A NOVEL EVALUATION APPROACH TO FINDING LIGHTWEIGHT MACHINE LEARNING ALGORITHMS FOR INTRUSION DETECTION IN COMPUTER NETWORK

Yuchen Wang[1], Shuxiang Xu[2] and Qiongfang Huang[1]

[1]College of Computer Science and Technology,
Zhejiang University of Technology, Hangzhou, China
[2]School of Engineering and ICT, University of Tasmania, Australia

ABSTRACT

Building practical and efficient intrusion detection systems in computer network is important in industrial areas today and machine learning technique provides a set of effective algorithms to detect network intrusion. To find out appropriate algorithms for building such kinds of systems, it is necessary to evaluate various types of machine learning algorithms based on specific criteria. In this paper, we propose a novel evaluation formula which incorporates 6 indexes into our comprehensive measurement, including precision, recall, root mean square error, training time, sample complexity and practicability, in order to find algorithms which have high detection rate, low training time, need less training samples and are easy to use like constructing, understanding and analyzing models. Detailed evaluation process is designed to get all necessary assessment indicators and 6 kinds of machine learning algorithms are evaluated. Experimental results illustrate that Logistic Regression shows the best overall performance.

KEYWORDS

Intrusion detection, Lightweight, Comprehensive evaluation, Machine learning algorithms

1. INTRODUCTION

Nowadays, computer network has been widely used in many aspects of human life and network security is becoming more essential to all companies and institutions using the Internet [1]. Intrusion detection is an active network security protection method which can collect and analyze network traffic data, aiming to find out whether there are malicious internal behaviours violating security policy and signs of being attacked by external intruders. Although there are already many off-the-shelf intrusion detection systems existing in market, these systems are too sophisticated or expensive to be afforded by companies and institutions lacking capital and technology. Hence, to find out algorithms which are able to be utilized simply, quickly and effectively is important to the information security of many small and medium-sized enterprises.

DOI : 10.5121/ijnsa.2015.7601

Since 1990s, machine learning techniques began to be used in intrusion detection. Different machine learning algorithms have different suitable circumstances in which they can be applied and get best performance. For a particular kind of field, it is necessary to make comparisons between various types of algorithms based on specific criteria which suit that field. Sangkatsanee et al. [2] used total detection rate as their evaluation index and compared 6 kinds of algorithms. Besides, they also applied information gain to select 12 important features and proposed a pretreatment method to reduce false alarm rate. Mitrokotsa et al. [3] analyzed 5 supervised learning algorithms for intrusion detection in mobile ad-hoc networks, and examined the discrimination capability under different parameters. Huang et al. [4] proposed a novel feature set extracted from application packages and evaluated 4 types of algorithms for detecting malicious behavior in Android platform. Singh et al. [5] evaluated 5 types of machine learning algorithms in terms of classifying Internet Protocol (IP) traffic. Chandrashekhar and Raghuveer [6] analyzed 4 most representative clustering methods and experiment showed that k-means and fuzzy c-means had a good result on accuracy and computation time. Chakchai et al. [7] used both KDD CUP dataset and recent HTTP BOTNET attacks and evaluated six classification schemes. Giray et al. [8] applied normal and noisy datasets for network intrusion detection and evaluated various classification algorithms. The results illustrated that noise is necessary for evaluating algorithms since the realistic environment is noisy. Nadiammai and Hemalatha [9] analyzed various rule and function based classifiers and found that SMO has a better performance than others.

In this paper, we propose a formula which incorporates 6 evaluation indexes, including precision, recall, root mean square error (RMSE), training time, sample complexity and practicability. These factors are all important in industrial areas and should be taken into account together to evaluate algorithms. In particular, precision measures the correctness of detection while recall shows whether algorithms have detect most of the attack data. RMSE indicates the magnitude of residuals and training time is the approximate time needed to get a trained model. Sample complexity means the necessary number of training samples demanded in order to achieve a stable level of accuracy. Practicability indicates whether it is easy to construct, understand and analyze models. Training time, sample complexity and practicability are given out as penalty values where bigger values will have a more negative impact on evaluation result.

