A Sequential Classifiers Combination Method to Reduce False Negative for Intrusion Detection System

Sornxayya PHETLASY\(^{(a)}\), Nonmember, Satoshi OHZAHATA\(^{(b)}\), Celimuge WU\(^{(c)}\), and Toshihito KATO\(^{(d)}\), Members

SUMMARY Intrusion detection system (IDS) is a device or software to monitor a network system for malicious activity. In terms of detection results, there could be two types of false, namely, the false positive (FP) which incorrectly detects normal traffic as abnormal, and the false negative (FN) which incorrectly judges malicious traffic as normal. To protect the network system, we expect that FN should be minimized as low as possible. However, since there is a trade-off between FP and FN when IDS detects malicious traffic, it is difficult to reduce the both metrics simultaneously.

In this paper, we propose a sequential classifiers combination method to reduce the effect of the trade-off. The single classifier suffers a high FN rate in general, therefore additional classifiers are sequentially combined in order to detect more positives (reduce more FN). Since each classifier can reduce FN and does not generate much FP in our approach, we can achieve a reduction of FN at the final output. As a result, we have the both of high accuracy while improving the sensitivity and accuracy.

key words: sequential classifiers combination, false negative, intrusion detection, machine learning

1. Introduction

Recently, people have been using the Internet in their daily life such as communication, research, commerce, entertainment, services, and so on. Network security is one of the most important issues at the present and the IDS becomes a necessity in computer security because crackers have been trying many tools and techniques to steal crucial information and the attacks are still increasing\(^{[1]}\). It is important to detect malicious traffic with high accuracy to avoid the damage.

When IDS detects malicious traffic, classifier is used\(^{[2]}\)--\(^{[4]}\). For the classification of IDS, two methods exist, a binary classification and a multiple classification. The binary classification method is to distinguish traffic into two categories normal and anomaly; as multiple classification method is to classify traffic into many labels. The binary classification is widely used for network traffic management of IDS to identify the network traffic into two main differences, Negative (normal traffic) and Positive (malicious traffic)\(^{[5]}\).

In the binary classification, false negative (FN) and false positive (FP) affect the classification performance, and thus we have to minimize them as low as possible. FN is rather an unacceptable false because a classifier allows malicious traffic to the network system. Since the accuracy measures the both negative and positive result of classification, the result of high accuracy could be from high FP with low FN or high FN with low FP. To evaluate FN and FP, Sensitivity and Specificity are used, respectively. High sensitivity means low FN, and high specificity is low FP. If one of them is very high, the accuracy is also high. However, it is impossible for both of them to be very high at the same time because there is a trade-off between FN and FP \(^{[6], [7]}\).

The aim of this research is to minimize FN of the binary classification so that we can obtain high sensitivity and accuracy, and we have designed a model of sequential classifiers combination to detect more positive that single classifier cannot handle. Reducing FN is more important, and therefore, each of the classifier focuses on detecting more positive. The failed detection of positive (FN) in the first classifier will be detected by the second classifier, and the failed detection of positive in the second classifier will be detected by the third classifier, and so on. After the detection of these classifiers in sequence, we obtain low FN as the final output. As a result, we have the both of high accuracy and high sensitivity. Although single classifier cannot be designed to reduce FN and FP simultaneously, our design also mitigates effect of the trade-off with the sequential combination. The proposed method is evaluated with NSL-KDD’99 dataset \(^{[8]}\), and outperforms the previous works in accuracy and FN.

The organization of this paper is following. Section 2 presents related works of the classification of intrusion detection system. In Sect. 3, the proposed method, a sequential classifiers combination approach, is described. Then evaluations and result of the experiments are presented in Sect. 4. The last section concludes this paper.

2. Related Work

Many researchers proposed different algorithms and techniques for classification of intrusion detection system, and NSL-KDD’99 dataset is widely used for their evaluations.
The parallel combination methods such as Bagging, Boosting, and Stacking [9] and voting [10] are not designed to solve the problem of trade-off between FN and FP, even though they obtain high accuracy and specificity with low sensitivity (high FN).

Jasmin Kevric, et al. [11] proposed a combining classifier based on tree algorithms. They are Random Tree, NB Tree, and C4.5; the combination of their algorithms based on the sum rule scheme. NSL-KDD Train 20 percent is used to create classifier models. They evaluated the models with two testing dataset: full test dataset and subset test dataset. Four different combinations are implemented, and the best combination is Random Tree and NB Tree with 83.9% Sensitivity, 96.2% Specificity, and 89.24% Accuracy. They obtained high accuracy and specificity with low sensitivity, the high accuracy is because of high specificity. High specificity means that many negatives are correctly classified, while low sensitivity means that much positive is not detected. Even though they obtained high accuracy with low sensitivity, means the classifier fails to detect malicious traffic.

Bayu, et al. [12] proposed a model of anomaly detection using two-level classifier ensemble. The model employs two ensemble learners (boosting and random subspace model) with particle swarm optimization (PSO-based feature selection technique). NSL-KDD dataset utilized in the experiment, the performance of the model outperforms the accuracy of 85.01% and false alarm rate (FAR) is 12.6%.

