Memory-Efficient Random Forest Generation Method for Network Intrusion Detection

Seok-Hwan Choi[1], DongHyun Ko[2], SeonJin Hwang[3] and Yoon-Ho Choi[4]

Abstract— **Along with the steady growth of wired and wireless networks, the various new attacks targeting networks are also constantly emerging and transforming. As a efficient way to cope with various attacks, the Random Forest(RF) algorithm has frequently been used as the core engine of intrusion detection because of the faster learning speed and the higher attack detection accuracy. However, the RF algorithm has to input the number of the tree composing the forest as a parameter. In this paper, we proposed a new algorithm that limit the number of trees composing the forest using the McNemar test. To evaluate the performance of the proposed RF algorithm, we compared learning time, accuracy and memory usage of the proposed algorithm with the original RF algorithm and other algorithm by using the KDDcup99 dataset. Under the same detection accuracy, the proposed RF algorithm improves the performance of the original RF algorithm by as much as 97.76% at learning time, 91.86% at test time, and 99.02% in memory usage on average.**

I. Introduction

Along with the steady growth of wired and wireless networks, various attacks targeting the network are also constantly emerging and transforming [1]. To use the network securely from the attacks, it is necessary to introduce Intrusion Detection System (IDS), which monitors the network and identifies known or unknown attack events.

In recent intrusion detection field, there has been a lot of interest in an ensemble machine learning method, which generates multiple classifiers and aggregates their results. As a representative ensemble machine learning method, the Random Forest (RF) algorithm has frequently been used as the core engine of intrusion detection because of the faster learning speed and the higher detection accuracy [2], [3]. However, there is a limitation that the RF algorithm has to use the number of trees composing the forest as a input parameter. Most IDS based on the RF algorithm use this input parameter as a sufficiently large values. This large value not only wastes memory and time costs of the IDS, but can also decrease the IDS's detection performance.

In this paper, we propose a new McNemar test based RF algorithm which limits the number of trees dynamically. Under the same detection accuracy, the proposed RF algorithm shows the higher performance than the original RF algorithm[4] at learning time and test time, and memory usage.

This paper consists of as follows. In section II, we overview research works related to the RF algorithm. After describing the operation of the proposed RF algorithm in section III, we show the experimental results for evaluating the performance of the proposed RF algorithm in section IV. Finally, we summarize this paper in section V.

II. Related Works

In this section, we overview research works that use the RF algorithm for intrusion detection and that limit the number of trees composing the forest.

To overcome the limitations of IDS that used data mining techniques such as [5], J. Zhang et. al. first proposed a method to apply RF algorithm to network IDS [4]. In [4], the RF algorithm is more suitable for network IDS because it can efficiently process large datasets compared to [5] that based on association rules. However, J. Zhang et. al.'s method was repeated by increasing the number of trees from 10 to 50 increase by 5 to find the optimal detection rate. It can cause a waste of memory and computation cost when the learning dataset is large.

A. Cuzzocrea et. al. proposed a algorithm for optimizing the number of trees in a RF algorithm using an information-theoretic approach [6]. To this end, A. Cuzzocrea et. al. used the relationship between the predictive power and the number of trees. However, since the predictive power is calculated in the process of finding the optimal number of trees, a large amount of memory and computation cost are wasted when the data set is large. To reduce memory and learning time, P. Latinne et. al. proposed a algorithm to limit the number of trees by using the McNemar test [7]. P. Latinne et. al. provides the same performance with fewer number of trees by comparing with the RF algorithm composed of the maximum number of trees. However, there is a problem that the optimal number of trees is dependent on the predefined maximum number of trees.

To overcome the performance degradation of the previous RF algorithms due to the dependancy on learning data and the number of trees, we propose a new algorithm that dynamically limits the number of trees using the McNemar test from each iteration. The proposed algorithm is similar to P. Latinne et. al.'s algorithm [7] in that it limits the number of trees using McNemar test. Compareed to the P. Latinne et. al.'s algorithm, the proposed algorithm does not need to know

[1]Seok-Hwan Choi is with School of Computer Science and Engineering, Pusan National University, Busan, 26241, Republic of Korea daniailshh@pusan.ac.kr

[2]DongHyun Ko is with School of Computer Science and Engineering, Pusan National University, Busan, 26241, Republic of Korea uyt1209@pusan.ac.kr

[3]SeonJin Hwang with School of Computer Science and Engineering, Pusan National University, Busan, 26241, Republic of Korea unlockable@pusan.ac.kr

[4]Yoon-Ho Choi is is with School of Computer Science and Engineering, Pusan National University, Busan, 26241, Republic of Korea yhchoi@pusan.ac.kr

the predefined maximum number of trees and to perform additional learning using the maximum number of trees.

III. Proposed algorithm

In this section, we overview the operation of the proposed algorithm in details. To determine whether to increase the number of trees based on a McNemar test dynamically, the proposed algorithm consists of three functional modules: (1) forest generation; (2) McNemar calculation; and (3) learning termination.

