AdaBoost-Based Algorithm for Network Intrusion Detection

Weiming Hu, *Senior Member, IEEE*, Wei Hu, and Steve Maybank, *Senior Member, IEEE*

***Abstract*—Network intrusion detection aims at distinguishing the attacks on the Internet from normal use of the Internet. It is an indispensable part of the information security system. Due to the variety of network behaviors and the rapid development of attack fashions, it is necessary to develop fast machine-learning-based intrusion detection algorithms with high detection rates and low false-alarm rates. In this correspondence, we propose an intrusion detection algorithm based on the AdaBoost algorithm. In the algorithm, decision stumps are used as weak classifiers. The decision rules are provided for both categorical and continuous features. By combining the weak classifiers for continuous features and the weak classifiers for categorical features into a strong classifier, the relations between these two different types of features are handled naturally, without any forced conversions between continuous and categorical features. Adaptable initial weights and a simple strategy for avoiding overfitting are adopted to improve the performance of the algorithm. Experimental results show that our algorithm has low computational complexity and error rates, as compared with algorithms of higher computational complexity, as tested on the benchmark sample data.**

***Index Terms*—AdaBoost, computational complexity, detection rate, false-alarm rate, intrusion detection.**

I. Introduction

With the development of the Internet, information security is of increasing importance. Information security on the Internet includes the following.

1) Protection: The information system is protected automatically to avoid security violations that are called intrusions.
2) Detection: Security violations are detected when they occur.
3) Reaction: Reactions, such as automatic alarm or pursuit of hackers, are performed when the system is intruded.
4) Recovery: The information system automatically repairs the damage caused by an intrusion.

Intrusion detection is a crucial part of information security. Only if intrusions are correctly detected can the subsequent reaction and recovery be implemented.

A. Definition and Categories

There is no standard definition of intrusion detection. Generally, intrusion detection is the detection of network behaviors that violate or endanger network security. Intrusion detection can be treated as a pattern recognition problem—distinguishing between network attacks and normal network behaviors or further distinguishing between different categories of attacks.

Manuscript received November 10, 2006; revised April 4, 2007. This work was supported in part by the National Natural Science Foundation under Grants 60520120099 and 60672040 and in part by the National 863 High-Tech R&D Program of China under Grant 2006AA01Z453. This paper was recommended by Associate Editor J. Basak.

W. Hu and W. Hu are with the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing 100080, China (e-mail: wmhu@nlpr.ia.ac.cn; whu@nlpr.ia.ac.cn).

S. Maybank is with the School of Computer Science and Information Systems, Birkbeck College, University of London, WC1E 7HX London, U.K. (e-mail: sjmaybank@dcs.bbk.ac.uk).

Digital Object Identifier 10.1109/TSMCB.2007.914695

a) Host- and network-based detections: As far as the data source is concerned, intrusion detection can be classified into host- and network-based detections [1]. Host-based approaches detect intrusions utilizing audit data that are collected from the target host machine. As the information provided by the audit data can be extremely comprehensive and elaborate, host-based approaches can obtain high detection rates and low false-alarm rates. However, there are disadvantages for host-based approaches, which include the following.

1) Host-based approaches cannot easily prevent attacks: when an intrusion is detected, the attack has partially occurred.
2) Audit data may be altered by attackers, influencing the reliability of audit data.

Network-based approaches detect intrusions using the IP package information collected by the network hardware such as switches and routers. Such information is not so abundant as the audit data of the target host machine. Nevertheless, there are advantages for network-based approaches, which include the following.

1) Network-based approaches can detect the so-called "distributed" intrusions over the whole network and thus lighten the burden on each individual host machine for detecting intrusions.
2) Network-based approaches can defend the machine against attack, as detection occurs before the data arrive at the machine.

b) Misuse and anomaly detections: Intrusion detection is divided into misuse and anomaly detections [1], [10]. Misuse detection utilizes signatures of attacks to detect intrusions by modeling attacks. Misuse detection has high detection rates for the well-known intrusions but fails to detect novel intrusions. Anomaly detection is based on the models for normal behaviors. Any deviation from the constructed models of normal behaviors is considered an anomaly [21]. As it is difficult to precisely model all normal behaviors, it is easy for anomaly detection to mistakenly classify normal behaviors as attacks.

B. Related Work

In the following, we review the related work, focusing on the types of algorithms that are described in the literature.

1) Statistics-Based Approaches: Denning [2] proposes a statistical method for intrusion detection. According to audit data, a profile is constructed to describe a given subject (network user) or a given object (task). Several metrics are defined for the profiles. The Gaussian models of the metrics are constructed to detect intrusions. Li *et al.* [14] utilize statistical characteristics of n-grams to detect intrusions in the host system. Vigna and Kemmerer [26] use data that are sourced from network nodes, rather than the audit data, to construct profiles, enlightening the research on network-based intrusion detection.

Some researchers propose more complex metrics and statistical models. Qu *et al.* [27] analyze the attacks to routing protocols by estimating the frequency of each event related to a protocol and propose a metric to describe similarity between the observed event distribution and the expected distribution. It is assumed that the metric obeys the chi-square distribution. Ye *et al.* [28] extract an event frequency vector and then measure the chi-square distance between this vector and the expected frequency vector. It is assumed that the distance obeys the standard Gaussian distribution. Li and Manikopoulos [13] propose some representative parameters of IP data flow, and they model the parameters using a hyperbolic distribution.

So far, we have discussed static data models. Dynamic data models are also used. Caberera *et al.* [29] assume that the first derivative of the number of observed events in a time segment obeys the Poisson distribution, from which the Kolmogorov statistical values are extracted to measure the dissimilarity between observation network and

normal behavior signals. Ye *et al.* [30] represent a sequence of events in time order as a Markov stochastic process. The joint probability for a particular sequence of events is used to distinguish between normal network behaviors and intrusions. In recent years, the hidden Markov model has been used in intrusion detection based on host audit data [31], [32].

