*Advances in Cyber Physical Social Systems (CPSS) - Research Article*

International Journal of Distributed Sensor Networks

*International Journal of Distributed Sensor Networks*
2019, Vol. 15(6)

DOI: 10.1177/1550147719846052
journals.sagepub.com/home/dsn
SAGE

# Semi-supervised tri-Adaboost algorithm for network intrusion detection

**Yali Yuan[1], Liuwei Huo[2], Yachao Yuan[3] and Zhixiao Wang[1,4,5]**

**Abstract**

Network intrusion detection is a relatively mature research topic, but one that remains challenging particular as technologies and threat landscape evolve. Here, a semi-supervised tri-Adaboost (STA) algorithm is proposed. In the algorithm, three different Adaboost algorithms are used as the weak classifiers (both for continuous and categorical data), constituting the decision stumps in the tri-training method. In addition, the chi-square method is used to reduce the dimension of feature and improve computational efficiency. We then conduct extensive numerical studies using different training and testing samples in the KDDcup99 dataset and discover the flows demonstrated that (1) high accuracy can be obtained using a training dataset which consists of a small number of labeled and a large number of unlabeled samples. (2) The algorithm proposed is reproducible and consistent over different runs. (3) The proposed algorithm outperforms other existing learning algorithms, even with only a small amount of labeled data in the training phase. (4) The proposed algorithm has a short execution time and a low false positive rate, while providing a desirable detection rate.

**Keywords**

Network intrusion detection, tri-training method, Adaboost algorithms, chi-square method, execution time, detection rate

Date received: 18 January 2019; accepted: 1 April 2019

Handling Editor: Shancang Li

## Introduction

In this age of connectivity, where services, utilities, and the majority of everyday tasks are reliant on the Internet, vast amounts of personal data are stored "in the cloud."[1] Network attacks that impact the availability or the confidentiality of these services may result in significant losses.[1] Network intrusion detection systems (IDSs) are often used for monitoring, detecting, and analyzing events which flagged as violating network security policies.[2] In general, IDSs are mainly categorized into two categories: host-based IDSs and network-based IDSs.[3,4] Based on that, the detection methods in IDSs can be classified into two groups: misuse-based IDSs and anomaly-based IDSs. For misuse-based IDSs, the rules of them often defined by domain experts. Herein, Snort[5] and Bro[6] are two typical examples. However, for anomaly-based IDSs, they require to learn

[1]Telematics Group, Institute of Computer Science, University of Göttingen, Göttingen, Germany
[2]School of Computer Science and Engineering, Northeastern University, Shenyang, China
[3]Smart Mobility Research Group, Chair of Information Management, Faculty of Economic Sciences, University of Göttingen, Göttingen, Germany
[4]School of Computer Science and Engineering, Xi'an University of Technology, Xi'an, China
[5]School of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an, China

**Corresponding author:**
Zhixiao Wang, School of Computer Science and Engineering, Xi'an University of Technology, No.5, Jinhua South Road, Xi'an 710048, China.
Email: wangzhx@xaut.edu.cn

the information of all normal types before detecting the abnormal behaviors.[7]

Machine learning has shown to be sufficiently robust in detecting malicious activities and network threats. In supervised learning, the method is trained using labeled dataset in order to make predictions on new (unlabeled) data.[8] Plenty of supervised machine learning methods including support vector machine (SVM),[9] neural network (NN),[10] Bayesian algorithm[10] as well as clustering methods[2] have been employed in IDSs for defense against intrusions. However, due to the constant evolution of today's network traffic, the majority of the recently obtained and stored data are unlabeled in practice. It is also time-consuming and expensive and in some cases, impractical, to obtain a large amount of labeled dataset to improve the detection accuracy, particularly in our data-driven society. Unsupervised learning methods can be trained using unlabeled data. The clustering is one of the most commonly used unsupervised learning methods.[11] In clustering algorithm, according to the data similarity, a set of data is divided into different groups. Then, different labels are assigned to each group. Samples in the same cluster usually have the similar characteristics. The disadvantage of unsupervised methods is the artificial classification of samples, which has lower detection efficiency and accuracy. To overcome these challenges, semi-supervised learning algorithms designed to leverage unlabeled dataset and labeled ones are becoming increasingly popular.

Thus, the focus of this article is not only to decrease the intrusion detection error rate but also to find a model which is capable of incorporating new data with a good generalization ability. Specifically, we propose a semi-supervised intrusion detection system (SS-IDS) by combining tri-training[12] with three different Adaboost algorithms. This combination scheme allows us to minimize the number of false positive, reduce the time consumption, and increase the detection accuracy of the system. Many simulations with different parameters are performed to prove the efficieny of the proposed algorithm. The change of the amount of the labeled and unlabeled samples could have an impact on the experimental results. KDDcup99,[13] the benchmark dataset, is employed in the experiments to test the performance of the proposed algorithm.

This article is structured as: Section "Related work" makes a reviews about the intrusion detection. In section "Knowledge background," we make an analysis about the different Adaboost methods, tri-training approach, and chi-square algorithm. The intrusion detection model for the proposed STA algorithm is presented in section "Proposed network intrusion detection algorithm." Evaluation results are shown in section "Evaluations." Finally, in section "Conclusion," the conclusion and next research work are presented.

## Related work

Machine learning methods are popular to be utilized in complement existing IDSs to enhance the detection accuracy at low false positive.[1] However, the current research on the semi-supervised IDSs is limited. Semi-supervised learning is a combination of supervised and unsupervised learning, which need a small amount of labeled samples and with a large amount of unlabeled samples to train the model.[14] However, obtaining the labeled data is expensive and consuming time, in some cases requires the expert efforts of many domain. In addition, the majority of the new data are unlabeled in practice. This is particularly true given the volume, variety, velocity, and veracity of today's dataset (i.e. 4 V's of big data). Thus, it is not surprising that attention has been diverted to exploring the utilization of semi-supervised IDSs such as self-training, co-training, tri-training, cluster-based.