Rifkie Primarth, et al. [13] proposed a Random Forest classifier with ten different tree-size of random forest. Using a grid search to obtain the best learning parameters. The dataset NSL-KDD, UNSW-NB15, and GPRS are employed.

Ali H. Mirza [14] proposed an ensemble method with three different classifiers (Neural Network, Decision Tree, and Logistic Regression) to boost the overall performance. Weighted majority voting scheme are employed for KDD Cup’99 dataset. The experiment result shows an increase of classification accuracy.

These researches implemented combining classifiers with ensemble methods, and have improved accuracy and specificity (false alarm rate). However, they did not improve more sensitivity, the specificity is greater than sensitivity. High specificity means low FP, and low sensitivity means high FN. FN means that classifier allows malicious traffic, and the meaning of FP is that the classifier does not allow normal traffic (IDS incorrect detects normal traffic). Referring to their results with high accuracy, high specificity, and lower sensitivity, means that FN is larger than FP. Therefore, our method is to solve the problem of high FN, and our first priority is to reduce FN (high sensitivity) without reducing accuracy.

### 3. Proposed Method

#### 3.1 Design Methodology

When we design a sequential combination method for classification of IDS, we aim to decrease FN of intrusion detection without reducing the accuracy. The more malicious traffic detected (low FN), the safer network system is. Since single classifier is not enough to obtain low FN with high accuracy because of the trade-off between FN and FP. Therefore, we have designed a sequential classifiers combination model with using different algorithms for each classifier in Fig. 1.

Firstly, the classifier1 classifies all incoming network traffic data, and then positive output (Pos1 = TP1 and FP1) will be in charge of network administrator. If FP is acceptable level, our interest is in the negative output (Neg1 = TN1 and FN1) because this part includes undetected malicious traffic. The undetected malicious traffic from classifier1 is rechecked by classifier2, and the negative output from classifier2 is also classified by classifier3. This process is repeated until the last classifier. Finally, we obtain low FN (only one false negative, FN5, for combining of five classifiers because FN1 to FN4 is the input of classifier 2 to classifier 5) with high accuracy because we detect more malicious traffic. In the procedures, a role of the sequential combination is to reduce FN and the trade-off between FP and FN in single classifier.

We will be able to add new classifier to the sequential classifiers combination to reduce false negative (increase the detection of malicious traffic). We consider reducing malicious traffic to enhance the network security. The single classifier may not be reliable to classify the traffic, and then we need another different classifier to reclassify the allowed traffic by earlier classifier to reduce the possibility of access of malicious traffic. When adding new classifier, we have to select a model with low FN and FP in order to obtain the final results of prediction with lower FN. We expect that the sequential classifier combination correctly detects more positive instances for the final output of sequential combination by adding different classifiers. Then, by adding a new different classifier, the sequential classifier combination has to improve the limitations of former classifiers; even when the former classifiers cannot classify some of the traffic, the latter different classifiers should classify the unclassified traffic. When adding new classifier, we also consider avoiding high FP because FN may be reduced but new FPs occur.

#### 3.2 Optimization Policy of Classifier

With our sequential design to minimize FN, positive output should be detected with higher possibility and not be
allowed through the network while negative output passes through the network. Since we do not relay on single classifier, we employ additional classifiers in sequence to distinguish positive instance from negative output, and ensure low rate of FN. By this procedure, we can reduce malicious traffic, and at the same time the new incorrect detection is occurred (FP) by the other sequential classifiers. We pay more attention for FN because it is very harmful to allow malicious traffic causing network problem, however, FP is also annoying for general users are not able to access network and the network administrator have to handle FPs occurred. In order to reduce these inconveniences, the rate of FP should not be excessively high.

3.3 Evaluation Metrics

In binary classification, a confusion matrix shows the detail of the output as show in Table 1. With this table we can evaluate a created model and testing, and we also enable to calculate evaluation metrics of binary classification.

| Actual class | Predicted class |
|--------------|-----------------|
|              | Negative        | Positive       |
| Negative     | True Negative (TN) | False Positive (FP) |
| Positive     | False Negative (FN) | True Positive (TP) |

| Equation     | Formula |
|--------------|---------|
| Sensitivity  | $\frac{TP}{P} \times 100\%$       (1) |
| Specificity  | $\frac{TN}{N} \times 100\%$       (2) |
| Accuracy     | $\frac{TP + TN}{P + N} \times 100\%$ (3) |

$N$ and $P$ are the total number of actual negative instances, and positive instances in testing dataset, respectively. To understand how we calculate these metrics for our method, we describe in Eq. (4) to Eq. (6).

| Equation     | Formula |
|--------------|---------|
| Sensitivity  | $\frac{TP1 + TP2 + TP3 + TP4 + TP5}{P} \times 100\%$ (4) |
| Specificity  | $\frac{TN5}{N} \times 100\%$ (5) |
| Accuracy     | $\frac{(TP1 + TP2 + TP3 + TP4 + TP5) + TN5}{P + N} \times 100\%$ (6) |

Where,

$P = TP1 + TP2 + TP3 + TP4 + TP5 + FN5$ (7)

$N = TN5 + FP1 + FP2 + FP3 + FP4 + FN5$ (8)

The number 1 to 5 refers to the classifier 1 to 5.