2) Data-Mining-Based Approaches: Data mining is used in intrusion detection [8], [11]. Lee *et al.* [11] use data-mining techniques to construct rules describing normal network behaviors. The rules include association rules that describe frequency associations between any two fields of the network record database and also frequent episodes that describe the frequency with which a field takes a certain value after two other fields have particular values in a definite time interval. Deviations from these rules indicate an attack on the network. Han *et al.* [34] analyze the content for network data packages and use the data-mining techniques to acquire attack signatures. Qin and Hwang [35] propose an approach, which dynamically omits some nonfunctionary frequent-episode rules, as a supplement to the data-mining-based approaches. Otey *et al.* [52] propose a general-purpose outlier detection algorithm that works on mixed attribute data in distributed settings. Furthermore, they extend their algorithm to handle dynamic and streaming data sets.

3) Supervised Learning-Based Approaches: Recently, methods from machine learning and pattern recognition have been utilized to detect intrusions. Supervised learning and unsupervised learning are both used. In this section, we review the supervised-learning-based methods, and the unsupervised learning-based methods are reviewed in Section I-B4.

For supervised learning for intrusion detection, there are mainly supervised neural network (NN)-based approaches [17], [24], and support vector machine (SVM)-based approaches [7], [25].

a) NN-based approaches: Bonifacio *et al.* [36] propose an NN for distinguishing between intrusions and normal behaviors. They unify the coding of categorical fields and the coding of character string fields in order to map the network data to an NN. Rapaka *et al.* [37] use execution numbers of system calls in a host machine as the features of network behaviors to train the NN. Zhang *et al.* [24] propose an approach for intrusion detection using hierarchical NNs. Han and Cho [38] use evolutionary NNs to detect intrusions.

b) SVM-based approaches: Mukkamala *et al.* [39], [40] use SVMs to distinguish between normal network behaviors and intrusions and further identify important features for intrusion detection. Mill and Inoue [41] propose the TreeSVM and ArraySVM for solving the problem of inefficiency of the sequential minimal optimization algorithm for the large set of training data in intrusion detection. Zhang and Shen [25] propose an approach for online training of SVMs for real-time intrusion detection based on an improved text categorization model.

Aside from the aforementioned supervised-learning-based approaches for intrusion detection, decision tree [42] and discriminant analysis [43] are applied to detect intrusions. Comparisons between different classifiers and fusion of multiple classifiers for intrusion detection are studied in [1], [19], and [44].

4) Unsupervised Learning-Based Approaches: Supervised learning methods for intrusion detection can only detect known intrusions. Unsupervised learning methods can detect the intrusions that have not been previously learned. Examples of unsupervised learning for intrusion detection include K-means-based approaches and self-organizing feature map (SOM)-based approaches [3], [9].

a) K-means-based approaches: Guan *et al.* [45] propose a K-means-based clustering algorithm, which is named Y-means, for intrusion detection. Xian *et al.* [46] combine the fuzzy K-means method and a clonal selection algorithm to detect intrusions. Jiang *et al.* [47] use the incremental clustering algorithm that is an extension of the K-means algorithm to detect intrusions.

b) SOM-based approaches: Hoglund *et al.* [48] extract features that describe network behaviors from audit data, and they use the SOM to detect intrusions. Kayacik *et al.* [9] propose a hierarchical SOM approach for intrusion detection. Specific attention is given to the hierarchical development of abstractions, which is sufficient to permit direct labeling of SOM nodes with connection type. Sarasamma *et al.* [21] propose a hierarchical SOM for intrusion detection. They use the classification capability of the SOM on selected dimensions of the data set to detect anomalies. Their results are among the best known for intrusion detection.

Previous discussions review the related work. Current approaches for intrusion detection have the following two problems.

1) Current approaches often suffer from relatively high false-alarm rates, whereas they have high detection rates. As most network behaviors are normal, resources are wasted on checking a large number of alarms that turn out to be false.
2) Their computational complexities are oppressively high. This limits the practical applications of these approaches.

C. Our Work

Aiming at constructing an intrusion detection approach with a low computational complexity, a high detection rate, and a low false-alarm rate, in this correspondence, we apply the AdaBoost algorithm to intrusion detection [57]. The motivation for applying the AdaBoost algorithm includes the following points.

1) The AdaBoost algorithm is one of the most popular machine-learning algorithms. Its theoretical basis is sound, and its implementation is simple. It has been applied to many pattern recognition problems, such as face recognition. However, the application of the AdaBoost algorithm to intrusion detection has not been explored so far.
2) The AdaBoost algorithm corrects the misclassifications made by weak classifiers, and it is less susceptible to overfitting than most learning algorithms. Recognition performances of the AdaBoost-based classifiers are generally encouraging.
3) Data sets for intrusion detection are a heterogeneous mixture of categorical and continuous types. The different feature types in such data sets make it difficult to find relations between these features. By combining the weak classifiers for continuous features and the weak classifiers for categorical features into a strong classifier, the relations between these two different types of features are handled naturally, without any forced conversions between continuous and categorical features.
4) If simple weak classifiers are used, the AdaBoost algorithm is very fast.

In our AdaBoost-based algorithm for intrusion detection, decision stumps are used as weak classifiers. The decision rules are provided for both categorical and continuous features. Special provision is made for overfitting. Experimental results show that our algorithm has low computational complexity and error rates, as compared with algorithms of higher computational complexity, as tested on the benchmark sample data.

The remainder of this correspondence is organized as follows. Section II gives an overview of our algorithm. Section III describes in detail our AdaBoost-based intrusion detection algorithm. Section IV analyzes the computational complexity of our algorithm. Section V illustrates the experimental results. The last section summarizes the correspondence.