Mao et al.[15] utilized the unlabeled data and multi-view data with a framework of co-training method to improve the intrusion detection accuracy. Moreover, an active learning framework was employed to identify the unlabeled data and gave the suitable label to them. The experiment results proved that compared to supervised learning method which using the unlabeled data alone, the proposed semi-supervised learning method reduced the false alarm rate by leveraging the labeled and unlabeled data. Two semi-supervised learning methods including spectral graph transduction and Gauss random fields were used in Chen et al.[16] to detect the unknown intrusions. In addition, a novel semi-supervised approach for clustering, MPCK-means, was proposed to improve the IDSs performance. Experiment results suggested that MPCK-means has an improved performance compared to previous methods. Self-training methods, a semi-supervised learning method, was used in Pachghare et al.[17] to solve the intrusion detection problems of IDSs. The performance of supervised methods was improved in this study. Specifically, supervised methods were trained using labeled data. Then, the test data with unlabeled data were labeled using trained supervised methods. After that, some of the test data filtered using a method based on self-predictions for their self-teaching methods were added to the training data. All steps were repeated until it produced a good detection result. A semi-supervised method was also proposed in Chiu et al.[18] Based on a great number of unlabeled samples, more information is generated by the two-teachers-one-student (2T1S). The false alarms were reduced successfully with connection information, while only a few labeled data were available.

In Li et al.,[19] in order to increase the detection accuracy of IDSs, a semi-supervised SVM method was proposed to improve the performance of anonymous detection system. Tri-training and SVMs were combined to have a better classification accuracy. KDDcup99 dataset was used to prove the performance

of their proposed algorithm. Although this algorithm can achieve a low classification error, it had a very high cost due to the computation cost of SVMs, which results in few practical applications. In Ashfaq et al.,[20] a fuzziness-based semi-supervised learning approach (combining both labeled and unlabeled data are used for anomalous detection) was used. The authors obtain a fuzzy membership vector by training a single hidden layer feed-forward neural network and then utilize the fuzzy quantity to classify the unlabeled samples. In Yuan et al.,[21] two layers multi-class detection method including the C5.0 method and the Naive Bayes algorithm is presented for adaptive network intrusion detection. Simulation results proved that the proposed method can achieve a good performance with the imbalanced KDDcup99 dataset. Haipeng Yao et al.[22] proposed a multi-level semi-supervised intrusion detection model to solve the imbalance and non-identical distribution problems of KDDcup99 dataset. In Camacho et al.,[23] a semi-supervised algorithm was proposed based on multivariate statistical network monitoring approach and partial least squares. It combined the advantages of supervised learning and unsupervised learning strategies. Experiment results based on the real traffic proved the practical applicability of the proposed approach.

## Knowledge background

### Adaboost approaches

Adaboost, "Adaptive Boosting," is a boosting method that builds a highly accurate method by combining multiple simple ones. For our proposed method, we use three well-known Adaboost approaches, namely, discrete Adaboost approach, real Adaboost algorithm, and gentle Adaboost method.

The output of the weak method $h(x)$ in the discrete Adaboost approach, as defined by Yoav Freund and Robert Schapire,[24] is binary, that is, $h(x) \in \{+1, -1\}$. The real Adaboost algorithm[25] uses weak methods, which return a class probability estimate $h(x) \in [0, 1]$ to update the additive logistic model, rather than the classifications themselves. Based on the real Adaboost approach, gentle Adaboost algorithm was proposed by Viola and Jones.[26] This algorithm uses the difference of the conditional class probabilities for the given value of the features to construct the updated values. In the gentle Adaboost approach, the weak method functions are updated by $h(x) = P(y = 1|x) - P(y = -1|x)$, while real Adaboost algorithm is defined by half the log ratio, $h(x) = \frac{1}{2} log \frac{P(y = 1|x)}{P(y = -1|x)}$.

The main difference between gentle Adaboost method and real Adaboost algorithm is the way they use the estimates of the weighted class probabilities.

### Generalized tri-training algorithm

The tri-training algorithm that uses three learners was proposed by Zhou and Li.[12] For consistency, we will use the same terminology and variable names as the original paper.

Let $L$ be one labeled dataset and $U$ be one labeled dataset. The three different methods $h_1$, $h_2$, and $h_3$ can be trained using the labeled dataset and unlabeled dataset.

In the training phase, three methods are obtained by applying the same learning algorithm to three bootstrap samples of the labeled dataset. Then, one unlabeled data $x$ from $U$ was labeled by two of the learners, for example, $h_1$, $h_2$ are chosen first. If both $h_1$ and $h_2$ agree on the label of $x$, both the label and the corresponding sample are added in the new labeled dataset $L_i$. $L_i$, along with the original dataset $L$, is utilized to re-train the third method $h_3$. In a similar way, all the methods will be re-trained based on the unlabeled dataset until all the methods do not change anymore. To explicitly measure, the label confidence of any learner is not required in this approach; thus, it is easy to use even without any sufficient and redundant view. In the testing phase, when an unknown sample comes, it will be labeled according to the majority rules based on the three learners. Moreover, with the amount of the newly labeled samples increasing, the classification noise rate will be compensated.

### Feature selection

In IDSs, feature selection and ranking have been contributed a lot to improve the IDSs performance. In the feature selection, the minimum number of features or attributes is chose to denote the whole dataset precisely.[27] As for network intrusion detection, there are many features, some of which are not useful in improving the detection performance. The detection accuracy can be enhanced by removing the useless features, which can also reduce the time cost significantly.