3.4 Classification Procedure

In order to classify network traffic data with classifiers, we have three steps. The first step is creating classifier models, the second step is classification, and finally combine all classifiers results. The detail of each step described in the following subsections.

3.4.1 Training Phase

This process is to choose the dataset, which we can obtain model of classifier. The other important thing is selecting algorithms and tuning parameters to obtain effective classifier models, we mentioned in Sect. 3.4. To avoid over fitting, we apply ten folds cross validation so that the models are more accurate in practice.

3.4.2 Testing Phase

When we obtain models from training phase, the classification can start. The testing phase is the phase to prove how the learning model effectively works for actual data. The testing data is fed into classifier model to predict. We intent to reduce FN by detecting positive from each classifier. The positive outputs (Pos1 to Pos5 in Fig. 1) passed through, and the negative outputs (Neg1 to Neg4) fed into the next classifiers (C2 to C5) to reclassify respectively. Finally, we have only one FN from the last classifier while FP is from the classifiers combined in sequence. However, the FP for each classifier is not such large number comparing to FN. The prediction of the dataset is binary classification which is a process of distinguishing the data with two different types; positive (malicious traffic) and negative (normal traffic).
3.4.3 Combining Classifiers

After running sequential classifiers, we have the final output from these classifiers’ outputs. Based on the design, we can evaluate our sequential classifiers model by calculating sensitivity, specificity, and accuracy from Eq. (4)–Eq. (6).

3.5 Classifier Algorithms

In this paper, we combine various algorithms of classification tool WEKA [15] for our sequential combination method. They are Random Tree, Decision Tree, K Nearest Neighbor, Neural Network, and Naïve Bayes. We applied these algorithms to detect more malicious traffic (positive instances) and tried to set the parameters of each algorithm to obtain effective classifier models with low FN and FP in order to obtain high sensitivity and accuracy.

Random Tree: works exactly like the decision tree, one different thing is the splitting random subset of attributes. The interior node corresponds to the input attributes. The leaf node describes the label of prediction. In the experiment, the random number seed used for selecting attributes is 2, the maximum deep of the tree is unlimited, and the other parameters in WEKA are not changed.

Decision TreeJ48: is one algorithm of decision tree, it breaks down the data into smaller subsets. The root node or decision node may obtain two or more branches, the interior node shows the attribute, and the leaf note represents the decision of decision tree. We set five folds for the amount of data used for reduced-error pruning, minimum number of instances per leaf is six, and the confidence factor used for pruning is 0.25.

K Nearest Neighbor or IBK classifier: works based on their similarity of the number of the nearest instances. The distance of the nearest neighbors is used to identify the class of that instance. The decisions made by the majority vote of its neighbors. We set K value, the number of neighbors to be used, is 100.

Multilayer Perceptron (MLP): is the algorithm for neural network. MLP comprises of a network of artificial neurons (nodes), there are three types of node connected to each other: Input nodes, hidden nodes, and output nodes. The connection from node to node is adjustable. We set Learning rate as 0.3, and the momentum applied to the weights during updating is 0.2.

Naïve Bayes classifier: is probabilistic classifier based on Bayes’ theorem with the independence assumptions. It is fast and easy to build. NB is widely used because it often outperforms.

4. Evaluations and Result

The experiment is conducted on an Intel(R) Core(TM) i7 CPU 2.60GHz, 16GB memory, 64-bit Operating System, Windows 10 Pro, and WEKA 3.8 [15] is used as classification tool.

| Dataset            | No. of normal instance | No. of anomaly instance | Total  |
|--------------------|------------------------|-------------------------|--------|
| KDDTrain+.ARFF     | 67,543 (53.45%)        | 58,630 (46.55%)         | 125,973|
| KDDTrain+.20Percent.ARFF | 13,449 (33.38%)   | 11,743 (66.62%)         | 25,192 |
| KDDTest+.ARFF      | 9,711 (43.07%)         | 12,833 (56.93%)         | 22,544 |
| KDDTest-21.ARRF    | 2,152 (18.16%)         | 9,698 (81.84%)          | 11,850 |

4.1 NSL-KDD Dataset

NSL-KDD dataset is the updated version of KDD Cup’99 [8]. It does not include redundant instances in the train and test dataset, so the classifiers will not be biased towards more frequent records. This updated dataset still suffers from some of the problems discussed by McHugh [16], and is not a perfect representative of existing real networks dataset. However, it is an effective benchmark dataset for researchers to compare the different classifier algorithms. NSL-KDD solves some of the inherent problems of KDD Cup’99 dataset and is publicly available for researchers. The datasets have two training datasets. “KDDTrain+. ARFF” consist of 125,973 instances and “KDDTrain+.20Percent.ARFF” consist of 25,192 instances. Two test dataset are “KDDTest-21.ARFF” and “KDDTest+.ARFF” include 11,850 and 22,544 instances, respectively.