II. Overview of our Algorithm

According to the characteristics of the AdaBoost algorithm and the characteristics of the network intrusion detection problem, the

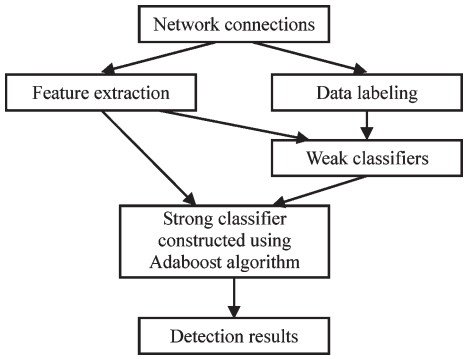

Fig. 1. Framework of our algorithm.

framework of our approach consists of the following four modules: feature extraction, data labeling, design of the weak classifiers, and construction of the strong classifier, as shown in Fig. 1.

a) Feature extraction: For each network connection, the following three major groups of features for detecting intrusions are extracted: basic features of individual Transmission Control Protocol (TCP) connections, content features within a connection suggested by domain knowledge, and traffic features computed using a 2-s time window [23]. The framework for constructing these features can be found in [10].

b) Data labeling: Because the AdaBoost algorithm uses supervised learning, a set of data has to be labeled for training. This labeled data set should contain both normal samples labeled as "+1" and attack samples labeled as "−1." We explain two points: 1) In contrast to misuse detection, which utilizes signatures of attacks to detect intrusions, and anomaly detection, which detects intrusions by modeling normal behaviors, as mentioned in Section I-Ab, our algorithm models both attacks and normal behaviors to detect intrusions. 2) It is hard to obtain a large amount of labeled data in realistic settings. In our previous paper [54], we propose a hierarchical graph-theoretic clustering active learning algorithm for automatically selecting highly informative data for people to label under the condition that a small number of labeled samples are ready for use.

c) Design of weak classifiers: The AdaBoost algorithm requires a group of weak classifiers designed beforehand. An individual weak classifier is simple and easy to implement. Its classification accuracy is relatively low.

d) Construction of the strong classifier: A strong classifier is obtained by combining the weak classifiers. The strong classifier has higher classification accuracy than each weak classifier.

A strong classifier is trained using the sample data. Then, a new network connection, which is represented by the three groups of features introduced in a) earlier, is input to the strong classifier and is classified as either "normal" or "attack" as the detection result.

III. Methodology

It is the construction of weak classifiers that mainly influences the computational complexity of AdaBoost. LogitBoost and Gentle AdaBoost algorithms employ recursion to construct weak classifiers; thus, their computational complexities are comparatively high. The continuous AdaBoost algorithm estimates weighted conditional probabilities, and all samples are checked in each iteration; therefore, the computational complexity of the algorithm is also comparatively high. In a discrete AdaBoost algorithm, the optimal weak classifier in each iteration is chosen from a group of weak classifiers; for that reason, its computational complexity is comparatively low. Therefore, we select the discrete AdaBoost algorithm to learn the classifier. In this section, we describe the technical details of our algorithm, including the construction of the weak classifiers, the choice of initial weights, and the handling of overfitting.

A. Weak Classifiers

Classification algorithms in common use, such as K-neighbors, supervised neural networks, and SVMs, can be used as weak classifiers. In our algorithm, we select decision stumps [49] as weak classifiers. A decision stump is a decision tree with a root node and two leaf nodes. For each feature in the input data, a decision stump is constructed. The following points support our selection of decision stumps as the weak classifiers: 1) the model that decision stumps use is very simple; 2) there is only one comparison operation in each decision stump for testing a sample; thus, the test time for each decision stump is very low; and 3) the decision stumps for continuous features and the decision stumps for categorical features can be constructed in a similar way, such that the strong classifier can easily combine the information from continuous and categorical features in the next step.

a) Decision stumps for categorical features: A categorical feature f can only take finite discrete values. A decision stump corresponds to a partition of the range of f into two nonoverlapping subsets C_p^f and C_n^f. Let X be the feature vector, and X_f be the component of X, which corresponds to feature f. Then, the decision stump corresponding to C_p^f and C_n^f is described as

$$h_f(X) = \begin{cases} +1, & X_f \in C_p^f \\ -1, & X_f \in C_n^f. \end{cases} \tag{1}$$

Let $\varepsilon_{h_f}^+$ and $\varepsilon_{h_f}^-$ denote the false-classification rates of the decision stump h_f for normal and attack samples, respectively. The optimal subsets $\widetilde{C}_p^f$ and $\widetilde{C}_n^f$ that correspond to the optimal decision stump $\widetilde{h}_f$ are determined by minimizing the sum of the false-classification rates for the normal and attack samples

$$\left(\widetilde{C}_p^f, \widetilde{C}_n^f\right) = \arg \min_{\left(C_p^f, C_n^f\right)} \left[\varepsilon_{h_f}^+ \left(C_p^f, C_n^f\right) + \varepsilon_{h_f}^- \left(C_p^f, C_n^f\right)\right]. \tag{2}$$

b) Decision stumps for continuous features: For a continuous feature f, given a segmentation value θ, a decision stump h_f can be constructed as

$$h_f(X) = \begin{cases} +1, & X_f \le \theta \\ -1, & X_f > \theta. \end{cases} \tag{3}$$

where X_f denotes the component of feature vector X, which corresponds to feature f.

The optimal segmentation value $\widetilde{\theta}$ corresponding to the optimal decision stump $\widetilde{h}_f$ is determined by minimizing the sum of the false-classification rates for normal and attack samples.

B. Algorithm

In the AdaBoost algorithm, weak classifiers are selected iteratively from a number of candidate weak classifiers and are combined linearly to form a strong classifier for classifying the network data. Let $H = \{\widetilde{h}_f\}$ be the set of constructed weak classifiers. Let the set of training sample data be $\{(x_1, y_1), \ldots, (x_i, y_i), \ldots, (x_n, y_n)\}$, where x_i denotes the ith feature vector; $y_i \in \{+1, -1\}$ is the label of the ith feature vector, denoting whether the feature vector represents a normal behavior or not; and n is the size of the data set. Let $\{w_1, \ldots, w_i, \ldots, w_n\}$ be the sample weights that reflect the importance degrees of the samples and, in statistical terms, represent an estimation of the sample distribution. The AdaBoost-based algorithm for intrusion detection is described as follows.