Chi-square is a widely used metric, which evaluates the usefulness/relevance of a feature by calculating the value of the chi-square statistic with respect to different classes (types). Table 1 presents the two-way contingency of feature $f$ and a class $c$,[28] where # denotes the number of the variable and ¬ expresses that it does not happen. The feature-goodness measure is given in equations (1) and (2).[29]

$$N = A + B + C + D \tag{1}$$

$$X_2 = \frac{N \times (AD - CB)_2}{(A + C) \times (B + D) \times (A + B) \times (C + D)} \tag{2}$$

where $N$ is the number of all samples, $A$ is the occurrence number of $f$ and $c$ co-occuring, and $D$ expresses

**Table 1.** Chi-square statistical two-way contingency table.

| | |
|---|---|
| $A = \#(f, c)$ | $C = \#(\neg f, c)$ |
| $B = \#(f, \neg c)$ | $D = \#(\neg f, \neg c)$ |

the times when $f$ and $c$ do not occur at the same time. $C$ is the number of the occurrence of $c$ without $f$, while $B$ is the number of times $f$ appearing without $c$.

## Proposed network intrusion detection algorithm

We illustrate the detailed implementation of our proposed STA algorithm in this section, including the construction of decision stumps and weak methods. Based on Yuan et al.,[30] STA algorithm gives more detail evaluation and better performance. In Yuan et al.,[30] tri-training algorithm with Adaboost method was first proposed. However, this article only gave a simple introduction and did not use the feature selection algorithm, leading to a high computation cost.

### *Implementation of weak methods*

In many cases, some of the learning methods, such as SVMs, neural networks, K-nearest neighbor, and Naive Bayes, are used as weak methods. In our proposed algorithm, the decision stump will be utilized as sub-weak-method and several sub-weak-methods are combined as the weak method of Adaboost algorithm. Decision stump is defined by one-node tree with two leaves.[31] The learning of the decision tree continues until the most informative feature is selected. In the input dataset, one feature is corresponding to one decision stump. Our choice of a decision stump is as follows:

1. It is simple and fast. In the testing dataset, each decision tree corresponding to one sample only needs one comparison to get the results.
2. Both continuous and categorical features can be implemented conveniently.

*Implementation of sub-weak-method based on categorical features.* Let $f$ be the feature vector. A categorical feature $f_j$, $j \in \{1, 2, \ldots, M\}$ is defined with the finite discrete values, where $M$ is given to define the total number of features. We use decision stump to cut the values of each feature $f_j$ into two non-overlapping subsets $C_{p_1}^{f_j}$ and $C_{p_2}^{f_j}$. Then, the decision stump can be defined as equation (3). The decision stump of each categorical feature with the smallest false classification rate for normal and attack samples will be selected as one sub-weak-method.

$$h_f(f) = \begin{cases} +1 & f_j \in C_{p_1}^{f_j} \\ -1 & f_j \in C_{p_2}^{f_j} \end{cases} \tag{3}$$

where $C_{p_1}^{f_j}$ and $C_{p_2}^{f_j}$ are obtained by the combination method based on the values of each feature.

*Implementation of sub-weak-method based on continuous features.* Provide that $f_j$ is a continuous feature and $\mu$ is a segmentation value. The construction of the decision stump of continuous features is shown as equation (4). The decision stump of each continuous feature with the smallest error rate for both normal and attack samples will be selected as one sub-weak-method.

$$h_f(f) = \begin{cases} +1 & f_j \leqslant \mu \\ -1 & f_j > \mu \end{cases} \tag{4}$$

In the above equation, $\mu$ is obtained by averaging the neighbor values of the feature $f$.

*Implementation of weak methods.* We integrate all obtained sub-weak-methods as one weak method. The weak method based on categorical features and continuous features is described as equation (5).

$$h(t) = \frac{1}{M} \sum_{i=1}^{M} h_f(f) \tag{5}$$

### *Implementation of STA algorithm*

The proposed STA algorithm in this article is designed to improve the performance of the anomaly detection, using a massive unlabeled dataset. We generalize the tri-training approach with three different Adaboost algorithms (i.e. discrete Adaboost, real Adaboost, and gentle Adaboost) instead of the bootstrap samples to create the diversity. Although these three Adaboost algorithms are different, they are easily integrated due to the similarity of their input and output form. The flow chart of STA algorithm is given in Figure 1.

## Evaluations

### *Dataset analysis*

In the experiment, the Knowledge Discovery and Data Mining CUP 1999 dataset[13] is used to evaluate our proposed algorithm. This dataset, developed by the Defense Advanced Research Projects Agency, has been widely used in the intrusion detection literature.[32]

The advantages of this dataset are given as follows:

1. Many intrusion detection algorithms used it to evaluate their performance and describe it as a reliable benchmark dataset.

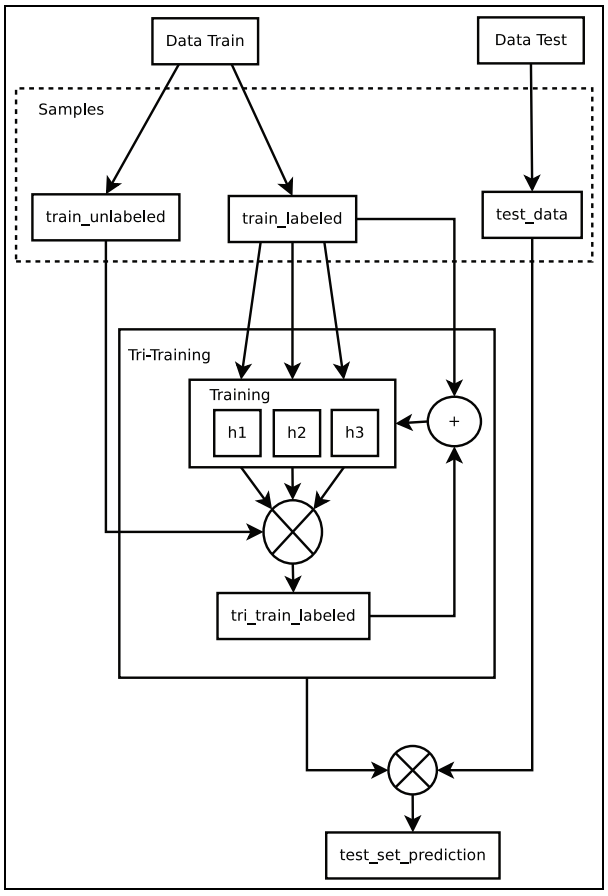


**Figure 1.** Flow chart of proposed STA algorithm.

2. Each TCP/IP connection has a predefined label, which is normal or a specific type of an attack, in this dataset. Since the dataset is labeled, it is easy for us to verify the efficiency of our proposed algorithm.
3. It is freely available online, so it can be accessed by anyone.