We also design a process to get all the needed indicators. First, we set penalty values of practicability for algorithms by analyzing whether it is easy to use these algorithms in terms of common understanding. Second, we prepare training set with various sizes to determine a rough sample complexity for all algorithms. Third, perform all algorithms on one unique dataset and get all necessary assessment indicators. Finally, we calculate the comprehensive evaluation scores for all algorithms and analyze the results. Experimental results illustrate that Logistic Regression shows the best overall performance and Decision Tree also has a high overall evaluation result. These two algorithms have strong potential to be applied to lightweight intrusion detection circumstances. However, MLP and RBF Network may not be practical for building lightweight intrusion detection systems since their overall performances are relatively low.

This paper is organized as follows. Section 2 briefly reviews related work. Section 3 illustrates our proposed evaluation formula, gives information about how to determine penalty values and sample complexity and presents the designed evaluation process. Section 4 shows experimental results and Section 5 summarizes this paper.

2. RELATED WORK

2.1. Machine learning

Machine learning is a set of algorithms which are able to find potential patterns in data and use these patterns to predict unlabeled data. In 1959, Arthur Samuel [10] informally defined machine learning as a subject which tries to figure out how to make computer solve real problems automatically without programming detailed code. For example, he developed the Samuel Checkers-playing Program which had the ability to learn a reasonable way to play and managed to defeat many human players. In this section, we briefly introduce the machine learning algorithms used in this paper, which are Decision Tree, Naïve Bayes, Multilayer Perceptron, Radial Basis Function Network, Logistic Regression and Support Vector Machine.

2.1.1. Decision Tree

Decision Tree [11] is a very important machine learning algorithm. It is a prediction model containing a series of nodes including one root node and a number of checking nodes. Each node classifies input data based on the mapping relation stored on that node. The process of classifying data starts with the root node and keeps checking data on each node until the data reach the end of the tree and get an output. The classification process is human-understandable and easy to implement. Also, little effort needs to be paid in terms of pretreating data such as reducing redundancy or transforming discrete data into numerical value.

2.1.2. Naïve Bayes

Naïve Bayes [12] is a simple probabilistic classifier based on Bayes' theorem. This method assumes that the features in sample are not relevant with each other. Although this assumption is not true in many cases, this method is still able to achieve a high accuracy in many practical applications. To estimate its main parameters, i.e. mean and variance of features, Naïve Bayes only needs a small amount of data and linear time, which makes it much more efficient and less cost on computation. Due to this simplicity in building features and calculation, be applied in many fields like information retrieval, text classification and automatic medical diagnosis.

2.1.3. Multilayer Perceptron

Multilayer Perceptron (MLP) [13] is a feedforward artificial neural network, which has been widely used in applications like speech recognition, image recognition and machine translation. A MLP has one or more hidden layers which connect input layer and output layer. In order to train the network, an algorithm called backpropagation is often used. This model has a strong ability to classify nonlinear data and to approximate functions.

2.1.4. Radial basis function network

Radial basis function network (RBF Network) [14] is a kind of artificial neural networks with radial basis functions as its activation functions. Different from the MLP, it often contains only one hidden layer and uses Euclidean distance to combine input data instead of inner product. This network can be used in function approximation, time series prediction and control of a chaotic time series.

2.1.5. Logistic Regression

Logistic Regression [15] is widely applied in many fields now, including sociology, marketing and computational advertising. One commonly used model is called binary logistic model which, by utilizing a logistic function, calculates the probability of a binary output value with several features. These features can be both continuous and discrete types, such as real, binary and categorical variables. The implementation is relatively easy and the outcomes also have clear explanation, but the features used should be independent with each other in order to get accurate parameters.

2.1.6. Support Vector Machine

Support Vector Machine (SVM) [16] has been regarded as the state-of-the-art method in machine learning. It aims to find the maximum-margin hyperplane that has the largest distance to the nearest training sample, thus considerably reducing the generalization error. Samples are classified based on the side a sample falls. SVM can effectively cope with both linear and nonlinear data. For samples that are not linearly separable, a practical way is to apply kernel trick to map data to a higher dimension which can discriminate data easier. In fields like hand-written characters and medical science, SVM has been used to solve these problems.

2.2. KDD99 dataset

The KDD99 dataset [17] was created by MIT Lincoln Labs. The researchers built up an environment and gained 9 weeks of raw TCP dump data from a simulated military network. Both normal data and attack data were created and recorded. The attack data mainly have 4 types, which are denial-of-service (DOS), unauthorized remote access (R2L), unauthorized local access to privileges (U2R) and probing. There are 41 features in this dataset and they can be divided into 4 categories: basic features of individual TCP connections, content features within a connection, traffic features in a two-second time window and features from recent 100 connections to the same host.