In our experiment, we used the original “KDDTrain+.20Percent.ARFF” and the both testing dataset of “KDDTest-21.ARFF” and “KDDTest+.ARFF” without applying feature selection before classification. The dataset consists of 42 attributes and two classes of label normal and anomaly. Table 2 shows the number of normal and anomaly instances of the dataset.

We use two files of test dataset of NSL-KDD’99, KDDTest+ (full NSL-KDD test dataset), and KDDTest-21 (subset of KDDTest+). We conducted experiment with both of them to evaluate the difference of full and subset test dataset, and also to compare our results with the previous works of the other researches.

4.2 Experiment Result

Our aim is to reduce more FN with high sensitivity and accuracy. As we know that single classifier is limited for detecting of the entire positives, therefore we propose the sequential classifiers combination method. We want to know the effectively work in case of increasing the number of classifiers. The experiment started with single classifier with low FP, and then we added more classifiers one by one. Note that we analyzed the output after combining classifiers, and found that the combination of the first three classifiers was good as more malicious (positive) are detected and a smaller number of FP is obtained. However, the improvement of sensitivity and accuracy of the fourth and fifth classifiers is
quite small, and then we stop adding more classifiers because the sixth classifier was not effective.

4.2.1 Order of Algorithms in Sequential Classifier Combination

The reasons of ordering classifiers in the sequential combination are to reduce FN, and not to produce large number of FP. The order of classifier algorithms is important because it can cause the increase of FN and FP which affect the sensitivity, specificity, and accuracy. Before selecting algorithms for each classifier, we should consider the output detail of model for each classifier to decide the sequential order of the classifiers and select the model with the number of FP. The classifier model with the smaller number of FP is placed in first because positive output (TP and FP) is not classified further in our design, only negative output (TN and FN) is reclassified by the sequential classifiers. These positive outputs are handled by network administrator, they are analyzed to distinguish false alert (or FP) and authentic alert (TP) because malicious traffics are hidden in the alerts. Therefore, we configure the order of classifier with low FP in the first place to obtain low FP in the final result because the final FP is a cumulative of FPs from the first classifier to the last classifier.

Table 3 shows the model result of each classifier algorithm of 20% training dataset (KDDTrain+20percent. ARFF). We know the accuracy of each classifier model, time taken to build model, and the detail number of TN, FN, FP and TP. The time taken to build model for Random tree, IBK and NB is less than a second, J48 model is 2.31 seconds, the larger amount of time is MLP model. According to the result of each model, we set the order of algorithms by selecting the small to large number of FP. The order of algorithms is as the following.

| Classifier | Accuracy | Time taken to build model | TN   | FN   | TP   |
|------------|----------|---------------------------|------|------|------|
| Random tree | 99.5%    | 0.31 second                | 13,396 | 72   | 11,671 |
| J48        | 99.46%   | 2.31 seconds               | 13,379 | 65   | 11,678 |
| IBK        | 97.58%   | 0.02 second                | 13,230 | 390  | 11,353 |
| MLP        | 97.08%   | 1.292.95 seconds           | 13,086 | 372  | 11,371 |
| NB         | 89.59%   | 0.16 second                | 12,272 | 1,445 | 1,177 |

4.2.2 Result of Proposed Method for KDDTest-21 Dataset

For obvious understanding of input and output of each classifier of KDDTest-21 and KDDTest+ dataset, we define as the following, refers to Fig. 1. Input1, output1 are the input and output of C1, and so on. We define the output of the C1, TN1 and FN1, are the input of the next classifier (C2), and so on.

Table 4 presents each step of the confusion matrix in the sequential combination of classifiers for KDDTest-21 dataset. The negative output of C1 (TN1 and FN1) will be the input of C2, and the negative output (TN and FN) of C2, C3 and C4 will be the input of C3, C4 and C5, respectively. We can calculate three metrics of single classifier and the combination of two, three, four and five classifiers, respectively, in Table 5.

In Table 4, consider the single classifier C1, Random tree, we have FN = 2,234 with Accuracy = 78.5%, it is not satisfied. And then we apply one more classifier (C2) in sequence, C2 reclassifies negative output (TN1 and FN1) of classifier one (see Fig. 1), it can reduce 2,234 FNs to 1,816 FNs. We repeat this process until the last classifier NB. Finally, we have final false negative from classifier five FN5 = 1,206 (it enables to reduce from FN1 to FN5 equals to 46.02% reduction). The total of time taken to classify for five classifiers is 22.28 seconds.