Algorithm 1 Overall operation of Proposed algoritm

Require: T the training dataset
Require: The global variable *isBreak* = False
Require: The iteration count $i = 0$
Require: The result of McNemar test R_{Mc}
1: **procedure** *ProposedAlgorithm*(T)
2: **while** *isBreak* == *FALSE* **do**
3: *Forest* = ForestGeneration(T);
4: **if** i % 2 == 0 **then**
5: R_{Mc} = McNemarCalculation(*Forest*);
6: *IsBreak* = LearnTermination(R_{Mc});
7: **end if**
8: $i = i + 1$;
9: **end while**
10: **end procedure**

As shown in Algorithm 1, the forest generation module creates a decision tree and adds it to the forest (Line 3). The procedure of forest generation is the same as the conventional RF algorithm [8], so we do not describe it in details. The McNemar calculation and the learning termination modules are performed every two iteration, this is because the number of "True to False" and the number of "False to True" have a large value alternately when performing each iteration (Line 4). The McNemar calculation module performs the McNemar test between the *Forest* of current iteration and the *Forest* of previous iteration (Line 5).

The McNemar test is a statistical hypothesis testing method used on paired nominal data [9]. It is applied to 2×2 contingency tables which used to test whether there is a difference between "before" and "after". As shown in Table I, the contingency table tabulates the outcomes of two tests on a sample of n subjects. The Mcnemar test requires the following assumptions. First, each of nominal data has two possible outcomes such as $TRUE$ or $FALSE$. Second, the null hypothesis $H0$ that there is no difference between before and after is the same as the result of testing whether there is a difference before and after the specific experiment. Finally, the sum of $b(FALSEtoTRUE)$ and $c(TUREtoFALSE)$ must be greater than 20. For the null hypothesis $H0$, the McNemar test can be calculated as follows:

$$R_{Mc} = \frac{(b-c)^2}{(b+c)}. \quad (1)$$

In Equation (1), The hypothesis $H0$ is rejected if test statistic is greater than significance level p. We used the value

TABLE I: contingency table

	Test 2 positive	Test 2 negative	Row total
Test 1 positive	a	b	a+b
Test 1 negative	c	d	c+d
Column total	a+c	b+d	n

of significance level p as 0.05, which means 3.841459 as the test statistic value. That is, when the test statistic is greater than 3.841459, the specific experiment have significantly difference between before and after. After the McNemar result R_{Mc} is computed, the learning termination module determines whether to generation an additional tree (Line 6). Specifically, If R_{Mc} is greater than 3.841459(the significance level $p < 0.05$), the learning termination module determines that the forest performance is unstable, and performs the additional tree derivation. On the contrary, if R_{Mc} is lower than 3.841459, the learning termination module determines that the forest performance is stable, and stops the additional tree derivation.

As mentioned above, the proposed algorithm can limit the number of trees during the learning of random forest without the analysis of learning datasets and the separate input parameters.

IV. Experimental Evaluation

To evaluate the performance of the proposed algorithm, we compared learning time, test time and memory usage of the proposed algorithm with the original RF algorithm [4] and P. Latinne et. al.'s algorithm [7]. Also, as in Zhang et. al. s experiments, the number of trees composing a forest of the original RF algorithm is fixed at 100.

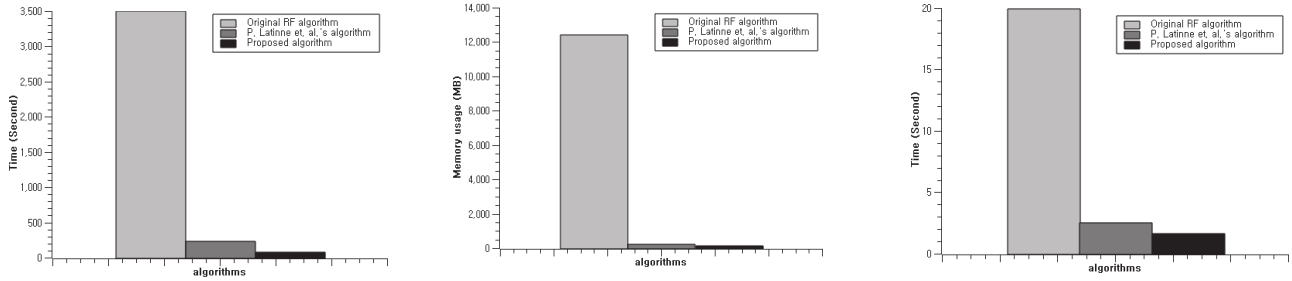
A. Experimental Environment

We evaluated the performance of the proposed algorithm in intrusion detection field by using a dataset which are mainly used for IDS performance evaluation: KDDcup99 [10]. The KDDcup99 dataset consists of 41 features and 5 classes: Probe, DoS, U2R, R2L and Normal. In this paper, 494021 training data and 311029 test data were used in the KDDcup99 dataset. Also, we measured the performance of the proposed algorithm on the Ubuntu 14.04.5 LTS with kernel version 4.2.0-27-generic, 2.40GHz CPU clock(Intel Xeon CPU E5-2630 v3), and 32GB memory.