2.3. Evaluation method

2.3.1. Confusion matrix

In machine learning, confusion matrix [18] is generally used to visualize the performance of an algorithm. This method illustrates test results in the form of a matrix, with each column of the matrix representing a prediction of samples to specific classes, and each line representing samples in actual classes. The typical confusion matrix with two classes is shown below:

Table 1. Confusion matrix with two classes

	Predicted as positive	**Predicted as negative**
Actual positive sample	True Positive	False Negative
Actual negative sample	False Positive	True Negative

In particular, the meaning of each cell is shown in the following table:

Table 2. Meaning of confusion matrix

Prediction result	Abbreviation	Meaning
True Positive	TP	The actual positive sample is correctly predicted to be positive.
False Negative	FN	The actual positive sample is wrongly predicted to be negative.
False Positive	FP	The actual negative sample is wrongly predicted to be positive.
True Negative	TN	The actual negative sample is correctly predicted to be negative.

Based on the above four basic definitions, we can get the following evaluation measurements:

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

$$Accuracy = \frac{TP + TN}{TP + FN + TN + FP} \tag{3}$$

$$F_1 Score = 2\frac{Precision \times Recall}{Precision + Recall} \tag{4}$$

where Precision represents the proportion of samples predicted as positive that are true positive, while Recall means the proportion of positive samples that are actually predicted as positive. Accuracy indicates the proportion of the samples that are correctly classified. F_1Score represents a trade-off between Precision and Recall, which can usually be used as a single index to measure the efficiency of algorithms.

2.3.2. Root Mean Squared Error

Root Mean Squared Error (RMSE) is often used to measure the deviation between the actual value of samples and the value predicted by algorithms. It calculates the square root of the mean of the square of all errors. The less RMSE means the algorithms' predictions are closer to actual value. Give n samples y and their corresponding predicted values $\hat{y}$, the RMSE is defined as:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2} \tag{5}$$

3. EVALUATION METHOD FOR FINDING LIGHTWEIGHT ALGORITHMS

3.1. Proposed evaluation formula

In this paper, we propose a novel evaluation method to find out a lightweight algorithm for intrusion detection. We incorporate 6 indexes into our comprehensive measurement, including precision, recall, RMSE, training time, sample complexity and practicability. Specifically, training time means the time needed to train a model, sample complexity is the number of samples needed in order to achieve its best performance. Practicability indicates whether it is easy to construct, understand and analyze trained models.

Given a set of algorithms $M=\{M_1,\ldots,M_6\}$, we proposed below formula for the evaluation of each algorithm M_i:

$$Ranker(M_i) = F_1Score_i \times (1 - RMSE_i) \times (1 - PV_i) \tag{6}$$

where F_1Score, according to formula (4), has combined precision and recall, and PV_i means penalty value for algorithm i. In order to take training time, sample complexity and practicability into account, we set corresponding penalty values PV_{i1}, PV_{i2} and PV_{i3} respectively to represent above 3 dimensions for each algorithm i. A bigger penalty values means longer time to train, more necessary numbers of samples needed and more complicated to use.

According to above definition of penalty values, the penalty value for algorithm i is calculated by:

$$PV_i = \alpha \sum_{j=1}^{3} PV_{ij} \tag{7}$$

where a is used to normalize the penalty values and is computed by:

$$\alpha = \frac{1}{\sum_{i=1}^{6}\sum_{j=1}^{3} PV_{ij}} \tag{8}$$

3.2. Define penalty values for practicability

In our evaluation scheme, we assign number 1, 2 and 3 to the penalty values PV_{ij} to indicate algorithms' training time, sample complexity and practicability. The penalty values for training time and sample complexity will be determined by experimental results in next section. As question about the advantages and disadvantages of different machine learning algorithms is a really complex topic, in this section, we will give penalty values for practicability by analyzing whether it is easy to use the algorithms in terms of common understanding. The detailed analysis is given below.