Step 1) Initialize weights $w_i(1)$ $(i = 1, \ldots, n)$ satisfying $\sum_{i=1}^{n} w_i(1) = 1$. The initialization is further described in Section III-C.

Step 2) Observe the following for $(t = 1, \ldots, T)$.

a) Let ε_j be the sum of the weighted classification errors for the weak classifier h_j

$$\varepsilon_j = \sum_{i=1}^{n} w_i(t) I\left[y_i \neq h_j(x_i)\right] \tag{4}$$

where

$$I[\gamma] = \begin{cases} 1, & \gamma = \text{True} \\ 0, & \gamma = \text{False}. \end{cases} \tag{5}$$

Choose, from constructed weak classifiers, the weak classifier $h(t)$ that minimizes the sum of the weighted classification errors

$$h(t) = \arg \min_{h_j \in H} \varepsilon_j. \tag{6}$$

b) Calculate the sum of the weighted classification errors $\varepsilon(t)$ for the chosen weak classifier $h(t)$.

c) Let

$$\alpha(t) = \frac{1}{2} \log \left(\frac{1 - \varepsilon(t)}{\varepsilon(t)} \right). \tag{7}$$

d) Update the weights by

$$w_i(t+1) = \frac{w_i(t) \exp\left(-\alpha(t) y_i h(t)(x_i)\right)}{Z(t)} \tag{8}$$

where $Z(t)$ is a normalization factor

$$Z(t) = \sum_{k=1}^{n} w_k(t) \exp\left(-\alpha(t) y_i h(t)(x_k)\right). \tag{9}$$

Step 3) The strong classifier is defined by

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha(t) h(t)(x) \right). \tag{10}$$

We explain two points: 1) By combining the decision stumps for both categorical and continuous features into a strong classifier, the relations between categorical and continuous features are handled naturally, without any forced conversions between continuous and categorical features. This is the main reason why our AdaBoost-based algorithm obtains good results for intrusion detection. 2) For the AdaBoost algorithm, it has been proved that the weighted classification error rate for the strong classifier converges to zero as the number of iterations increases [6], [51], i.e., when $T \to \infty$, $\sum_{i=1}^{n} w_i^{(1)} I[H(x_i) \neq y_i] \to 0$, provided that the misclassification rates for the weak classifiers are less than 50%. The decision stumps minimize the sum of the false-classification rates for normal and attack samples. It is guaranteed that the misclassification rates for the selected weak classifiers are lower than 50%; this ensures the convergence of the algorithm. As we focus on the application of the AdaBoost algorithm to intrusion detection in this correspondence, we only briefly discuss the convergence of our algorithm. Rudin *et al.* [51] make a thorough study of the convergence properties of the AdaBoost algorithm. Readers may refer to [51] for details.

C. Adjustable Initial Weights

The initial weights $(w_i(1)\ (i = 1, \ldots, n))$ reflect the importance degrees of the samples and influence the sum of the weighted errors for the strong classifier. Usually, the initial weights are chosen to be equal: $w_i(1) = 1/n\ (i = 1, \ldots, n)$. In the AdaBoost theory, the uniform initial weights have a strong influence on the mean of the classification errors. This is not very suitable for intrusion detection because it is necessary to reduce the false-alarm rate rather than the mean error: In real applications, almost all behaviors are normal. A high false-alarm rate wastes resources, as each alarm must be checked. In the following, we propose adjustable initial weights to make a tradeoff between the false-alarm and detection rates.

Let n_+ and n_- be the numbers of normal and attack samples in the training set, and let ε_+ and ε_- be the false-alarm and false-classification rates (the sum of the false-classification and detection rates equals "1") for the normal and attack samples, respectively. Then, a weighted mean ε of the classification errors is defined as follows:

$$\begin{aligned} \varepsilon &= \frac{1}{n} \sum_{i=1}^{n} I\left[H(x_i) \neq y_i\right] \\ &= \frac{1}{n} \left|\{i : y_i = +1, H(x_i) = -1\}\right| \\ &\quad + \frac{1}{n} \left|\{i : y_i = -1, H(x_i) = +1\}\right| \\ &= \frac{n_+}{n} \varepsilon_+ + \frac{n_-}{n} \varepsilon_-. \end{aligned} \tag{11}$$

From (11), it is shown that the degrees of punishment to the false-alarm and detection rates are different in the case of the uniform weights. In order to make a tradeoff between the false-alarm and detection rates, the weighted classification rate is written in the following form:

$$\varepsilon_w = r\varepsilon_+ + (1 - r)\varepsilon_- \tag{12}$$

where r is the scale factor that penalizes the false-alarm rate. Correspondingly, the initial weights are defined as follows:

$$w_i(1) = \begin{cases} \frac{r}{n_+}, & y_i = +1 \\ \frac{1-r}{n_-}, & y_i = -1 \end{cases}, \qquad (i = 1, \ldots, n). \tag{13}$$

The larger r is, the greater the tendency to reduce the false-alarm rate during learning. By choosing a suitable value for r, the detection and false-alarm rates can be well balanced.

D. Over fitting Handling

In the AdaBoost algorithm, overfitting to some weak classifiers can easily occur [33], [50]. In our algorithm, we adopt the following simple method to avoid the overfitting: In the first T_q iterations, if the sum of the weighted errors for a weak classifier is less than a threshold θ_q, it is considered that this weak classifier fits the samples over-well, i.e., overfits the samples. Then, we select, from the weak classifiers for each of which the sum of the weighted errors is not less than θ_q, the optimal weak classifier that produces the minimum of the weighted errors compared with other weak classifiers for each of which the sum of the weighted errors is not less than θ_q. After the first T_q iterations, this correction for overfitting is discarded. The experimental results show that this simple method avoids the overfitting very well.