In Table 5, if we have one classifier Random tree, we obtain Sensitivity = 76.96%, Specificity = 85.45%, and Accuracy = 78.5%. It is not acceptable; it is still low
Table 5  Sensitivity, Specificity, and Accuracy of single and combinational sequential classifiers for KDDTest-21 dataset (testing).

| Combination of classifiers | Metrics (%) |
|----------------------------|-------------|
| Single classifier (1C): Random tree | Sensitivity = 76.96  
Specificity = 85.45  
Accuracy = 78.5 |
| Combine two classifiers (2Cs): Random tree + J48 | Sensitivity = 81.27  
Specificity = 84.1  
Accuracy = 81.78 |
| Combine three classifiers (3Cs): Random tree + J48 + IBK | Sensitivity = 85.72  
Specificity = 81.27  
Accuracy = 84.91 |
| Combine four classifiers (4Cs): Random tree + J48 + IBK + MLP | Sensitivity = 86.49  
Specificity = 61.8  
Accuracy = 82 |
| Combine five classifiers (5Cs): Random tree + J48 + IBK + MLP + NB | Sensitivity = 87.56  
Specificity = 58.96  
Accuracy = 82.37 |

Fig. 2  Sensitivity, Specificity, Accuracy of single classifier and combinational classifiers for KDDTest-21 dataset.

because of high FN = 2,234 and FP = 313 (in Table 4). To improve the result of single classifier, we add C2 to reclassify the negative output of C1 (TN1 and FN1); after combination of C1 and C2 we obtain better sensitivity and accuracy as 81.27%, 81.78%, respectively, while specificity little decreases 1.35% (from C1 85.45 % to C2 = 84.1%). After that, we continue combining the third, fourth, and fifth classifiers, the sensitivity gradually increases. The accuracy also increases until C4’s accuracy drops because of TP4 is less than FP4, but sensitivity still increases because of dropping of FN.

We compare one single classifier Random tree and combination sequential classifiers (Random tree + J48 + IBK + MLP + NB), the combination of five sequential classifiers is better than single classifier in term of sensitivity and accuracy.

Figure 2 shows that sensitivity gradually increases after combining two, three, four and five classifiers. This means that FN is reduced when we combine more classifiers. Since Sensitivity and Specificity are trade-off, specificity decreases from single classifier to combining five sequential classifiers. However, the overall accuracy is improved from the single classifier to five combining sequential classifiers. The results show that a combination of five classifiers is better than four, three, two and single classifier in term of sensitivity. If we focus on the accuracy, the number of best combination is three classifiers. The accuracy of four classifiers combination drops because TP4 is less than FP4. Even though the accuracy of four classifiers drops but more positive is detected, means more FN is reduced. Since this research prioritizes FN, we continue the combination to the last classifier to reduce more FN.

4.2.3 Result of Proposed Method for KDDTest+ Dataset

Next evaluations are for the full NSL-KDD test dataset of KDDTest+. Table 6 shows the detail of classifier output for each five classifiers. FN1 = 2,234, FP1 = 368. Then C2 reduces FN1 to 1,816 (decreased by number of TP2 = 418). Next C3 reduces FN2 to 1,816, C4 additionally reduces FN3, and the last C5 farther reduces FN4 by 104. Therefore, FN1 = 2,234 for single classifier can be reduced to 1,206 by the five sequential combination of the classifiers. The time taken to classify of Random tree, J48, and NB is less than a second, IBK classifier took larger time compared to the rest, 43.55 seconds. The time taken to classify for these five classifiers is 46.23 seconds.

Table 7 and Fig. 3 show the sensitivity, specificity, and accuracy of single classifier and each combination of classifiers for KDDTest+ dataset. The sensitivity, specificity, and accuracy of single classifier Random tree are 82.59%, 96.21%, and 88.45%, respectively. To increase the percentage of sensitivity and accuracy, we combine two, three, four, and five classifiers in sequence. More positive is detected (FN reduces) and little negative is also detected (FP
Table 7  Sensitivity, Specificity, and Accuracy of single and combinational sequential classifiers for KDDTest+ dataset (testing).

| Combination of classifiers | Metrics (%) |
|----------------------------|-------------|
| Single classifier (1C): Random tree | Sensitivity = 82.59  
Specificity = 96.21  
Accuracy = 88.45 |
| Combine two classifiers (2Cs): Random tree + J48 | Sensitivity = 85.84  
Specificity = 95.59  
Accuracy = 90.17 |
| Combine three classifiers (3Cs): Random tree + J48 + IBK | Sensitivity = 89.21  
Specificity = 95.11  
Accuracy = 91.75 |
| Combine four classifiers (4Cs): Random tree + J48 + IBK + MLP | Sensitivity = 88.79  
Specificity = 90.79  
Accuracy = 90.22 |
| Combine five classifiers (5Cs): Random tree + J48 + IBK + MLP + NB | Sensitivity = 90.6  
Specificity = 90.08  
Accuracy = 90.37 |

Fig. 3  Sensitivity, Specificity, Accuracy of single classifier and combinational classifiers for KDDTest+ dataset.