The mathematic forms for Logistic Regression and Naïve Bayes are both simple to implement and easy to train. The outputs of them both have a reasonable probabilistic interpretation, which will be useful for other purpose like ranking instead of classification. In addition, there are

already many learning systems based on these algorithms such as predicting buying preferences and filtering spam emails. Therefore, we set the penalty values for Logistic Regression and Naïve Bayes as 1.

The classification process of Decision Tree is understandable for humans and it can easily treat nonlinear data. However, the model is easily to overfit data and repeated modeling process is often needed. It also has problems in dealing with missing value so that thorough pretreatment for data is necessary. For SVM, although it can effectively cope with nonlinear data, the choice of kernel function will be a potential question. Besides, the trained model is incomprehensible for humans, which prevent further improvement by analyzing the model. It also needs massive tuning process to adjust its models to perform better, which may be hard for practical environment. Thus, we set penalty values for Decision Tree and SVM as 2.

Despite the fact that MLP and RBF Network have the ability to achieve a high accuracy, they are very complicated in terms of finding appropriate network topology, understanding the meaning of the learning process and interpreting the output value. We therefore set penalty values for MLP and RBF Network as 3.

The penalty values for practicability is organized in below table:

Table 3. Penalty values for practicability

Algorithms	Penalty values for practicability
Decision Tree	2
Logistic	1
MLP	3
Naïve Bayes	1
RBF Network	3
SVM	2

3.3. Determine sample complexity

To determine the samples needed for every algorithms to get its best performance, we set up a sequence of datasets S with various sizes. Given a set of algorithms $M=\{M_1,\ldots,M_6\}$, we train every algorithm i on dataset S_j and get a hypothesis noted as h_{ij}:

$$h_{ij} = train\left(M_i, S_j\right) \tag{9}$$

For each h_{ij}, we test its performance on testing set and get its F_1Score:

$$F_1Score_{ij} = test(S_{test}, h_{ij}) \tag{10}$$

To determine the number of samples at which point a particular algorithm is convergence, we calculate change of F_1Score on every two consecutive sizes of training sample, i.e. the gradient of F_1Score:

$$f'_{ij}\left(S_j\right) \approx F_1Score_{ij} - F_1Score_{i(j-1)} \tag{11}$$

If the change rate $f'_{ij}(S_j)$ is less than a threshold, we will choose S_j as its sample complexity. The choice of threshold for an algorithm can be specified manually from the result of experiment. For instance, we can pick up threshold of the change rate of F_1Score when the second derivative of F_1Score is less than a selected small value and does not fluctuate as the size of training set grows.

3.4. Designed evaluation process

Our whole evaluation process can be described in following steps:

Step 1: as discussed in section 3.2, we set penalty values of practicability for algorithms by analyzing whether it is easy to use these algorithms in terms of common understanding.

Step 2: prepare training set with various sizes to determine a rough sample complexity of all algorithms and set penalty values of sample complexity for each algorithm. We pick the maximum number among all algorithms to ensure that all algorithms can reach their best performance, and then build experimental training set to train all algorithms.

Step 3: perform all algorithms on one unique dataset and, by using 10-fold cross-validation method, get performance results such as F_1Score, RMSE and training time. Then, we can set penalty values of training time for each algorithm and all required evaluation measurements are obtained.

Step 4: calculate the comprehensive evaluation scores for all algorithms based on our proposed formula (6), analyze the result and choose the algorithms with best overall performance.

4. EXPERIMENTAL RESULTS

The machine learning tool Weka [19] is selected to run all algorithms. The parameters for Decision Tree, Naïve Bayes, MLP, RBF Network and Logistic Regression use default setting. For SVM, we use radial basis function as kernel function and set parameter of normalize as true.

To determine sample complexity, we randomly pick 493965 samples from KDD99 training dataset and divide them into 13 subsets $S=\{S_1,\ldots,S_{13}\}$. The sizes of these subset range from 60 to 247010. In particular, they are 60, 121, 242, 483, 965, 1930, 3860, 7720, 15439, 30877, 61753, 123505 and 247010. In addition, we also randomly select 6000 samples from KDD99 testing dataset to test performance. The F_1Score we get on every training set is showed below:

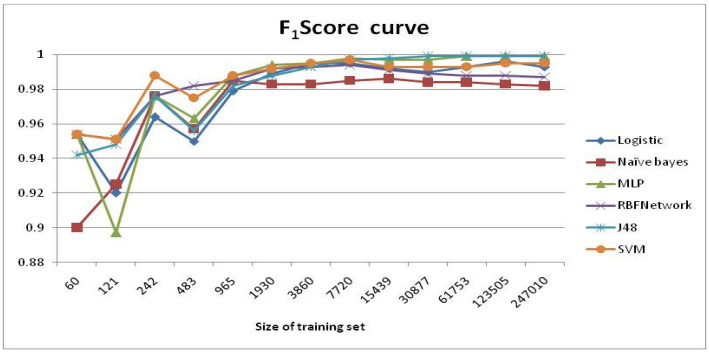

Figure 1. F_1Score changes with the size of training set

The above figure shows that the performances of algorithms, when the size of training set increases, all go up and begin to remain stable after a particular size of training set. Different algorithms need different number of samples to achieve its maximum detection rate. To get the sample complexity of each algorithm, we calculate change of F_1Score on every two consecutive sizes of training samples. The obtained gradient change figure with respect to training set size is showed below:

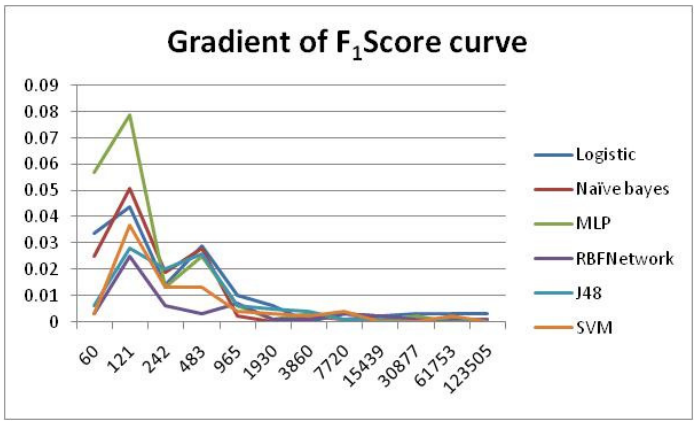

Figure 2. Gradient of F_1Score curve

It is clear from above figure that the gradient becomes smaller when the size of training set grow up. We select the thresholds for judging convergence when the change rates are less than specific small values. These thresholds will vary from algorithm to algorithm. The obtained sample complexity list for all algorithms is showed below:

Table 4. Sample complexity of 6 machine learning algorithms

Algorithms	Sample complexity
Decision Tree	30877
Logistic	3860
MLP	61753
Naïve Bayes	965
RBF Network	3860
SVM	1930

The list shows that Naïve Bayes needs the least number of samples to get a stable accuracy (about 1000), while MLP needs more than 60000 samples, which is the highest among all algorithms. In order to evaluate the performance of algorithms, we must ensure that all algorithms are trained on the same training set, which means we should choose a set containing more than about 60000 samples.

According to above test result, we establish our experimental dataset by randomly pick training samples from KDD99 dataset for evaluating algorithms, with the size of experimental dataset being 82337. By using 10-fold cross-validation, nine tenth of whole dataset is used as training data. Thus, in our experiment, 74104 samples are used as training data, which is bigger than 60000, making sure that all algorithms can reach their best performance. The detailed information of our evaluation environment is given below:

Table 5. Experiment setup for evaluation

Sample number	82337
Source of sample	KDD99 dataset
Model validation technique	10-fold cross-validation
Algorithm tool	Weka 3.6.13
SVM Tool	Libsvm 3.20
CPU type	Intel(R) Core(TM) i3-3120M CPU @ 2.50GHz
RAM	4 GB
Operating system	Windows 7 Ultimate 32 bit

We perform 6 machine learning algorithms on above environment and get following result table:

Table 6. Performance results of 6 machine learning algorithms

Algorithms	**F_1Score**	**RMSE**	**Training time**
Decision Tree	0.999	0.000623	15.79
Logistic	0.994	0.006016	10.6
MLP	0.999	0.000953	2538.49
Naïve Bayes	0.982	0.017732	1.37
RBF Network	0.988	0.01217	22.37
SVM	0.993	0.006477	736.34

Based on table 4 and above table, we can assign penalty values of training time and sample complexity for all algorithms.