E. Handling of Noise

There are noise and outliers in the training data set for intrusion detection. We handle them as follows. First, the classifier is trained using the full training data that include noise and outliers. Then, the sample data whose weights are exceptionally high are selected and treated as noise or outliers (as shown in Section III-B, each sample is associated with a weight in the AdaBoost algorithm). Finally, the noise or outliers in the training data are removed, and the classifier is retrained using the new training data. This way, we extend the applicability of the AdaBoost algorithm to intrusion detection.

TABLE I
NUMBER OF SAMPLES OF VARIOUS TYPES IN THE TRAINING SET

Normal	Attack				Total
	DOS	U2R	R2L	PROBE	
	391458	52	1126	4107	
97278	396743				494021

TABLE II
NUMBER OF SAMPLES OF VARIOUS TYPES IN THE TEST SET

Normal	Attack					Total
	DOS	U2R	R2L	PROBE	Others	
	223298	39	5993	2377	18729	
60593	250436					311029

IV. COMPUTATIONAL COMPLEXITY ANALYSIS

In the training stage, the computational complexity of the algorithm arises from the construction of the decision stumps and strong classifier. For the construction of the decision stumps, all samples should be searched for each feature; thus, the computational complexity for constructing the decision stumps is $O(nM)$, where n is the number of samples, and M is the number of features, i.e., the number of decision stumps. There are T iterations for constructing the strong classifier. Therefore, in the training stage of the AdaBoost algorithm, the computational complexity is only $O(nTM)$. For the SOM-based algorithm or the supervised NN-based algorithms, the computational complexity in the training stage is at least $O(n^2)$ or, in the worst case, $O(n^2M^2)$. For the SVM-based and K-means-based algorithms, the complexity of training is at least $O(nM\log_2(nM))$ or, in the worst case, $O(n^2M^2)$. Thus, the computational complexities of the existing algorithms are higher than the computational complexity of our AdaBoost-based algorithm, particularly when n is very large.

In the testing stage, the computational complexity of our AdaBoost algorithm is also very low. As discussed in Section III-A, there is only one comparison operation in each decision stump for testing a sample; thus, the test time for each decision stump is extremely low. The strong classifier is a combination of decision stumps. As there are T iterations in the construction of the strong classifier, the computational complexity of testing a sample is $O(T)$. Because T is commonly of the same order as the number of features, the test time for our algorithm is very low.

In short, our AdaBoost-based algorithm possesses the lowest computational complexity in the published learning algorithms for intrusion detection. This property is very attractive and promising because the classifiers for intrusion detection should be retrained very quickly in practice, and fast detection is essential for an effective defense against intrusions.

V. EXPERIMENTS

We utilize the Knowledge Discovery and Data Mining CUP 1999 data set [23] to test our algorithm. This intrusion data set originated from the Lincoln Laboratory, Massachusetts Institute of Technology. It was developed for intrusion detection evaluation by the Defense Advanced Research Projects Agency [19]. Despite some drawbacks mentioned in [18], it has served as a reliable benchmark data set for many network-based intrusion detection algorithms. In this data set, each TCP/IP connection was labeled, and 41 continuous or categorical features were extracted.

There are four general types of attack in the data set: denial of service (DOS), user to root (U2R), remote to local (R2L), and PROBE. Detailed descriptions of the four general types of attack can be found in [15] and [19]. The numbers of samples of various types in the training set and those in the test set are listed in Tables I and II, respectively. "Others" in Table II represents the attacks whose types do not appear in the training set.

In all our experiments, the parameters are set as follows: $T = 40$, $T_q = 5$, and $\theta_q = 0.1$. In the following, we first introduce the results with adaptable initial weights and overfitting handling, and we then compare the performance of our algorithm with the published performances of the existing algorithms.

TABLE III
RESULTS WITH UNIFORM INITIAL WEIGHTS AND WITHOUT OVERFITTING HANDLING

Training Set		Test Set	
Detection rate	False alarm rate	Detection rate	False alarm rate
99.159%	2.766%	90.738%	3.428%

TABLE IV
RESULTS WITH ADAPTABLE INITIAL WEIGHTS AND WITHOUT OVERFITTING HANDLING

r	Training Set		Test Set	
	Detection rates	False alarm rates	Detection rates	False alarm rates
0	100%	99.999%	100%	100%
0.1	99.396%	6.325%	91.823%	8.874%
0.2	99.159%	2.766%	90.738%	3.428%
0.3	99.106%	2.724%	90.711%	3.365%
0.4	99.078%	2.712%	90.690%	3.335%
0.5	98.519%	0.851%	90.140%	2.200%
0.6	98.432%	0.402%	90.016%	1.677%
0.7	98.411%	0.143%	89.914%	1.431%
0.8	98.364%	0.142%	89.903%	1.408%
0.9	98.306%	0.586%	89.868%	0.361%
1	0.312%	0%	0.0371%	0.0891%

A. Initial Weights and Overfitting

Table III shows the detection and false-alarm rates for the training and test data sets when uniform initial weights are used and overfitting is not handled. Table IV shows the detection and false-alarm rates when overfitting is not handled, but adaptable initial weights formulated by (13) are used, where r varies from 0 to 1 with the step length of 0.1. It can be seen that, when r is neither too small nor too large, i.e., from 0.3 to 0.7, the results with adaptable initial weights are better than those with uniform initial weights.

Table V shows the detection and false-alarm rates for the training and test data sets when the uniform weights are used and overfitting is handled. From the comparison between Tables III and V, it can be seen that overfitting handling improves both the detection and false-alarm rates.

Table VI shows the detection and false-alarm rates when overfitting is handled and adaptable initial weights are used. From the comparison between Tables IV and VI, it can be seen that, when the value of r is taken between 0.3 and 0.7, the detection rates with overfitting handling are all higher than those without overfitting handling, and the false-alarm rates with overfitting handling are all lower than those without overfitting handling.