This dataset includes four general types of attack:

1. Denial-of-service attack (DoS): Attacker, as its name implies, tries to make some resources, such as computing or memory, very busy or full. In this situation, users cannot access to the online resources.
2. Remote to local attack (R2L): Attacker pretends to be a legal user on the victim machine by exploiting some vulnerabilities to get the access right, and in fact, it does not have an account to that machine.
3. User to root attack (U2R): Attacker attempts to be the super user and uses its privileges on the victim machine; however, in fact, it only has the local access account.
4. Probing attack (PROBE): Attacker tries to access the information related to the target host.

Table 2 illustrates the proportions of the attack types with all contained specific attack types in the training data. The portions of specific attack types in testing data are described in Table 3. It is noticeable that the training dataset and testing dataset are different. Seventeen attack types in the testing dataset are not in the training dataset, as shown in bold format in Table 3. In addition, the distributions of attack types in the training and testing dataset are different. There are about 20% attacks in both datasets.

## *Data preprocessing*

The proposed STA algorithm requires a labeled and an unlabeled dataset for its input. We name these datasets as train_labeled and train_unlabeled. We use the testing dataset named test_data to test our algorithm. The procedure to obtain these datasets is described as follows.

We remove some samples from the testing dataset or training dataset. The samples' features where their values are unique to one dataset are removed from the testing and training dataset. For example, the value of one sample feature in the testing dataset is "icmp," which is not in the training dataset. Similarly, the value of one sample feature in the training dataset is "red_i" but not in the testing dataset. We removed them in both dataset. Since the number of such samples is small, they do not really affect the overall distribution of the dataset.

Since our focus is a binary classification of the dataset to normal or attack, we continue by labeling all the data which have a type "normal" with 1. All data with abnormal type will be marked as 2 which denotes as attack. This procedure is done only once at the beginning of the simulations both for the training and testing dataset.

## *Feature selection and dataset extraction*

*Feature selection using chi-square method.* In this section, chi-square method is used to improve the detection efficiency by reducing the number of feature dimensions and selecting the minimum useful features to express the whole dataset accurately. Eighteen features are selected as optimal subset based on the ranking search method. The detailed descriptions and feature weights are given in Table 4, where Dhsspr is Dst_host_same_src_port_rate; Dhsdhr is Dst_host_srv_diff_host_rate; Dhsc is Dst_host_srv_count; Dhdsr is Dst_host_diff_srv_rate; Dhssr is Dst_host_same_srv_rate; Sdhr is Srv_diff_host _rate; Dhsser is Dst_host_srv_serror_rate as well as Dhsr is Dst_host_serror_rate.

**Table 2.** Attack types and distribution in training dataset.

| Classification of attacks | Attack name | Proportion (%) |
|---|---|---|
| PROBE | Port-sweep, IP-sweep, Nmap, Satan | 0.83 |
| DoS | Neptune, Smurf, Pod, Teardrop, Land, Back | 79.23 |
| U2R | Buffer-overflow, Load-module, Perl, Rootkit | 0.01 |
| R2L | Guess-password, Ftp-write, Imap, Phf<br>Multihop, spy, warezclient, Warezmaster | 0.022 |

PROBE: probing attack; Dos: denial-of-service; U2R: user to root; R2L: remote to local.

**Table 3.** Attack types and distribution in testing dataset.

| Classification of attacks | Attack name | Proportion (%) |
|---|---|---|
| PROBE | Port-sweep, IP-sweep, Nmap, Satan<br>**Saint**, **Mscan** | 1.34 |
| DoS | Neptune, Smurf, Pod, Teardrop, Land, Back<br>**Apache2**, **Udpstorm**, **Processtable**, **Mail-Bomb** | 73.90 |
| U2R | Buffer-overflow, Load-module, Rootkit, Perl<br>**Xterm**, **Ps**, **Sqlattack** | 0.02 |
| R2L | Guess-password, Ftp-write, Imap, Phf<br>Multihop, spy, warezclient, Warezmaster<br>**Snmpgetattack**, **Named**, **Xlock**, **Xsnoop**<br>**Send-Mail**, **Http-Tunnel**, **Worm**, **Snmp-Guess** | 5.26 |

PROBE: probing attack; DoS: denial-of-service; U2R: user to root; R2L: remote to local.

