

Application of Variant of AdaBoost-Based Machine Learning Algorithm in Network Intrusion Detection

V.P.Kshirsagar

*Lecturer, Department of CSE
Government College of Engineering
Aurangabad, MS, India.*

Dharmaraj R.Patil

*M.E (CSE) Part-VI, Department of CSE
Government College of Engineering
Aurangabad, MS, India.*

dharmaraj.rcpit@gmail.com

Abstract

In recent years of Information Technology network-based technology is advances and it is important to assure reliable and secure operation of the system. A lot of increase in malicious activities across the network has been observed due to which both industry and research community have brought more emphasis on solving network intrusion detection problems. Network intrusion detection deals with the classification problem and various machine learning techniques providing promising approach in intrusion detection. One such machine learning algorithm is AdaBoost, which have a better classification accuracy. The AdaBoost algorithm combines the weak classifiers for continuous features and weak classifiers for categorical features into a strong classifier which gives more correct classification results. We have designed the NIDS based on AdaBoost machine learning algorithm using Java Technology and tested it on the NSL-KDD intrusion detection dataset. This paper is focused on the application of variant of AdaBoost-based machine learning algorithm in Network Intrusion Detection System.

Area: Network Security

Keywords: Machine Learning Algorithms, Network Intrusion Detection Systems, AdaBoost Algorithm, Detection Rate, False Alarm Rate, KDDCup'99 Dataset, NSL-KDD Dataset.

1. INTRODUCTION

An intrusion is somebody ("hacker" or "cracker") attempting to break into or misuse your system. The word "misuse" is broad and can reflect something severe as stealing confidential data to something minor such as misusing your email system for spam. An "Intrusion Detection System (IDS)" is a system for detecting such intrusions. There are two types of intrusion detection systems namely Host-based systems base their decisions on the information obtained from a single host and Network-based intrusion detection systems obtain data by monitoring the traffic in the network to which the hosts are connected [1].

1.1 Host-based Intrusion Detection Systems

Host-based IDS's are installed on the host they are intended to monitor. The host can be a server, workstation or any networked device. HIDS's install as a service or daemon or they modify the underlying operating systems kernel or application to gain first inspection authority. While a HIDS may include the ability to sniff network traffic intended for the monitored host. Application attacks can include memory modifications, maliciously crafted application requests, buffer overflows or file-modification attempts. A HIDS can inspect each incoming command, looking for signs or maliciousness or simply track unauthorized file changes.

1.2 Network-based Intrusion Detection Systems

Network-based IDS's are work by capturing and analyzing network packets speeding by on the wire. Unlike, HIDS NIDS are designed to protect more than one host. They can protect a group of computer hosts, like a server farm, or monitor an entire network. Captured traffic is compared against protocol specifications and normal traffic trends or the packets payload data is examined for malicious content. If a security threat is noted, the event is logged and an alert is generated. Figure 1 shows a network with Network Intrusion Detection System [9].

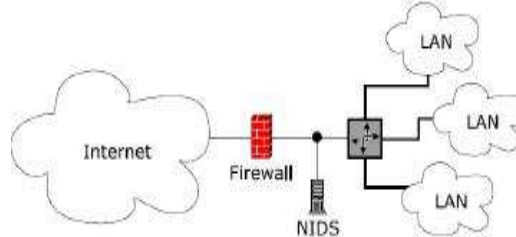


Figure 1: Network-based Intrusion Detection System

1.3 Features of Network Intrusion Detection System

Some of the important features of a Network Intrusion Detection System are as follows [1],

- It should be fault tolerant and run continuously with minimal human supervision.
- A Network Intrusion Detection System must be able to recover from the crashes, either accidental or caused by malicious activity.
- A Network Intrusion Detection System must be able to detect any modifications forced on the IDS by an attacker.
- It should impose minimal overhead on the system.
- It should be configurable so as to accurately implement the security policies of the system.
- It should be easy to use by the operator.
- It should be capable to detect different types of attacks and must not recognize and legitimate activity as an attack.

2. MODEL OF THE NIDS USING ADABOOST-BASED ALGORITHM

Considering the characteristics of the AdaBoost algorithm and characteristics of intrusion detection system, the model of the system consists of four parts: feature extraction, data labeling, and design of weak classifiers and construction of the strong classifier as shown in the figure 2 [3].