B. Selection of r

From a comparison between Tables III and IV, it can be seen that, when $r = 0.5$, the detection rate slightly decreases from 90.738% to 90.140%, but the false-alarm rate has a larger decrease from 3.428% to 2.200%. A similar comparison between Tables V and VI shows the

TABLE V
RESULTS WITH UNIFORM WEIGHTS AND OVERFITTING HANDLING

Training Set		Test Set	
Detection rate	False alarm rate	Detection rate	False alarm rate
99.166%	2.755%	91.207%	3.143%

TABLE VI
RESULTS WITH ADAPTABLE INITIAL WEIGHTS AND OVERFITTING HANDLING

R	Training Set		Test Set	
	Detection rates	False alarm rates	Detection rates	False alarm rates
0	100%	99.999%	100%	100%
0.1	99.188%	6.928%	91.908%	8.971%
0.2	99.166%	5.504%	91.825%	8.212%
0.3	98.956%	2.754%	90.875%	1.787%
0.4	98.855%	0.856%	90.531%	0.682%
0.5	98.791%	0.844%	90.477%	0.665%
0.6	98.791%	0.822%	90.475%	0.668%
0.7	98.445%	0.523%	90.040%	0.307%
0.8	98.346%	0.0884%	89.879%	0.264%
0.9	97.837%	0.0915%	89.539%	0.393%
1	0.312%	0%	0.0371%	0.0891%

TABLE VII
COMPARISON OF DETECTION AND FALSE-ALARM RATES

Methods	False alarm rates	Detection rates
Genetic Clustering [16]	0.3%	79%
Hierarchical SOM [21]	2.19%-3.99%	90.04%-93.46%
SVM [5]	6%-10%	91%-98%
Bagged C5 [4,20]	0.55%	91.81%
RSS-DSS [22]	0.27%-3.5%	89.2%-94.4%
Our algorithm	0.31%-1.79%	90.04%-90.88%

same result. This illustrates that adjustable initial weights are a good way for balancing the detection and false-alarm rates.

In fact, the selection $r = 0.5$ gives the best balance between the detection and false-alarm rates, as it ensures that we pay equal attention to the detection and false-alarm rates, i.e., the normal and attack samples are equally emphasized at the beginning of the algorithm. If a lower false-alarm rate is requested, we can moderately increase r. For example, when $r = 0.7$, the false-alarm rate decreases to 0.307% on the test set, but the detection rate simultaneously decreases to 90.04%. This trend is consistent with the theoretical analysis.

C. Comparisons

In the following, we first compare the detection and false-alarm rates of our algorithm with those published ones of the existing algorithms, which are tested on the benchmark data set, and then compare the running speed of our algorithm with the published running speeds of the existing algorithms.

1) Detection and False-Alarm Rates: The detection and false-alarm rates of our algorithm and those published ones tested on the KDD CUP data set are listed in Table VII. Our results are competitive with others in that the false-alarm rate is kept low without many missed detections.

2) Computational Times: Our AdaBoost-based algorithm is implemented on a Pentium IV computer with 2.6-GHz CPU and 256-MB RAM, using MATLAB 7. The mean training time is only 73 s, using all 494 021 training samples. This is an empirical substantiation that the computational complexity of our algorithm is particularly low. In [12], the least training times of the SOM and the improved competitive neural network are 1057 and 454 s, respectively, using only 101 000 samples for training. The Bagged C5 algorithm [4], [20] took slightly more than a day on a machine with a two-processor UltraSparc2 (2–300 MHz) and 512-MB main memories. The random subset selection-dynamic subset selection (RSS–DSS) algorithm [22] needs 15 min to finish the learning process on a 1-GHz Pentium III laptop with 256-MB RAM. In [52], 8 min is taken for processing the training data in an eight-node cluster, where each node has dual 1-GHz Pentium III processors and 1-GB memory, running Red Hat Linux7.2, whereas 212 min is taken for the algorithm in [53].

From the aforementioned comparisons, we find that our algorithm has the best running speed, which is an important property in practice. This is consistent with the earlier analysis of computational complexity in Section IV.

D. Limitation

It is noted that the implementation of our AdaBoost-based intrusion detection algorithm is not amenable to incremental learning. When the system changes over time, the newly produced data should be labeled and merged with the previous sample data, and the classifier is retrained using the merged sample data. Although our algorithm has a very low computational complexity, which makes it possible to frequently retrain the classifier, offline learning is still a limitation when it is necessary to adapt to complicated and changing network environments.

VI. CONCLUSION

We have proposed an AdaBoost-based algorithm for intrusion detection. In the algorithm, decision stumps are used as weak classifiers. The decision rules are provided for both categorical and continuous features. The relations between categorical and continuous features are handled naturally, without any forced conversions between these two types of features. A simple overfitting handling is used to improve the learning results. In the specific case of network intrusion detection, we use adaptable initial weights to make the tradeoff between the detection and false-alarm rates. The experiment results show that our algorithm has a very low false-alarm rate with a high detection rate, and the run speed of our algorithm is faster in the learning stage compared with the published run speeds of the existing algorithms. The experimental results illustrate that our algorithm has a competitive performance, as compared with the published intrusion detection algorithms, as tested on the benchmark sample data.

Our future work will focus on the following aspects.

1) We will find new types of weak classifiers to further improve the error rates.
2) We will investigate the application of online versions [55], [56] of boosting algorithms to incremental learning for network intrusion detection in order to implement online updating of the model in the presence of streaming data.

REFERENCES

[1] S. Chebrolu, A. Abraham, and J. P. Thomas, "Feature deduction and ensemble design of intrusion detection systems," *Comput. Secur.*, vol. 24, no. 4, pp. 295–307, Jun. 2005.
[2] D. Denning, "An intrusion detection model," *IEEE Trans. Softw. Eng.*, vol. SE-13, no. 2, pp. 222–232, Feb. 1987.
[3] M. O. Depren, M. Topallar, E. Anarim, and K. Ciliz, "Network-based anomaly intrusion detection system using SOMs," in *Proc. IEEE 12th Signal Process. Commun. Appl. Conf.*, Apr. 2004, pp. 76–79.