**Table 4.** Features selected by chi-square method.

| Rank | Feature name | Data type | Description | Weight |
|---|---|---|---|---|
| 1 | Count | Continuous | Sum of connections to specific destination | 0.9698 |
| 2 | Src_bytes | Continuous | Number of data bytes from source to destination | 0.9485 |
| 3 | Service | Discrete | Different network services on the destination, such as http, ftp, and telnet | 0.9179 |
| 4 | Dst_bytes | Continuous | Number of data bytes from source to destination | 0.8615 |
| 5 | Srv_count | Continuous | Sum of connections to the same destination | 0.7763 |
| 6 | Dhsspr | Continuous | Percentage of connections to the same source port | 0.7645 |
| 7 | Protocol_type | Discrete | Type of protocol, such as tcp and udp | 0.7408 |
| 8 | Logged_in | Discrete | 1 if successfully logged in; 0 otherwise | 0.7402 |
| 9 | Dst_host_count | Continuous | Total connections to specific IP | 0.6335 |
| 10 | Dhsdhr | Continuous | Percentage of connections to various destinations | 0.5842 |
| 11 | Dhsc | Continuous | Sum of connections to the same service | 0.4934 |
| 12 | Dhdsr | Continuous | Percentage of connections to different services | 0.4859 |
| 13 | Dhssr | Continuous | Percentage of connections to the same service | 0.4755 |
| 14 | Sdhr | Continuous | Percentage of connections to different hosts | 0.4434 |
| 15 | Same_srv_rate | Continuous | Percentage of connections to the same service | 0.4037 |
| 16 | Diff_srv_rate | Continuous | Percentage of connections to different services | 0.4001 |
| 17 | Dhsser | Continuous | Percentage of connections that activated the flag | 0.3804 |
| 18 | Dhsr | Continuous | Percentage of connections that activated the flag s0, s1, s2, or s3 among connections in dst_host_count | 0.3736 |

*Dataset extraction.* To extract the labeled and unlabeled dataset, first we split the KDDcup99 training dataset into two separate parts. For this split, we use stratified sampling[33] without replacement. This ensures that even though after every run of the algorithm, we have different data at each sample, and the number of normal samples equals to the number of attack samples. From these two parts, one part is used for the labeled dataset and the other one is for the unlabeled ones. Based on the required size for the labeled, or unlabeled dataset, and the ratio of the normal/attacks, we select again random stratified samples from each part. We obtain

**Table 5.** Symbols used throughout equations (6)–(9).

| Symbol | Description |
|---|---|
| *TP* (true positives) | $x_i$ predicted to be in $C_i$ and in fact it is in it |
| *TN* (true negatives) | $x_i$ not predicted to be in $C_i$ and in fact it is not in it |
| *FP* (false positives) | $x_i$ predicted to be in $C_i$; however, in fact it is not in it |
| *FN* (false negatives) | $x_i$ not predicted to be in $C_i$; however, in fact it is in it |

the testing dataset following the same procedure. This procedure of randomly partitioning and sampling the dataset is repeated every time that we execute our algorithm. This ensures that we have different data for every iteration.

### Simulations

We run simulations where we vary the size of the testing dataset, the training dataset including the unlabeled and the labeled dataset. During each simulation, the ratio of normal/attacks is 75%/25% for the labeled dataset and 70%/30% for the unlabeled dataset in the training step. In the testing step, the ratio of normal/attacks is 83%/17%. For each different set of parameters, the results are obtained by averaging 30 times simulations. As already mentioned above, for each iteration, we will have different training and testing samples.

In order to get the average results, the proposed STA method is run 30 times for each simulation. A Lenovo G490 laptop, with an Intel(R) core(TM) i5-3230M CPU at 2.60 GHz and 6 GB (DDR3 1600 MHz) of RAM is used to run all the simulations. The R version used is 3.3.1.

### General results

We use several classical evaluation metrics to measure the performance of the proposed STA method. The definition of them is shown in equations (6)–(9). The symbols used throughout equations (6)–(9) are given in Table 5.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{6}$$

$$Precision = \frac{TP}{TP + FP} \tag{7}$$

$$Sensitivity = \frac{TP}{TP + FN} \tag{8}$$

$$Detection\ rate = \frac{TN}{TN + FN} \tag{9}$$

**Table 6.** Average results over the total number of simulations.

| Name | FFV | SFV |
|---|---|---|
| Testing size | 12,000 | 12,000 |
| Unlabeled size | 32,000 | 32,000 |
| Labeled size | 6000 | 6000 |
| Accuracy (%) | 96.97 | **96.99** |
| Precision (%) | 98.57 | **98.61** |
| Sensitivity (%) | 97.81 | 97.80 |
| Detection rate (%) | 88.96 | 88.91 |
| False positive rate (%) | 7.42 | **7.26** |
| Train-time (s) | 85.13 | **47.63** |
| Predict-time (s) | 4.82 | **2.66** |

Bold faced values for SFV represents that, SFV has a better performance on some metrics, such as accuracy or precision, compared with those of FFV. FFV: full feature value; SFV: subset feature value.

**Table 7.** Results for 10% ratios of labeled and unlabeled datasets with the subset of features.

| Testing size | 5000 | 25,000 | 45,000 |
|---|---|---|---|
| Unlabeled size | 500 | 2500 | 4500 |
| Labeled size | 50 | 250 | 450 |
| Accuracy (%) | 93.82 | 96.39 | 96.59 |
| Precision (%) | 97.35 | 97.94 | 98.08 |
| Sensitivity (%) | 95.43 | 97.73 | 97.83 |
| Detection rate (%) | 76.20 | 88.63 | 89.13 |
| False positive rate (%) | 12.07 | 10.41 | 9.68 |

$$False\ positive\ rate = \frac{FP}{FP + TN} \tag{10}$$

The average values, which are based on the dataset with full feature set as well as with subset of features selected by chi-square method, are shown in Table 6, where FFV is the full feature value and SFV is the subset feature value. All values are obtained from 30 runs. It is worth to note that in comparison with the method using the full feature set, the time cost is reduced by about 45% with the subset of features. The overall performance also increases with the subset of features.

Table 7 shows that just by using 450 labeled samples and 4500 unlabeled ones, we can classify a testing set of size 45,000, with a precision of 98.08%.