Table 8  Confusion matrix of prediction of single classifier and its accuracy for KDDTest-21 and KDDTest+ dataset (testing).

| KDDTest-21 dataset | KDDTest+ dataset |
|--------------------|------------------|
| Random tree Accuracy = 78.5%  
Time taken to classify: 0.23 second | Random tree Accuracy = 88.45%  
Time taken to classify: 0.52 second |
| TN= 1,839  
FP= 313 | TN= 9,343  
FP= 368 |
| FN= 2,234  
TP= 7,466 | FN= 2,234  
TP= 10,599 |
| J48 Accuracy = 63.29%  
Time taken to classify: 0.21 second | J48 Accuracy = 80.7%  
Time taken to classify: 0.48 second |
| TN= 1,889  
FP= 263 | TN= 9,446  
FP= 265 |
| FN= 4,086  
TP= 5,612 | FN= 4,086  
TP= 8,747 |
| IBK Accuracy = 57.6%  
Time taken to classify: 50.26 seconds | IBK Accuracy = 77.65%  
Time taken to classify: 84.59 seconds |
| TN= 1,834  
FP= 318 | TN= 9,379  
FP= 332 |
| FN= 4,706  
TP= 4,992 | FN= 4,706  
TP= 8,127 |
| MLP Accuracy = 53.4%  
Time taken to classify: 0.9 second | MLP Accuracy = 75.44%  
Time taken to classify: 1.65 seconds |
| TN= 1,426  
FP= 726 | TN= 8,971  
FP= 740 |
| FN= 4,796  
TP= 4,902 | FN= 4,796  
TP= 8,037 |
| NB Accuracy = 55.77%  
Time taken to classify: 0.85 second | NB Accuracy = 76.56%  
Time taken to classify: 1.34 seconds |
| TN= 1,460  
FP= 692 | TN= 9,010  
FP= 701 |
| FN= 4,549  
TP= 5,149 | FN= 4,582  
TP= 8,251 |

FP4. At the final point 5Cs, sensitivity still keeps increasing and the accuracy is little improved because FN is decreased (TP5 = 104 is greater than FP5 = 69) in Table 6. The final result of five sequential classifiers combination (5Cs), we can reduce FN more than the increasing of FP generated by five sequential classifiers. In conclusion, the sequential combination is improved in term of sensitivity and accuracy.

4.2.4 Results of Single Classifier for KDDTest-21 and KDDTest+ Dataset

To confirm the sequential classifiers combination of our proposed method is better than single classifier, we conducted experiments of the single classifier. Table 8 shows detail of the single classifier output, accuracy, and the time taken to classify for KDDTest-21 and KDDTest+ dataset. The accuracies of the single classifier for both testing dataset are not good because of high FN which causes low accuracy and sensitivity. We also enable to calculate the sensitivity, and specificity of single classifier, and they are described in Table 9.

In Table 9, the results for KDDTest-21 dataset are low because of high FN and FP, while the results of KDDTest+ dataset obtain high specificity for Random tree, J48, and IBK classifiers that means FP is low. However, the other two metrics, sensitivity and accuracy become low because of high FN.

For comparison of the proposed method shows in Table 5 for KDDTest-21 dataset and Table 7 for KDDTest+ dataset with the single classifier in Table 9 for both testing dataset, the proposed method outperforms in term of...
Table 9: Sensitivity, Specificity, and Accuracy of Single Classifier for KDDTest-21 and KDDTest+ Dataset (Testing).

| Single Classifier | KDDTest-21 Dataset | KDDTest+ Dataset |
|-------------------|-------------------|------------------|
|                   | Metrics (%)       | Metrics (%)      |
| **Random tree**   | Sensitivity = 76.96 | Sensitivity = 82.59 |
|                   | Specificity = 85.45 | Specificity = 96.21 |
|                   | Accuracy = 78.5    | Accuracy = 88.45 |
| **J48**           | Sensitivity = 57.86 | Sensitivity = 68.16 |
|                   | Specificity = 87.77 | Specificity = 97.27 |
|                   | Accuracy = 63.29   | Accuracy = 80.7  |
| **IBK**           | Sensitivity = 51.47 | Sensitivity = 63.32 |
|                   | Specificity = 85.22 | Specificity = 96.58 |
|                   | Accuracy = 57.6    | Accuracy = 77.65 |
| **MLP**           | Sensitivity = 50.54 | Sensitivity = 62.62 |
|                   | Specificity = 66.26 | Specificity = 92.37 |
|                   | Accuracy = 53.4    | Accuracy = 75.44 |
| **NB**            | Sensitivity = 53.09 | Sensitivity = 64.29 |
|                   | Specificity = 67.84 | Specificity = 92.78 |
|                   | Accuracy = 55.77   | Accuracy = 76.56 |

Table 10: Comparison of Overall Accuracy of KDDTest+ and KDDTest-21.

| Methods                   | KDDTest+ (%) | KDDTest-21 (%) |
|---------------------------|--------------|----------------|
| Ensemble (boosting and random subset) [12] | 85.01 | - |
| NBTree [16]               | 82.02 | 66.16 |
| Decision tree [18]        | 80.14 | 80.14 |
| Fuzzy [19]                | 82.74 | - |
| Random tree + NBTree [11] | 89.24 | 80.9 |
| Our method                | 90.6 | 82.37 |

4.3 Discussion

4.3.1 Comparison with Related Works

Table 10 shows the comparison of overall accuracy for KDDTest+ and KDDTest-21 dataset of the previous works. Our method obtained better overall accuracy for both KDDTest+ and KDDTest-21 because their methods, single classifier [16], [18], [19], and combination of classifiers [11], [12] did not reduce more FN. Moreover, they focused only improvement of accuracy without considering sensitivity (the model can detect positive), while our method focused to minimize FN. We can improve detection rate, and not too much FP, that is why our method obtained better results.