[4] C. Elkan, "Results of the kdd99 classifier learning contest," *SIGKDD Explor.*, vol. 1, no. 2, pp. 63–64, 2000.

[5] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo, "A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data," in *Applications of Data Mining in Computer Security*, D. Barbara and S. Jajodia, Eds. Norwell, MA: Kluwer, 2002, ch. 4.

[6] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, Aug. 1997.

[7] P. Hong and R. E. Schapire, "An intrusion detection method based on rough set and SVM algorithm," in *Proc. Int. Conf. Commun., Circuits Syst.*, Jun. 2004, vol. 2, pp. 1127–1130.

[8] H. Jin, J. Sun, H. Chen, and Z. Han, "A fuzzy data mining based intrusion detection model," in *Proc. l0th IEEE Int. Workshop Future Trends Distrib. Comput. Syst.*, May 2004, pp. 191–197.

[9] H. G. Kayacik, A. N. Zincir-Heywood, and M. I. Heywood, "On the capability of an SOM based intrusion detection system," in *Proc. Int. Joint Conf. Neural Netw.*, Jul. 2003, vol. 3, pp. 1808–1813.

[10] W. Lee and S. J. Stolfo, "A framework for constructing features and models for intrusion detection systems," *ACM Trans. Inf. Syst. Secur.*, vol. 3, no. 4, pp. 227–261, Nov. 2000.

[11] W. Lee, S. J. Stolfo, and K. Mok, "A data mining framework for building intrusion detection models," in *Proc. IEEE Symp. Secur. Priv.*, May 1999, pp. 120–132.

[12] J. Z. Lei and A. Chorbani, "Network intrusion detection using an improved competitive learning neural network," in *Proc. 2nd Annu. Conf. Commun. Netw. Serv. Res.*, May 2004, vol. 4, pp. 190–197.

[13] J. Li and C. Manikopoulos, "Novel statistical network model: The hyperbolic distribution," *Proc. Inst. Electr. Eng.—Commun.*, vol. 151, no. 6, pp. 539–548, Dec. 2004.

[14] Z. W. Li, A. Das, and S. Nandi, "Utilizing statistical characteristics of N-grams for intrusion detection," in *Proc. Int. Conf. Cyberworlds*, Dec. 2003, pp. 486–493.

[15] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das, "The 1999 DARPA off-line intrusion detection evaluation," *ACM Trans. Inf. Syst. Secur.*, vol. 34, no. 4, pp. 579–595, Oct. 2000.

[16] Y. G. Liu, K. F. Chen, X. F. Liao, and W. Zhang, "A genetic clustering method for intrusion detection," *Pattern Recognit.*, vol. 37, no. 5, pp. 927–942, May 2004.

[17] Y.-H. Liu, D.-X. Tian, and A.-M. Wang, "Annids: Intrusion detection system based on artificial neural network," in *Proc. Int. Conf. Mach. Learn. Cybern.*, Nov. 2003, vol. 3, pp. 1337–1342.

[18] J. McHugh, "Testing intrusion detection systems: A critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory," *ACM Trans. Inf. Syst. Secur.*, vol. 3, no. 4, pp. 262–294, Nov. 2000.

[19] S. Mukkamala, A. H. Sung, and A. Abraham, "Intrusion detection using an ensemble of intelligent paradigms," *J. Netw. Comput. Appl.*, vol. 28, no. 2, pp. 167–182, Apr. 2005.

[20] B. Pfahringer, "Winning the kdd99 classification cup: Bagged boosting," *SIGKDD Explor.*, vol. 1, no. 2, pp. 65–66, 2000.

[21] S. T. Sarasamma, Q. A. Zhu, and J. Huff, "Hierarchical Kohonenen net for anomaly detection in network security," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 35, no. 2, pp. 302–312, Apr. 2005.

[22] D. Song, M. I. Heywood, and A. N. Zincir-Heywwd, "Training genetic programming on half a million patterns: An example from anomaly detection," *IEEE Trans. Evol. Comput.*, vol. 9, no. 3, pp. 225–239, Jun. 2005.

[23] S. Stolfo, *The Third International Knowledge Discovery and Data Mining Tools Competition*. Univ. California, 2002. [Online]. Available: http://kdd.ics.uci.edu/databases/kddCup99/kddCup99.html

[24] C. Zhang, J. Jiang, and M. Kamel, "Intrusion detection using hierarchical neural networks," *Pattern Recognit. Lett.*, vol. 26, no. 6, pp. 779–791, May 2005.

[25] Z. Zhang and H. Shen, "Online training of SVMs for real-time intrusion detection," in *Proc. Int. Conf. Adv. Inf. Netw. Appl.*, 2004, vol. 1, pp. 568–573.

[26] G. Vigna and R. A. Kemmerer, "NetSTAT: A network-based intrusion detection approach," in *Proc. Comput. Secur. Appl. Conf.*, Dec. 1998, pp. 25–34.

[27] D. Qu, B. M. Vetter, F. Wang, R. Narayan, S. F. Wu, Y. F. Hou, F. Gong, and C. Sargor, "Statistical anomaly detection for link-state routing protocols," in *Proc. 6th Int. Conf. Netw. Protocols*, 1998, pp. 62–70.

[28] N. Ye, S. M. Emran, X. Li, and Q. Che, "Statistical process control for computer intrusion detection," in *Proc. DARPA Inf. Survivability Conf. Expo. II*, 2001, vol. 1, pp. 3–14.

[29] J. B. D. Caberera, B. Ravichandran, and R. K. Mehra, "Statistical traffic modeling for network intrusion detection," in *Proc. Model., Anal. Simul. Comput. Telecommun. Syst.*, 2000, pp. 466–473.