### Varying the size of datasets

In order to better explain our proposed STA algorithm, we vary the size of datasets. Specifically, we utilize the sizes of testing dataset from 5000 to 65,000 samples, with a step of 10,000. The size of the unlabeled dataset is 10% of the size of current testing dataset. Similarly, the size of the labeled dataset is 10% of the current unlabeled dataset size. Both labeled and unlabeled samples are randomly selected from the KDDcup99 dataset.[13] Figures 2–4 report that in comparison with the

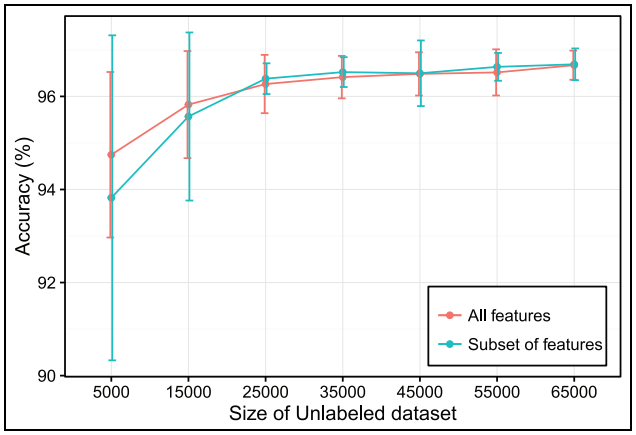


**Figure 2.** Accuracy over unlabeled dataset size increases.

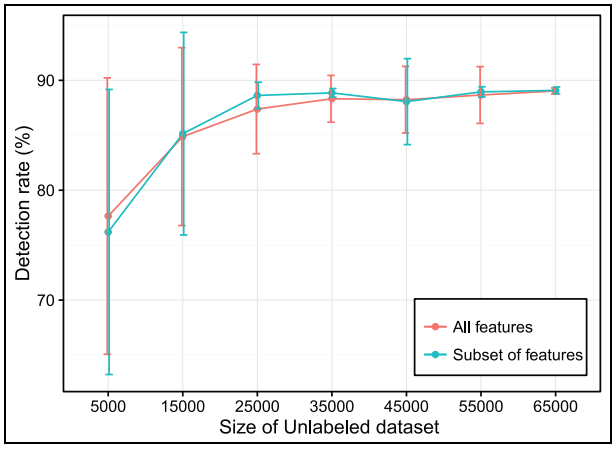


**Figure 3.** Detection rate over unlabeled dataset size increases.

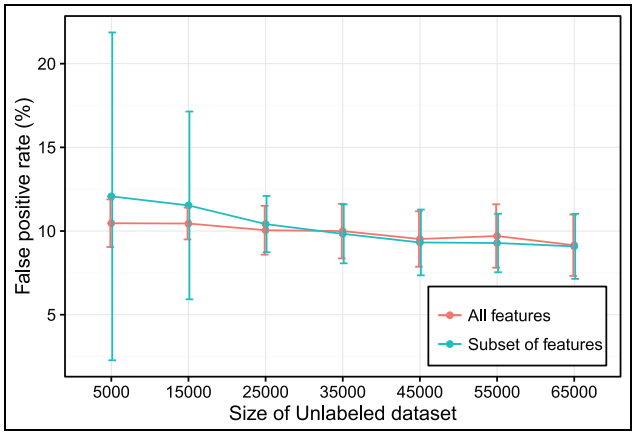


**Figure 4.** False positive rate over unlabeled dataset size increases.

proposed STA method using the full feature set, the accuracy, detection rate and false positive rate are improved using the subset of features.

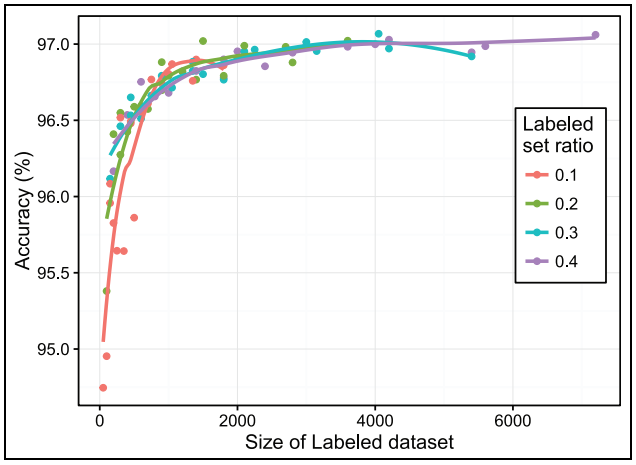


**Figure 5.** Accuracy over a different set of labeled/unlabeled data ratios.

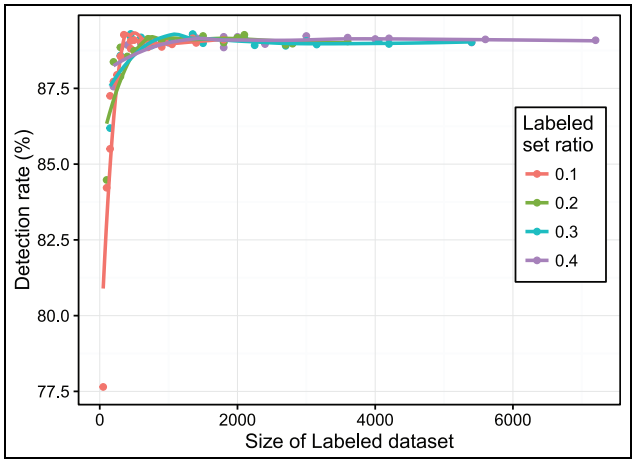


**Figure 6.** Detection rate over a different set of labeled/unlabeled data ratios.

We observe that with the number of unlabeled training data increasing, the performance of our experiment increases. This empirical result is very useful in real applications since the unlabel data are cheap to obtain and can help to increase the penalty for the misclassified samples in the proposed algorithm. Therefore, the experiment performance can be improved by simply using a large number of unlabeled samples.