For training time, as can be seen from above table, MLP and SVM need training time which is significantly higher than other algorithms, and the other 4 types of algorithms require relatively low training time. Thus, we assign 2 to MLP and SVM, and 1 to others.

For sample complexity, table 4 shows that Naïve Bayes needs a considerably low number of samples. Hence we set 1 for it. Logistic Regression, RBF Network and SVM demand approximately the same number of samples so that we assign 2 for them. The samples needed for Decision Tree and MLP are much higher than other algorithms and we set 3 for them.

Now, all penalty values for measuring performance are gotten and are summarized in following table:

Table 7. Penalty values for training time, sample complexity and practicability

Algorithms	**Penalty values**		
	Training time	**Sample complexity**	**Practicability**
Decision Tree	1	3	2
Logistic	1	2	1
MLP	2	3	3
Naïve Bayes	1	1	1
RBF Network	1	2	3
SVM	2	2	2

Based on above table, we can calculate penalty values for every algorithm according to formula (7) and (8). At this stage, all necessary indicators for evaluating performance are achieved. The F_1Score, RMSE and penalty values for every algorithm are summarized in following table:

Table 8. All performance indicators

Algorithms	F_1Score	RMSE	Penalty values
Decision Tree	0.999	0.000623	0.182
Logistic	0.994	0.006016	0.121
MLP	0.999	0.000953	0.242
Naïve Bayes	0.982	0.017732	0.091
RBF Network	0.988	0.01217	0.182
SVM	0.993	0.006477	0.182

According to our proposed evaluation formula (6), we can get the comprehensive score for every algorithm showed in below figure:

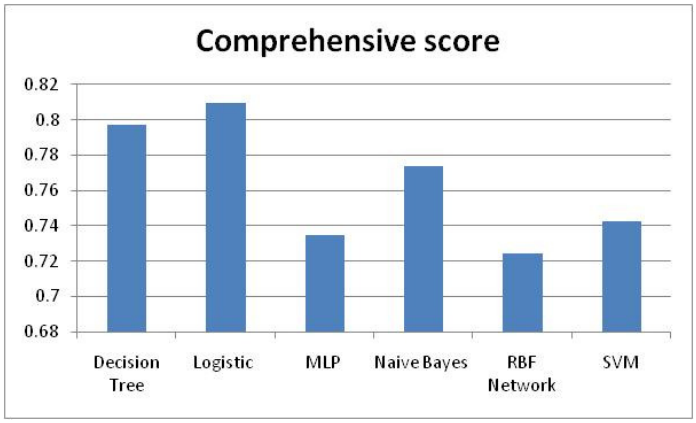

Figure 2. Comprehensive scores of 6 machine learning algorithms

As can be seen from above figure and previous tables, Logistic Regression has the biggest comprehensive evaluation score and Decision Tree comes next, with their score much bigger than other 4 algorithms. This means that Logistic Regression shows the best overall performance in terms of 6 evaluation aspects. Decision Tree also have a high overall evaluation result. Thus, these two algorithms have strong potential to be applied to lightweight intrusion detection circumstances. Naïve Bayes ranks third and SVM ranks at fourth place. Although Naïve Bayes has the lowest penalty values regarding training time, sample complexity and practicability, the detection rate and RMSE are the weakness points of it and prevent it from getting a high overall performance. MLP has very high detection rate and low RMSE, but the model is hard to train, understand and do incremental modification, making the overall performance relatively low than Logistic Regression and Decision Tree. Another neural network-based algorithm, RBF Network, also shows a very low overall performance because it has no significant advantages in terms of all evaluation criteria.

5. CONCLUSIONS

In this paper, we propose a novel evaluation formula which incorporates 6 indexes into our comprehensive measurement, including precision, recall, root mean square error, training time, sample complexity and practicability. Training time, sample complexity and practicability are given out as penalty values where bigger values will have a more negative impact on evaluation result. A detailed evaluation process is also designed to set penalty values of training time, sample complexity and practicability for all algorithms and get all other evaluation indicators. First, we set penalty values of practicability for algorithms by analyzing whether it is easy to use these algorithms in terms of common understanding. Second, we prepare training set with various sizes to determine a rough sample complexity for all algorithms. Third, perform all algorithms on one unique dataset and get all necessary assessment indicators. Finally, we calculate the comprehensive evaluation scores for all algorithms and analyze the results. Experimental results illustrate that Logistic Regression shows the best overall performance. Decision Tree also has a high performance, while MLP and RBF Network may not be practical for building lightweight intrusion detection systems.