B. Experimental Results

For the KDDcup99 dataset, the result of McNemar test is less than significance level p after the 4_{th} decision tree is generated. Thus, the number of trees composing the forest of the proposed algorithm is 4, which is 96 less than the original RF algorithm and 6 less than the P. Latinne et. al.'s method. However, the detection accuracy of the three algorithms is the same at 99.97%. This means that under the same detection accuracy, the proposed algorithm constitutes the forest with the fewest number of trees.

In Figure 1, we shows the experimental results of each algorithm. As shown in Figure 1(a), for the learning time,



(a) Learning time (b) Memory usage (c) Test time

Fig. 1: The Comparison graphs of experimental results

TABLE II: Experimental Results

	Original RF algorithm	Proposed algorithm	P. Latinne et. al.'s algorithm [1]
#Trees	100	4	10
Accuracy	99.97%	99.97%	99.97%
Memory usage	12398.89MB	120.73MB	203.11MB
Learning Time	3498.79sec	78.34sec	236.41sec
Test Time	19.91sec	1.62sec	2.51sec

the proposed algorithm was 78.34 seconds on average, which was faster than 3,420.45 seconds and 158.07 seconds of the original RF algorithm and the P. Latinne et. al.s algorithm, respectively. In Figure 1(b) we shows the comparison result for memory usage. The memory usage of the proposed algorithm is 120.73 MB, which is lower than the other two algorithms by as much as 99.02% and 40.55% respectively. Also, as shown in Figure 1(c), the test time of the proposed algorithm is 1.62 seconds, which takes 18.29 seconds less than the original RF algorithm that takes 19.91 seconds and 0.89 seconds less than the P. Latinne et. al.'s algorithm that take 2.51 seconds. We summarize the experimental results in Table II.

Through the above experiments, under the same detection accuracy, the proposed algorithm shows that learning time, test time and memory usage are efficient compared to the original RF algorithm and P. Latinne et. al.'s algorithm.

V. Conclusion

As a representative ensemble machine learning method, the RF algorithm has commonly been used as the core engine of intrusion detection system. In this paper, to decrease the number of trees in the RF algorithm, we proposed a new algorithm that limit the number of trees composing the forest. The proposed algorithm finds the stop point of the forest derivation using McNemar test which is a statistical hypothesis testing method used on paired nominal data. The proposed method estimated the minumum number of trees composing a forest while ensuring the same detection accuracy. From the experimental results under KDDcup99 dataset, we observed that compared to the original RF algorithm, the proposed algorithm showed the fast learning time by as much as 97.76% on average, the fast test time by as much as 91.86% on average and the less memory usage by as much as 99.02% on average. Also, compared to P. Latinne et. al.'s algorithm, the proposed algorithm showed the fast learning time by as much as 66.86% on average, the fast test time by as much as 40.55% on average and the less memory usage by as much as 35.45% on average. From these experimental results, we believe that the proposed algorithm can be be efficiently applied to the low power IDS environment.

ACKNOWLEDGMENT

This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program(2014-1-00743) supervised by the IITP(Institute for Information & communications Technology Promotion) and under the National Program for Excellence in SW (2016-0-00019), supervised by the IITP(Institute for Information & communications Technology Promotion).

References

[1] INDEX, Cisco Visual Networking. Global IP traffic growth, 20162021. Cisco White Paper, June. 2017.
[2] N. Farnaaz, M. A. Jabbar, Random Forest Modeling for Network Intrusion Detection System. Procedia Computer Science, vol. 89, pp. 213-217, 2016.
[3] S. R. Johnson, A. Jain, An Improved Intrusion Detection System using Random Forest and Random Projection. Probe, 2, U2R. International Journal of Scientific and Engineering Research, Vol 7, Oct. 2016.
[4] J. Zhang, M. Zulkernine, and A. Haque, Random-forests-based network intrusion detection systems. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), Vol. 38, pp. 649-659, Sept. 2008.
[5] P. Kaur, A. Kumar Sharma, S. K. Prajapat, Madam id for intrusion detection using data mining, International Journal of Research in IT & Management vol. 2, pp. 256-263, Feb. 2012.
[6] A. Cuzzocrea, S. L. Francis, M. M. Gaber, An information-theoretic approach for setting the optimal number of decision trees in random forests, In Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference, pp. 1013-1019, Oct. 2013.
[7] P. Latinne, O. Debeir, C. Decaestecker, Limiting the number of trees in Random Forests. Multiple Classifier Systems, pp. 178-187, 2001.
[8] L. Breiman, Random forests. Machine learning, 45(1), 5-32. 2001.
[9] Q. McNemar Note on the sampling error of the difference between correlated proportions or percentages, Psychometrika, vol. 12, pp. 153-157, 1947.
[10] The UCI KDD Archive, "KDD Cup 1999 Data," http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html