Next, we will compare the sensitivity, specificity, and accuracy of KDDTest+ in Table 11 because KDDTest+ is full test dataset. In Table 11, our method is better in term of Sensitivity (Lower FN) by 6.7%, and a little better 1.13% for overall Accuracy compared to [11]. At the same time, our specificity is worse than [11] by 6.12%. The overall our method is better because our method obtained both high sensitivity and accuracy, FN (sensitivity) is more important than FP (specificity).

Methods in Refs. [16], [18] and [19] applied only one classifier and are not enough to improve accuracy because various positive and negative instances are not correctly classified. Especially, one classifier is limited to obtain high accuracy by detecting more positive instance. These studies did not evaluate sensitivity and specificity, they evaluated only accuracy. However, we can say that their sensitivities and specificities are low based on their low accuracies. References [11], [12] employed two and three different classifiers but combination of two classifiers is better than three. They did not focus to minimize FN, they focused on overall accuracy, based on their results with high specificity and low sensitivity may not be acceptable in the real practice.

4.3.2 Processing Delay of the Proposed Method

The proposed method contains five classifier algorithms as Random tree, J48, IBK, MLP, and NB. The processing time of each algorithm is different. The time taken to build model in training phase, and the time taken to classify network traffic for both testing dataset in testing phase show in the Table 3, Table 4 and Table 6 respectively. The time taken to build model for MLP model is larger than the other models that is 1,292.95 seconds. In case of the training dataset is getting big, we can parallel build models in different machines for training phase. For the time taken to classify in testing phase is small because the input for next classifier is only negative output from the former classifier and it takes short time to classify. The total time taken to classify of our method for KDDTest-21 and KDDTest+ dataset are 22.28 seconds and 46.23 seconds respectively.

The time taken to classify for single classifier for both testing dataset shows in Table 8. The time taken to classify of four classifiers, Random tree, J48, MLP, and NB is very small, except IBK classifier. Comparing the time taken to classify of single classifier IBK and the proposed method for the both testing dataset, the time of single classifier is larger than that of the proposed method with five classifiers because the single classifier classifies all the testing dataset while the IBK classifier for the proposed method
is in the third place, and classifies only negative output from the second classifier. Therefore, we can say that the proposed method does not affect the processing delay in testing phase.

4.3.3 Implementation of the Proposed Method in Actual Use Case

We design a sequential classifiers combination method of five different classifiers to reduce FN for IDS. The experimental results of our proposed method using NSL KDD’99 dataset were almost the same number of positives and negatives. Even though, we can reduce the number of final FN from single classifier (C1) by our method almost half from 2,234 to 1,206, while the number of final FP is increased (sum of five classifiers’ FP = 963).

Since the proposed method has a trade-off between the number of classifiers and the number of FNs, a case of the less number of classifiers results the larger number of FNs with the smaller number of FPs. This is because that single classifier produces the larger number of FNs with the smaller number of FPs referring to the experiment results in Table 4 and Table 6. In contrast, the larger number of classifiers cases result the lower number of FNs with the larger number of FPs because FPs is produced by the latter classifiers and then the final FP is cumulative of all the classifiers’ FP.

The requirements in an actual use case of IDS are no false alert (no FP) and no intrusion (no FN) in order to manage network efficiently. Since almost the traffic is normal traffic and very low number of malicious traffic in practice, we have to consider not only FN but also FP. When implementing the proposed method, the network administrator can determine the number of the sequential classifiers depending on the situation. As an example, for the normal situation, traffic includes the small number of malicious traffic and the administrator will be able to reduce the number of the sequential classifiers so that the sequential classifiers do not generate the large number of FPs because the administrator do not have enough time to handle all FPs in the positive outputs. On the other hand, in case of happening attacks or intrusions, the administrator will increase the number of sequential classifiers to detect more intrusion and analyze alerts. The detail analysis is effective to plan to prevent such a situation in the future.

5. Conclusion

In this paper, we designed a sequential classifiers combination method with five different algorithms to minimize false negative and the proposed method was evaluated with NSL KDD’99 dataset. Each classifier can reduce FN with considering acceptable FP. Therefore, we obtain the final classification output with high accuracy with the low false negative. Our results showed that combining five sequential classifiers is better than single classifier in term of sensitivity and accuracy because false negatives were additionally detected as the next sequential classifiers.

In the future work, we will apply feature selection to our dataset before training and testing, and consider the ratio of four main types of attack in order to improve performance. We will also modify our combining classifier design to obtain better result.

References

[1] G.S. Rao, P.N. Kumar, P. Swetha, and G. BhanuKiran, “Security assessment of computer networks- an ethical hacker’s perspective,” Proc. International conference on computing and communication technologies, pp.1–5, 2014.

[2] R.C. Aygul and A.G. Yavuz, “Network anomaly detection with stochastically improved autoencoder based models,” Proc. IEEE 4th international conference on cyber security and cloud computing, pp.193–198, 2017.