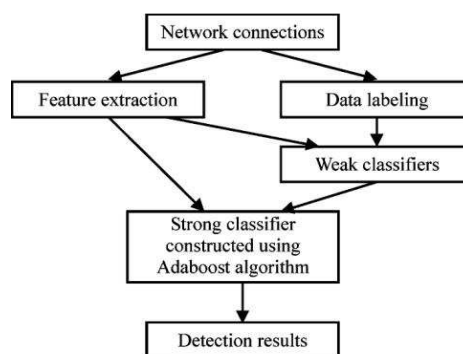


Figure 2: Model of NIDS using AdaBoost algorithm

2.1 Feature Extraction

For each network connection, contains 41 features and can be classified into three groups [5].

i. Basic features

This category encapsulates all the attributes that can be extracted from a TCP/IP connection.

ii. Traffic Features

This category includes features that are computed with respect to a window interval and is divided into two groups [5].

a. Same Host features

These features examine only the connections in the past 2 seconds that have same destination host as the current connection, and calculate statistics related to protocol behavior, service etc.

b. Same Service features

These features examine only the connections in the past 2 seconds that have the same service as the current connection.

iii) Content Features

Unlike most of the DoS and probing attacks, the R2L and U2R attacks don't have any intrusion patterns. This is because the DoS and probing attacks involves many connections to the same host in a very short period of time, however the R2L and U2R attacks are embedded in the data portions of the packets and normally involved only a single connection. To detect these kind of attacks, we need some features to be able to look for suspicious behavior in the data portion. These features are called content features [5].

2.2 Data Labeling

The AdaBoost algorithm labels a set of data as either normal or an attack. The normal data samples are labeled as "+1" and attack data samples are labeled as "-1".

2.3 Design of Weak Classifiers

For classification of the intrusive data, the AdaBoost algorithm requires a group of weak classifiers. The weak classifier's classification accuracy is relatively low.

2.4 Construction of Strong Classifier

In AdaBoost algorithm a strong classifier is constructed by combining the weak classifiers. The strong classifier has high classification accuracy than each weak classifier. The strong classifier is then trained using training sample data. Then a test data sample is input to the strong classifier to test it as a "normal" or "attack" sample.

3. MACHINE LEARNING ALGORITHMS AND ADABOOST

3.1 Boosting

Boosting is general method for improving the accuracy of any given learning algorithm. Boosting refers to a general and provably effective method of producing a very accurate prediction rule by combining rough and moderately inaccurate rules. Boosting has its roots in a theoretical framework for studying machine learning called the "PAC" learning model [10]. With the help of boosting a "weak" learning algorithm can be "boosted" into an arbitrarily accurate "strong" learning algorithm. Here decision stumps are used as weak learning learners. They can be combined into a strong learning algorithm for better classification accuracy [2].

3.2 Introduction to AdaBoost Algorithm

The AdaBoost algorithm, introduced in 1995 by Freund and Schapire [11], solved many of the practical difficulties of the earlier boosting algorithms. The algorithm takes as input a training set $(x_1, y_1) \dots (x_m, y_m)$ where x_i belongs to some domain or instance of space X , and each label y_i is in some label set Y . Assume $Y = \{-1, +1\}$. AdaBoost calls a given weak or base learning algorithm, here decision stump repeatedly in a series of rounds $t=1 \dots T$. One of the main ideas of the algorithm is to maintain a distribution or set of weights over the training set. The weights of this distribution on training example i on round t is denoted $D_t(i)$. Initially all weights are set equally, but on each round the weights of incorrectly classified examples are increased so that the weak learner is forced to focus on the hard examples in the training set. The weak learner's job is to find a weak hypothesis [2], $h_t: X \rightarrow \{-1, +1\}$ appropriate for the distribution D_t . The goodness of a weak classifier is measured by its error,

$$\begin{aligned} \epsilon_t &= \Pr_{i \sim D_t}[h_t(x_i) \neq y_i] \\ &= \sum_{i: h_t(x_i) \neq y_i} D_t(i) \end{aligned} \quad (1)$$

AdaBoost works by combining several "votes". Instead of using support vectors, AdaBoost uses weak learners [4],

