
5000	6	99%	99%

Table 4 Time for Supervised Random Forest and Semi-supervised Random Forest [7].

Examples		Execution Time (sec)	
# per class	# of classes	Semi-supervised RF	Supervised RF
10	16	N/A	1.832
1000	9	9.745	5.745
3000	6	17.13	8.717
5000	6	18.29	14.47

The result under semi-supervised took a little bit longer time to execute but the accuracy is similar as the one used supervised method. The advantage of semi-supervised approach is that it only requires part of the training dataset is labeled.

3.4 Semi-Supervised Random Forest compared with Semi-Supervised Ladder Network

Table 5 and Table 6 are the result comparison for semi-supervised Random Forest and semi-supervised Ladder Network of Mutahir *et al* [7].

To be consistent, the result used the samples as 10, 1000, 3000 and 5000 examples per class. The training data set includes both labeled and unlabeled examples. The ratio of labeled data is 50%.

Table 5 Accuracy for Semi-Supervised Random Forest and Semi-Supervised Ladder Network.

Examples	Accuracy %	
# per class	Semi-supervised RF	Semi-supervised Ladder Network
10	N/A	N/A
1000	95%	92.18%
3000	99%	98.13%
5000	99%	99.03%

Table 6 Time for Semi-Supervised Random Forest and Semi-Supervised Ladder Network.

Examples	Execution Time (sec)	
# per class	Semi-supervised RF	Semi-supervised Ladder Network
10	N/A	N/A
1000	9.745	12.7
3000	17.13	21.8
5000	18.29	22.2

Compared with the result of semi-supervised Random Forest, the Ladder Network took more time to execute. For large amount of data, semi-supervised RF and Ladder Network has similar accuracy. But for the medium amount of data, the Ladder Network has lower accuracy than that of Random Forest.

4 Conclusions

In summary, the Semi-supervised Random Forest performed better than the Semi-supervised Ladder Network. Even though it took more time to train in semi-supervised Random Forest than that in supervised Random Forest, it has advantage because it only needs small amount of labeled data. This is practical because in the real world, it is hard to label large amount of data. Also, training the classifier using unlabeled data can make the margin more accurate and smooth.

Our future work includes improving the training algorithm of semi-supervised Random Forest and making it faster. Also we will develop semi-supervised Support

Vector Machine (SVM) and Deep Belief Network (DBN) on KDD 99 and compare their results.

Acknowledgement

This work is partially supported by NSF under the grant CNS-1460864. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of NSF.

Reference

[1] Leo Breiman. 2001. Random forests. *Machine Learning,* 45(1): 5-32, University of California Berkeley, CA.

[2] Xiao Liu, Mingli Song, Dacheng Tao, Zicheng Liu, Luming Zhang, Chun Chen and Jiajun Bu. 2013. *Semi-supervised Node Splitting for Random Forest Construction*. CVPR2013.

[3] Christian Leistner, Amir Saffari, Jakob Santner and Horst Bischof. *Semi-supervised Random Forests*. Institute for Computer Graphics and Vision. Graz University of Technology.

[4] Md Al Mehedi Hasan, Mohammed Nasser, Biprodip Pal and Shamim Ahmad. 2014. Support vector machine and random forest modeling for intrusion detection system (ids). *Journal of Intelligent Learning Systems and Applications*, 6(1):45.

[5] Mostafa A. Salama, Heba F. Eid, Rabie A. Ramadan, Ashraf Darwish and Aboul Ella Hassanien. 2011.*Hybrid Intelligent Intrusion Detection Scheme,* pages 293-303. Springer Berlin Heidelberg, Berlin, Heidelberg.

[6] Zehra Cataltepe, Umit Ekmekci, Tanju Cataltepe, and Ismail Kelebek. *Online Feature Selected Semi-supervisd Decision Trees for Network Intrusion Detection*. Istanbul Technical University.

[7] Mutahir Nadeem, Ochaun Marshall, Sarbjit Singh, Xing Fang and Xiaohong Yuan. 2016. *Semi-supervised Deep Neural Network for Network Intrusion Detection*. Conference on Cybersecurity Education, Research and Practice, October 29-30, 2016.

[8] KDD cup 1999 data, October 1999.

[9] Kevin-kerandren. 2016, March 13. Random forest in Python. *GitHub.* https://github.com/kevin-keraudren/randomforest-python