ACKNOWLEDGEMENTS

I would like to extend my sincere gratitude to my teacher, Xu Shuxiang, for his inspiring advice and instant guidance. I am deeply grateful of his help.

REFERENCES

[1] Hoque M S, Mukit M A, Bikas M A N. An Implementation of Intrusion Detection System Using Genetic Algorithm[J]. International Journal of Network Security & Its Applications, 2012, 4(2):109-120.

[2] Sangkatsanee P, Wattanapongsakorn N, Charnsripinyo C. Practical real-time intrusion detection using machine learning approaches[J]. Computer Communications, 2011, 34(18): 2227-2235.

[3] Mitrokotsa A, Dimitrakakis C. Intrusion detection in MANET using classification algorithms: The effects of cost and model selection[J]. Ad Hoc Networks, 2013, 11(1): 226-237.

[4] Huang C Y, Tsai Y T, Hsu C H. Performance Evaluation on Permission-Based Detection for Android Malware[M]// Advances in Intelligent Systems and Applications - Volume 2Springer Berlin Heidelberg, 2013:111-120.

[5] Singh K, Agrawal S. Comparative analysis of five machine learning algorithms for IP traffic classification[C]//Emerging Trends in Networks and Computer Communications (ETNCC), 2011 International Conference on. IEEE, 2011: 33-38.

[6] Chandrashekhar A M, Raghuveer K. Performance evaluation of data clustering techniques using KDD Cup-99 Intrusion detection data set[J]. International Journal of Information & Network Security, 2012.

[7] So-In C, Mongkonchai N, Aimtongkham P, et al. An evaluation of data mining classification models for network intrusion detection[C]// Digital Information and Communication Technology and it's Applications (DICTAP), 2014 Fourth International Conference onIEEE, 2014:90-94.

[8] Giray S M, Polat A G. Evaluation and Comparison of Classification Techniques for Network Intrusion Detection[C]// 2013 IEEE 13th International Conference on Data Mining WorkshopsIEEE, 2013:335-342.

[9] Nadiammai G V, Hemalatha M. Perspective analysis of machine learning algorithms for detecting network intrusions[C]// Computing Communication & Networking Technologies (ICCCNT), 2012 Third International Conference onIEEE, 2012:1 - 7.

[10] Arthur Samuel: Pioneer in Machine Learning [EB/OL].http://infolab.stanford.edu/pub/voy/museum/samuel.html

[11] Farid D M, Harbi N, Rahman M Z. Combining Naive Bayes and Decision Tree for Adaptive Intrusion Detection[J]. International Journal of Network Security & Its Applications, 2010, 2(2).
[12] Salah K, Kahtani A. Performance evaluation comparison of Snort NIDS under Linux and Windows Server[J]. Journal of Network and Computer Applications, 2010, 33(1): 6-15.
[13] Sivaram G S V S, Hermansky H. Sparse multilayer perceptron for phoneme recognition[J]. Audio, Speech, and Language Processing, IEEE Transactions on, 2012, 20(1): 23-29.
[14] Chandrashekhar A M, Raghuveer K. Fortification of Hybrid Intrusion Detection System Using Variants of Neural Networks and Support Vector Machines[J]. International Journal of Network Security & Its Applications, 2013, 5(1).
[15] Hosmer Jr D W, Lemeshow S. Applied logistic regression[M]. John Wiley & Sons, 2004.
[16] Sedjelmaci H, Feham M. Novel hybrid intrusion detection system for clustered wireless sensor network[J]. International Journal of Network Security & Its Applications, 2011, 3(4).
[17] Pfahringer B. Winning the KDD99 classification cup: bagged boosting[J]. ACM SIGKDD Explorations Newsletter, 2000, 1(2): 65-66.
[18] Batista G E, Prati R C, Monard M C. A study of the behavior of several methods for balancing machine learning training data[J]. ACM Sigkdd Explorations Newsletter, 2004, 6(1): 20-29.
[19] Hall M, Frank E, Holmes G, et al. The WEKA Data Mining Software: An Update[J]. Acm Sigkdd Explorations Newsletter, 2009, 11(1):10-18