The accuracy, detection rate as well false positive rate with the different ratios of labeled dataset can be seen in Figures 5–7. Similarly, we extract the different labeled dataset with the ratios of 10%, 20%, 30%, and 40% from the current unlabeled dataset. In these figures, the vertical bar of each point presents the value of standard deviation, with lower values being better. Figures 5–7 illustrate that the proposed STA algorithm can achieve a very good result even with only a very small fraction of labeled dataset. The proposed STA

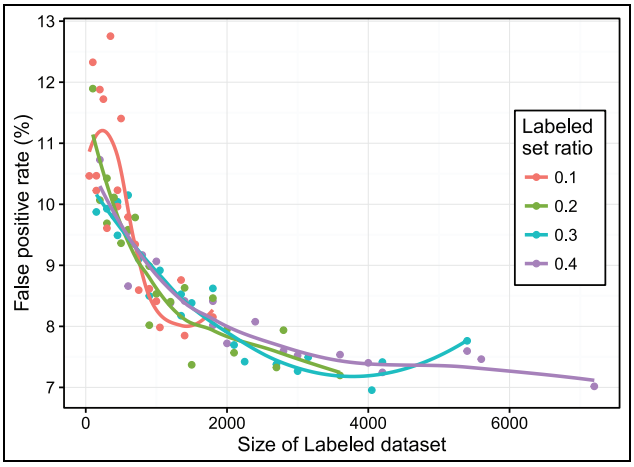


**Figure 7.** False positive rate over a different set of labeled/unlabeled data ratios.

**Table 8.** Semi-supervised tri-Adaoobst (STA) approach and semi-supervised SVM (SSM) method.

| | STA | SSM |
|---|---|---|
| Detection rate (%) | 89.05 | 97.34 |
| False positive rate (%) | **7.54** | 8.65 |
| Precision rate (%) | **98.54** | 92.35 |
| Detection time (s) | **7.34** | 98.14 |

Bold faced values for STA represents that, STA has a better performance on some metrics, such as false positive rate, compared with those of SSM. SVM: support vector machine.

algorithm starts with a small number of labeled samples in the training dataset and then increases the number of labeled samples incrementally using a large amount of unlabeled samples in order to improve the detection performance.

### *Comparison with existing methods*

Finally, to have a comparison of our model, we compare some of our results with those obtained by Jimin Li et al.[19] This article used tri-training approach with three different SVM algorithms and employed the KDDcup99 dataset[13] to prove the performance of their algorithm.

Both the training dataset and testing dataset in our experiment have the same sizes or ratios with those in the paper by Jimin Li et al.[19] to evaluate the performance of STA algorithm. Specifically, the labeled dataset consists of 4000 samples, with 3000 normal and 1000 attacks. The unlabeled training dataset has 10,000 samples, with 7000 normal samples and 3000 attacks. Finally, the testing set consists of 30,000 samples, with 25,000 normal samples and 5000 attacks. Table 8 shows a basic comparison of our results.

Thirty rounds are performed in order to get the average results and in each round, different samples are randomly selected. From the simulation results, it is obvious that the proposed method has a good detection rate and a low false positive rate. In addition, the detection time of the proposed method is low, which proves that STA is efficient.

## Conclusion

In many real-world deployments, particularly in systems where the data volume, variety, velocity, and veracity are significant, labeled network samples are difficult, expensive, or time-consuming to obtain. Compounding the challenge, experienced human annotators are generally required to classify the data. However, unlabeled samples are easier to be collected in the real world. For instance, by capturing the network traffic, say in a hospital network or a government agency network, using existing tools.

In our paper, we proposed the STA algorithm based on the training dataset which consists of only a few labeled samples and a massive unlabeled samples. Three different Adaboost methods were combined using the tri-training approach. In the evaluations, we utilized different sizes of labeled dataset and mixed these labeled datasets with unlabeled datasets of varying sizes to generate the training dataset. We demonstrated that our method achieves a low false positive rate, a good detection rate, and a high precision, even when the size of the labeled dataset is 1% of the testing dataset. Just by using 450 labeled samples and 4500 unlabeled ones, we were able to classify a testing set of size 45,000, with a precision of 98.08%. This showed that even using only a very small labeled dataset in the training step, the proposed STA method can achieve a very good performance. It is noteworthy that the detection time of our algorithm is very low, which is a crucial factor in real-world deployment.

As for the future work, we plan to implement a prototype of the proposed algorithm in a real-world environment, perhaps within a close network. This will allow us to more effectively evaluate its real-world utility.

### Acknowledgements

The authors would like to thank our colleague Arne Bochem for his suggestions. Y.Y. and Z.W. would like to thank the scholarship support from the China Scholarship Council (CSC).

### Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

## Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: Z.W. was also supported by the grants from the NSFC (nos 61772407, 61771387), Youth Fund of the Ministry of Education of China (no. 16YJCZH109), Shaanxi Science (no. 2014M18), Shaanxi Department of Education (nos 16JS080, 18JK1216, SGH18H466), CERNET (NGII20171202, NGII20170303), YANAN Science and Technology Project (no. 2018KG-02), Ph.D Innovation (no. 112-451115006), and Innovation research and development (nos 2016JXKY-20, 310/252051835).

## ORCID iDs

Yali Yuan https://orcid.org/0000-0002-9258-9929
Zhixiao Wang https://orcid.org/0000-0002-8143-1579