[30] N. Ye, Y. Zhang, and C. M. Borror, "Robustness of the Markov-chain model for cyber-attack detection," *IEEE Trans. Rel.*, vol. 53, no. 1, pp. 116–123, Mar. 2004.

[31] D. Yeung and Y. Ding, "Host-based intrusion detection using dynamic and static behavioral models," *Pattern Recognit.*, vol. 36, no. 1, pp. 229–243, Jan. 2003.

[32] X. A. Hoang and J. Hu, "An efficient hidden Markov model training scheme for anomaly intrusion detection of server applications based on system calls," in *Proc. IEEE Int. Conf. Netw.*, 2004, vol. 2, pp. 470–474.

[33] Y. Freund and R. Schapire, "Experiments with a new boosting algorithm," in *Proc. 13th Int. Conf. Mach. Learn.*, 1996, pp. 148–156.

[34] H. Han, X. L. Lu, and L. Y. Ren, "Using data mining to discover signatures in network-based intrusion detection," in *Proc. Int. Conf. Mach. Learn. Cybern.*, 2002, vol. 1, pp. 13–17.

[35] M. Qin and K. Hwang, "Frequent episode rules for Internet anomaly detection," in *Proc. IEEE Int. Symp. Netw. Comput. Appl.*, 2004, pp. 161–168.

[36] J. M. Bonifacio, Jr., A. M. Cansian, A. C. P. L. F. De Carvalho, and E. S. Moreira, "Neural networks applied in intrusion detection systems," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, 1998, vol. 1, pp. 205–210.

[37] A. Rapaka, A. Novokhodko, and D. Wunsch, "Intrusion detection using radial basis function network on sequences of system calls," in *Proc. Int. Joint Conf. Neural Netw.*, 2003, vol. 3, pp. 1820–1825.

[38] S. J. Han and S. B. Cho, "Evolutionary neural networks for anomaly detection based on the behavior of a program," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 36, no. 3, pp. 559–570, Jun. 2006.

[39] S. Mukkamala, G. Janoski, and A. H. Sung, "Intrusion detection using neural networks and support vector machines," in *Proc. Int. Joint Conf. Neural Netw.*, 2002, vol. 2, pp. 1702–1707.

[40] A. H. Sung and S. Mukkamala, "Identifying important features for intrusion detection using support vector machines and neural networks," in *Proc. Symp. Appl. Internet*, 2003, pp. 209–216.

[41] J. Mill and A. Inoue, "Support vector classifiers and network intrusion detection," in *Proc. Int. Conf. Fuzzy Syst.*, 2004, vol. 1, pp. 407–410.

[42] T. Abbes, A. Bouhoula, and M. Rusinowitch, "Protocol analysis in intrusion detection using decision tree," in *Proc. Int. Conf. Inf. Technol.: Coding Comput.*, 2004, vol. 1, pp. 404–408.

[43] M. Asaka, T. Onabura, T. Inoue, and S. Goto, "Remote attack detection method in IDA: MLSI-based intrusion detection using discriminant analysis," in *Proc. Symp. Appl. Internet*, 2002, pp. 64–73.

[44] S. Mukkamala and A. H. Sung, "A comparative study of techniques for intrusion detection," in *Proc. Int. Conf. Tools Artif. Intell.*, 2003, pp. 570–577.

[45] Y. Guan, A. A. Ghorbani, and N. Belacel, "Y-means: A clustering method for intrusion detection," in *Proc. IEEE Can. Conf. Electr. Comput. Eng.*, 2003, vol. 2, pp. 1083–1086.

[46] J. Xian, F. Lang, and X. Tang, "A novel intrusion detection method based on clonal selection clustering algorithm," in *Proc. Int. Conf. Mach. Learn. Cybern.*, 2005, vol. 6, pp. 3905–3910.

[47] S. Jiang, X. Song, H. Wang, J. Han, and Q. Li, "A clustering-based method for unsupervised intrusion detections," *Pattern Recognit. Lett.*, vol. 27, no. 7, pp. 802–810, May 2006.

[48] A. J. Hoglund, K. Hatonen, and A. S. Sorvari, "A computer host-based user anomaly detection system using the self-organizing map," in *Proc. Int. Joint Conf. Neural Netw.*, 2000, vol. 5, pp. 411–416.

[49] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. Int. Conf. Comput. Vis. Pattern Recog.*, 2001, vol. 1, pp. I-511–I-518.

[50] Y. Freund and R. Schapire, "A short introduction to boosting," *J. Jpn. Soc. Artif. Intell.*, vol. 14, no. 5, pp. 771–780, 1999.

[51] C. Rudin, I. Daubechies, and R. E. Schapire, "The dynamics of AdaBoost: Cyclic behavior and convergence of margins," *J. Mach. Learn. Res.*, vol. 5, pp. 1557–1595, Dec. 2004.

[52] M. E. Otey, A. Ghoting, and S. Parthasarathy, "Fast distributed outlier detection in mixed-attribute data sets," *Data Min. Knowl. Discov.*, vol. 12, no. 2/3, pp. 203–228, May 2006.

[53] S. Bay and M. Schwabacher, "Mining distance-based outliers in near linear time with randomization and a simple pruning rule," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, 2003, pp. 29–38.

[54] W. Hu and W. M. Hu, "HIGCALS: a hierarchical graph-theoretic clustering active learning system," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, 2006, vol. 5, pp. 3895–3900.

[55] N. C. Oza, "Online ensemble learning," Ph.D. dissertation, Univ. California at Berkeley, Berkeley, CA, 2001.

[56] A. Fern and R. Givan, "Online ensemble learning: An empirical study," in *Proc. 17th Int. Conf. Mach. Learn.*, 2000, pp. 279–286.

[57] W. Hu and W. M. Hu, "Network-based intrusion detection using Adaboost algorithm," in *Proc. IEEE/WIC/ACM Int. Conf. Web Intell.*, Sep. 2005, pp. 712–717.