[3] S. Choudhury and A. Bhowal, “Comparative analysis of Machine learning algorithms along with classifiers for network intrusion detection,” Proc. International conference on smart technologies and management for computing, communication, controls, energy and materials (ICSTM), pp.89–95, 2015.

[4] B.A. Tama and K.-H. Rhee, “HFSTE: Hybrid feature selections and tree-based classifiers ensemble for intrusion detection system,” IIECE Trans. Inf. & Syst., vol.E100-D, no.8, vol.214, pp.1729–1737, 2017.

[5] T. Liu, H. Jiang, and J. Wen, “A Binary Classification Method Based on Class Space Model,” Proc. International Conference on Intelligent Computing and Integrated Systems, pp.807–809, 2010.

[6] S. Yang, T. Khot, K. Kersting, G. Kunapuli, K. Hauser, and S. Natarajan, “Learning from imbalanced data in relational domains: a soft margin approach,” Proc. IEEE International conference on Data Mining, pp.1085–1090, 2014.

[7] S. Shibaguchi, Y. Nakayama, and K. Okada, “Worm path identification using visualization system,” Proc. International conference on computational science and engineering, pp.498–503, 2009.

[8] M. Tavallaee, E. Bagheri, W. Lu, and A.A. Ghorbani, “A detailed analysis of the KDD Cup 99 Data set,” Proc. 2009 IEEE Symposium on Computational Intelligence in Security and Defense Applications (CISDA), pp.1–6, 2009.

[9] A. Verma and S. Mehta, “A comparative study of ensemble learning methods for classification in bioinformatics,” Proc. 7th International conference on cloud computing, data science & engineering, pp.155–158, 2017.

[10] J. Kittler, M. Hatef, R.P.W. Duin, and J. Matas, “On combining classifiers,” IEEE Trans. Pattern Anal. Mach. Intell., pp.226–239, 1998.

[11] J. Kevcic, S. Jukic, and A. Subasi, “An effective combining classifier approach using tree algorithms for network intrusion detection,” Neural comput & Appl, vol.28, Sup. 1, pp.1051–1058, 2017.

[12] B.A. Tama, A.S. Patil, and K.-H. Rhee, “An improved model of anomaly detection using two-level classifier ensemble,” Proc. 12th Asia Joint Conference on Information security, pp.1–4, 2017.

[13] R. Primartha and B.A. Tama, “Anomaly detection using random forest: a performance revisited,” Proc. International conference on data and software engineering (ICoDSE), pp.1–6, 2017.

[14] A.H. Mirza, “Computer network intrusion detection using various classifiers and ensemble learning,” Proc. 26th Signal processing and communications applications conference (SIU), pp.1–4, 2018.

[15] WEA, https://www.cs.waikato.ac.nz/ml/weka/.

[16] J. Mchugh, “Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory,” ACM Transactions on information and system security, vol.3, no.4, pp.262–293, 2000.

[17] S. Umadevi and K.S.J. Marseline, “A Survey on data mining classification algorithms,” Proc. International conference on signal processing and communication, pp.264–268, 2017.
[18] M. Mohammadi, B. Raahemi, A. Akbari, and B. Nassersharif, “Class dependent feature transformation for intrusion detection systems,” Proc. 19th Iranian Conference on Electrical Engineering (ICEE), p.1, 2011.

[19] P. Kromer, J. Platos, V. Snasel, and A. Abraham, “Fuzzy classification by evolutionary algorithms,” Proc. IEEE international conference on systems, man, and cybernetics, pp.313–318, 2011.

Phetlasy Sornxayya received M.E. degree from the Graduate School of Information Systems, the University of Electro-Communications, Tokyo, Japan, in 2014. He is currently a Ph.D. candidate at the same university. His current research is to distinguish normal and anomalous traffic of IDS dataset.

Satoshi Ohzahata received M.E., and D.E. degrees from the University of Tsukuba in 1998, 2000 and 2003, respectively. He was a Research Associate, Department of Computer, Information & Communication Sciences at Tokyo University Agriculture and Technology from 2003 to 2007, and was an assistant professor of the same university from 2007 to 2009. Since 2009, he has been an associate professor at Graduate School of Information Systems, the University of Electro-Communications. His interests are mobile ad hoc networks, the Internet architecture in mobile environments and Internet traffic measurement. He is a member of IEEE, ACM and IPSJ.

Celimuge Wu received M.E. degree from Beijing Institute of Technology, Beijing, China, in 2006, and the Ph.D. degree from the University of Electro-Communications, Tokyo, Japan, in 2010. He is currently an associate professor of the Graduate School of Information Systems, the University of Electro-Communications. His current research interests include mobile ad hoc networks, networking architectures and protocols.

Toshihiko Kato received B.E., M.E., and Dr. Eng. degrees electrical engineering from the University of Tokyo, in 1978, 1980 and 1983, respectively. He joined KDD in 1983 and worked in the field of communication protocols of OSI and Internet until 2002. From 1987 to 1988, he was a visiting scientist at Carnegie Mellon University. He is now a professor of the Graduate School of Information Systems in the University of Electro-Communications in Tokyo, Japan. His current research interests include protocol for mobile Internet, high speed Internet and ad hoc network.