## References

1. Aburomman AA and Reaz MI. A survey of intrusion detection systems based on ensemble and hybrid classifiers. *Comput Security* 2017; 65: 135–152.
2. Buczak AL and Guven E. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Commun Surv Tut* 2016; 18(2): 1153–1176.
3. Iqbal S, Kiah MM, Dhaghighi B, et al. On cloud security attacks: a taxonomy and intrusion detection and prevention as a service. *J Netw Comput Appl* 2016; 74: 98–120.
4. Peng J, Choo KR and Ashman H. User profiling in intrusion detection: a review. *J Netw Comput Appl* 2016; 72: 14–27.
5. Roesch M. Snort: lightweight intrusion detection for networks. *Lisa* 1999; 99(1): 229–238.
6. Paxson V. Bro: a system for detecting network intruders in real-time. *Comput Netw* 1999; 31(23–24): 2435–2463.
7. Kruegel C, Valeur F and Vigna G. Intrusion detection and correlation: challenges and solutions. *Springer Science & Business Media, Vol. 14* 2004.
8. Chitrakar R and Huang C. Selection of candidate support vectors in incremental SVM for network intrusion detection. *Comput Security* 2014; 45: 231–241.
9. Enache AC and Patriciu VV. Intrusions detection based on support vector machine optimized with swarm intelligence. In: *Proceedings of the 2014 IEEE 9th IEEE international symposium on applied computational intelligence and informatics (SACI)*, Timisoara, 15–17 May 2014, pp.153–158. New York: IEEE.
10. Bukhtoyarov V and Zhukov V. Ensemble-distributed approach in classification problem solution for intrusion detection systems. In: *Proceedings of the international conference on intelligent data engineering and automated learning*, Salamanca, 10–12 September, pp.255–265. New York: Springer.
11. Luo M, Wang L, Zhang H, et al. A research on intrusion detection based on unsupervised clustering and support vector machine. In: *Proceedings of the international conference on information and communications security*, Huhehaote, 10–13 October 2003, pp.325–336. New York: Springer.
12. Zhou Z and Li M. Tri-training: exploiting unlabeled data using three classifiers: knowledge and Data Engineering. *IEEE Trans* 2005; 17(11): 1529-1541.
13. *The third international knowledge discovery and data mining tools competition*, http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html (2002, accessed 20 February 2016).
14. Zhu X and Goldberg AB. Introduction to semi-supervised learning. *Syn Lect Artifi Intell Mach Learn* 2009; 3(1): 1–130.
15. Mao C, Lee H, Parikh D, et al. Semi-supervised co-training and active learning based approach for multi-view intrusion detection. In: *Proceedings of the 2009 ACM symposium on applied computing*, Honolulu, HI, 2009, pp.2042–2048, https://dl.acm.org/citation.cfm?id=1529735
16. Chen C, Gong Y and Tian Y. Semi-supervised learning methods for network intrusion detection. In: *Proceedings of the 2008 IEEE international conference on systems, man and cybernetics*, Singapore, 12–15 October 2008, pp.2603–2608. New York: IEEE.
17. Pachghare VK, Khatavkar V and Kulkarni P. Performance analysis of semi-supervised intrusion detection system. *Int J Comput Appl* 2011; 15–19, https://pdfs.semanticscholar.org/ccbc/39054f52bb33047ece2196f8a6be07063659.pdf
18. Chiu Y, Lee J, Chang C, et al. *Applications and theoretical aspects*. New York: Springer, 2010, pp.595–605.
19. Li J, Zhang W and Li K. A novel semi-supervised SVM based on tri-training for intrusition detection. *J Comput* 2010; 5(4): 638–645.
20. Ashfaq RR, Wang X, Huang J, et al. Fuzziness based semi-supervised learning approach for intrusion detection system. *Inform Sci* 2017; 378: 484–497.
21. Yuan Y, Huo L and Hogrefe D. Two layers multi-class detection method for network intrusion detection system. In: *Proceedings of the 2017 IEEE symposium on computers and communications (ISCC)*, Heraklion, 3–6 July 2017, pp.767–772. New York: IEEE.
22. Yao H, Fu D, Zhang P, et al. MSML: a novel multi-level semi-supervised machine learning framework for intrusion detection system. *IEEE Internet Thing J* 2018; 6: 1949–1959.
23. Camacho J, Maciá-Fernández G, Fuentes Garc í a NM, et al. Semi-supervised multivariate statistical network monitoring for learning security threats. *IEEE Trans Inform Forens Security* 2019; 14: 2179–2189.
24. Freund Y and Schapire RE. Experiments with a new boosting algorithm. In: *ICML'96 proceedings of the thirteenth international conference on international conference on machine learning*, vol. 96, Bari, 3–6 July 1996, pp.148–156. New York: ACM.
25. Schapire RE and Singer Y. Improved boosting algorithms using confidence-rated predictions. *Mach Learn* 1999; 37(3): 297–336.
26. Viola P and Jones M. Rapid object detection using a boosted cascade of simple features. In: *Proceedings of the 2001 IEEE computer society conference on computer vision*

*and pattern recognition (CVPR 2001)*, Kauai, HI, 8–14 December, https://ieeexplore.ieee.org/document/990517
27. Liu H and Setiono R. Chi2: feature selection and discretization of numeric attributes. In: *Proceedings of 7th IEEE international conference on tools with artificial intelligence*, Herndon, VA, 5–8 November, pp.388–391. New York: IEEE.
28. Yang Y and Pedersen JO. A comparative study on feature selection in text categorization. In: *ICML'97 proceedings of the fourteenth international conference on machine learning*, vol. 97, San Francisco, CA, 8–12 July 1997, pp.412–420. New York: ACM.
29. Abdelwadood M. Chi square feature extraction based SVMS Arabic language text categorization system. *J Comput Sci* 2007; 3(6): 430–435.
30. Yuan Y, Kaklamanos G and Hogrefe D. A novel semi-supervised adaboost technique for network anomaly detection. In: *MSWiM'16 Proceedings of the 19th ACM international conference on modeling, analysis and simulation of wireless and mobile systems*, 2016, pp.111–114, https://dl.acm.org/citation.cfm?id=2989177
31. Holte R. Very simple classification rules perform well on most commonly used datasets. *Mach Learn* 1993; 11(1): 63–90.
32. Mukkamala S, Sung AH and Abraham A. Intrusion detection using an ensemble of intelligent paradigms. *J Netw Comput Appl* 2005; 28(2): 167–182.
33. Neyman J. On the two different aspects of the representative method: the method of stratified sampling and the method of purposive selection. *J Roy Stat Soc* 1934; 97(4): 558–625.