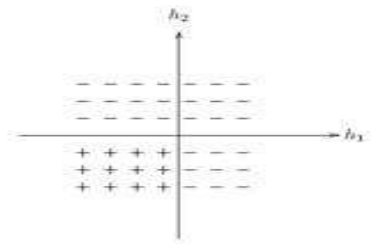


FIGURE 3: Neither h_1 nor h_2 is a perfect learner; AdaBoost combines them to obtain a “good” learner. Figure illustrates how AdaBoost combines two learners, h_1 and h_2 . It initially chooses the learner that classifies more data correctly. In the next step, the data is re-weighted to increase the “importance” of misclassified samples. This process continues and at each step the weight of each weak learner among other learners is determined. Therefore the algorithm is as follows:

1. Set all sample weights equal, and find h_1 to maximize $\sum_i y_i h(x_i)$ (2)
2. Perform re-weighting to increase the weight of the misclassified samples.
3. Find the next h to maximize $\sum_i y_i h(x_i)$. Find the weight of this classifier, α . (3)
4. Go to step 2. The final classifier will be: $\text{sgn}(\sum_{i=0}^T \alpha_i h_i(x))$ (4)

The strong classifier has good classification accuracy as compared to the weak classifiers and hence gives better detection results.

4. NSL-KDD INTRUSION DETECTION DATA SET

NSL-KDD is a dataset suggested to solve some of the inherent problems of the KDD'99 dataset [6], [7]. The NSL-KDD dataset has the following advantages over the original KDD'99 dataset.

1. It does not include redundant records in the training set, so the classifiers will not be biased towards more frequent records.
2. There are no duplicate records in the proposed test sets, therefore the performance of the learners are not biased by the methods which have better detection rates on the frequent records.
3. The number of selected records from each difficulty level group is inversely proportional to the percentage of the records in the original KDD dataset. As a result the classification rates of distinct machine learning methods vary in a wider range, which makes it more efficient to have an accurate evaluation of different learning techniques.
4. The number of records in the training and testing sets are reasonable, which makes it affordable to run the experiments on the complete set without the need to randomly select a small portion.
5. Statistical observations: one of the most important deficiencies in the KDD dataset is the huge number of redundant records, which causes the learning algorithms to be biased towards the frequent records and thus prevent them from learning unfrequent records which are usually more harmful to networks such as U2R and R2L attacks.

Table 1 and Table 2 show the statistics of the redundant records in the KDD Cup'99 training and testing datasets.

	Original Records	Distinct Records	Reduction Rate
Attacks	3,925,650	262,178	93.32%
Normal	972,781	812,814	16.44%
Total	4,898,431	1,074,992	78.05%

Table 1: Statistics of redundant records in the KDD TRAINING DATASET [6]

	Original Records	Distinct Records	Reduction Rate
Attacks	250,436	29,378	88.26%
Normal	60,591	47,911	20.92%
Total	311,027	77,289	75.15%

Table 2: Statistics of redundant records in the KDD testing Dataset [6]

5. EXPERIMENTAL ANALYSIS

We utilize the NSL-Knowledge Discovery and Data Mining data set [7] to test the system. The NIDS based on AdaBoost-based Machine Learning algorithm is implemented on a Pentium IV computer with 2.6-GHz CPU and 1 GB RAM, using Java Technology. We have taken 20% of the training dataset of NSL-KDD as input to the system which contains about 125,973 connection records to train the system and a test dataset contains 22,544 connection records to test the system. The figure 4 shows the detection results on NSL-KDD testing datasets. Figure 5 shows the comparison of the classification performance of various learning algorithms on the NSL-KDD test dataset. The detection performance of our system is 90.31 %, which is very good as compare to other learning algorithms. Due to the redundant records in the KDDCup'99 dataset the performance of the learning algorithms biased. The results of the accuracy and performance of learning algorithms on the KDDCup'99 data set are hence unreliable. NSL-KDD testing set provide more accurate information about the capability of the classifiers

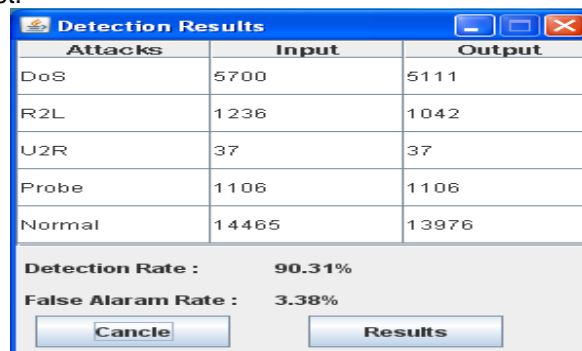
Two indices are commonly used to judge the accuracy of a network intrusion detection system. One is detection rate (DR) [12],

$$DR = \frac{\#Detected\ Attacks}{\#All\ Attacks} \quad (5)$$

And the other is false alarm rate :

$$False\ Alarm\ Rate = 1 - \frac{\#Detected\ Normals}{\#All\ Normals} \quad (6)$$

The DR and False Alarm Rate of the AdaBoost-Based NIDS learning system are 90.31% and 3.38% respectively. This result is competitive with other published results of learning algorithms on NSL-KDD dataset.



Attacks	Input	Output
DoS	5700	5111
R2L	1236	1042
U2R	37	37
Probe	1106	1106
Normal	14465	13976

Detection Rate : 90.31%

False Alarm Rate : 3.38%

Buttons: Cancel, Results

Figure 4: Detection Results of AdaBoost-based NIDS on NSL-KDD Test Dataset

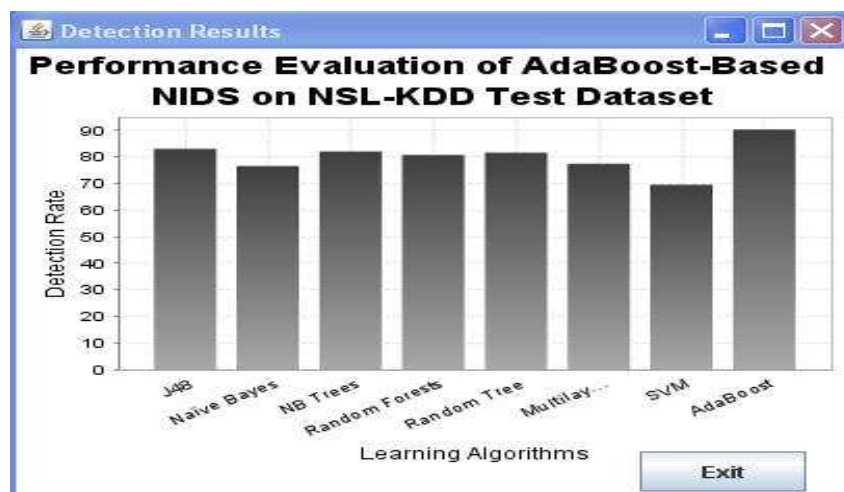


Figure 5: Comparison of performance of AdaBoost-based NIDS with other learning algorithms.

6. CONSLUSION

This paper is deals with the application of variant of AdaBoost-based machine learning algorithm in network intrusion detection. We have designed this system using Java Technology and the NSL-KDD intrusion detection dataset. Our experimental results are competitive with other published results of learning algorithms on NSL-KDD dataset given in the literature. The false alarm rate of the system is also low and considerable.

7. REFERENCES

- [1] S.Chebrolu, A. Abraham and J.P.Thomos, "Features deduction and Ensemble design of Intrusion Detection Systems", Computer Security, Vol.24, Sept.2004.
- [2] Y. Freund, R. E. Schapire, " A short Introduction to Boosting" Journal of Japanese Society For Artificial Intelligence, Sept.1999.
- [3] Weiming Hu, Wei Hu, "AdaBoost-Based Algorithm for Network Intrusion Detection", IEEE Transactions on Systems, Man and Cybernetics-Part B, Cybernetics- Vol.38, April 2008.
- [4] AdaBoost Algorithm, *Hussein Falaki*, Course Notes.
- [5] KDDCup 1999 Data, <http://www.kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>, 1999.
- [6] M. Tavallaee, E. Bagheri, W. Lu, and A. Ghorbani, "A Detailed Analysis of the KDDCup'99 Data Set," Submitted to Second IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA), 2009.
- [7] "Nsl-kdd data set for network-based intrusion detection Systems."Available on: <http://nsl.cs.unb.ca/NSL-KDD/>, March 2009.
- [8] Elkan, Charles, "Results of the KDD'99 classifier learning" ,SIGKDD Exploring, 2000.
- [9] Wafa S. Al -Sharafat, Reyadh Sh. Naoum, " Adaptive Framework for Network Intrusion Detection by Using Genetic –Based Machine Learning Algorithm", IJCSNS International Journal of Computer Science and Network Security, VOL.9 No.4, April 2009.
- [10] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, November 1984.
- [11] Yoav Freund and Robert E. Schapire. A decision theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, August 1997.
- [12] 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