International Journal of Research in Advent Technology, Vol.2, No.2, April 2014
E-ISSN: 2321-9637

A Comparative Performance Evaluation of Machine Learning-Based NIDS on Benchmark Datasets

Dharmaraj R.Patil[1], *Tareek M.Pattewar*[2]
Department of Computer Engineering, R.C.Patel Institute of Technology, Shirpur, M.S., India.
Email: dharmaraj.rcpit@gmail.com[1]
Department of Information Technology, R.C.Patel Institute of Technology, Shirpur, M.S., India.
Email: tareekpattewar@gmail.com[2]

Abstract- As network-based computer systems play increasingly vital roles in modern society, they have become the targets of malicious activities, which both industry and research community have brought more emphasis on solving network intrusion detection problems. Machine learning algorithms have proved to be an important tool in network intrusion detection problems. In this paper we have presented an application of AdaBoost-based machine learning algorithm in network intrusion detection. Network intrusion detection deals with the classification problem and AdaBoost-based algorithm have good classification accuracy. As well as AdaBoost-based algorithm have high detection rate and low false-alarm rate. This algorithm combines the weak classifiers for continuous features and weak classifiers for categorical features into a strong classifier. We have developed the AdaBoost-based NIDS and tested the system on KDDCup'99 and NSL-KDD intrusion detection datasets. A comparative performance evaluation of the NIDS on both the datasets are shown. The experimental results show that AdaBoost-based NIDS performance on NSL-KDD dataset is very good as compare to KDDCup'99 dataset.

Index Terms- Machine Learning, Network Intrusion Detection, AdaBoost Algorithm, Detection Rate, False-alarm Rate.

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.

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. MATERIALS AND METHODS

2.1 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 [8]. 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].

2.2 Introduction to AdaBoost Algorithm

The AdaBoost algorithm, introduced in 1995 by Freund and Schapire [9], solved many of the practical difficulties of the earlier boosting algorithms. The algorithm takes as input a training set $(x_1, y_1)\dots(x_m, ym)$ where xi 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… 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,

$$\epsilon t = PriDt\,[ht(xi) + yi] = \sum_{i:ht(xi)=yi} Dt\,(i) \qquad (1)$$

AdaBoost works by combining several “votes”. Instead of using support vectors, AdaBoost uses weak learners.

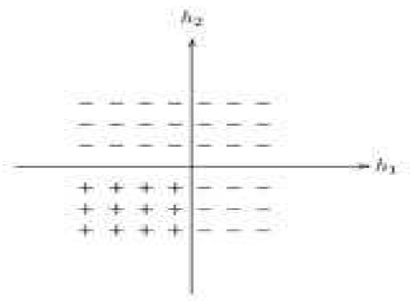

Fig.1: Neither h1 nor h2 is a perfect learner; AdaBoost combines them to obtain a “good” learner

Figure illustrates how AdaBoost combines two learners, h1 and h2. 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.

2.3 Introduction to Weak Classifiers

Here decision stumps are used as weak classifiers. A decision stump is a decision tress with a root node and two leaf nodes. For each feature in the input data, a decision stump is constructed [3].The decision stumps for categorical features and decision stumps for continuous features are given as follows,

2.3.1 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 no overlapping 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 follows [3],

$$h_f(X) = \begin{cases} +1, Xf \in C_p^f \\ -1, Xf \in C_n^f \end{cases} \qquad (2)$$

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 $\hat{C}_p^f$ and $\hat{C}_n^f$ that correspond to the optimal decision stump $\hat{h}_f$ are determined by minimizing the sum of the false classification rates for the normal and attack samples

$(\hat{C}_p^f, \hat{C}_n^f)$ =arg min

$$_{(C_p^f,\, C_n^f)}[\varepsilon + h_f(C_p^f, C_n^f) + \varepsilon - hf(C_p^f, C_n^f)]. \qquad (3)$$

2.3.2 Decision stumps for continuous features

For a continuous feature *f*, given a segmentation value θ, a decision stump h_f can be constructed as [3],

$$h_f(X) = \begin{cases} +1, Xf \le \theta \\ -1, Xf > \theta \end{cases} \qquad (4)$$

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

2.4 Working of the Algorithm

The algorithm works as follows [3],

1) Initialize weights *wi* (1) (*i* = 1. . . *n*) satisfying $\sum_{i=1}^{n} wi(1) = 1.$

2) Observe the following for (*t* = 1. . . *T*).

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

$\varepsilon_j = \sum_{i=1}^{n} wi(t) I[yi \neq hj(xi)]$ (5)

where,

$I[\gamma] = \begin{cases} 1, \gamma = True, \\ 0, \gamma = False. \end{cases}$ (6)

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

$h(t) = \arg\min_{hj \in H} \varepsilon j$ (7)

b) Calculate the sum of the weighted classification errors ε (*t*) for the chosen weak classifier *h* (*t*).

c) Let

$\alpha(t) = ½ log(1 - \varepsilon(t)/\varepsilon(t))$ (8)

d) Update the weights by

$w_i(t+1) = \frac{wi(t) exp(-\alpha(t) yih(t)(xi))}{Z(t)}$ (9)

where *Z* (*t*) is a normalization factor,

$Z(t) = \sum_{k=1}^{n} wk(t) exp(-\alpha(t) yih(t)(xk))$ (10)

3) The strong classifier is defined by

$H(x) = sign(\sum_{t=1}^{T} \alpha(t) h(t)(x))$ (11)

3. ARCHITECTURE OF 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].

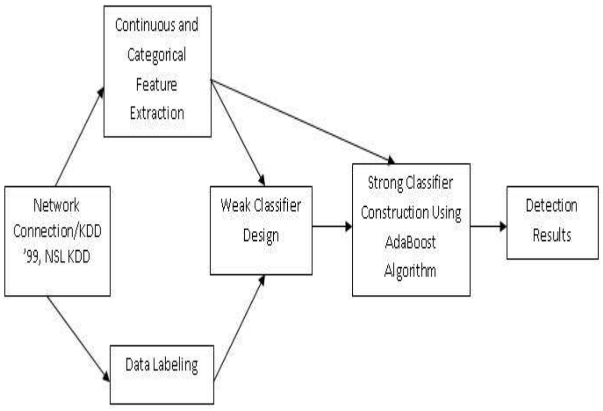
Continuous and Categorical Feature Extraction
Network Connection/KDD '99, NSL KDD
Weak Classifier Design
Strong Classifier Construction Using AdaBoost Algorithm
Detection Results
Data Labeling

Fig. 2. Architecture of NIDS using AdaBoost algorithm.

3.1 Feature Extraction

For each network connection, contains 41 features and can be classified into three groups,

3.1.1 Basic features

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

3.1.2. Traffic Features

This category includes features that are computed with respect to a window interval and is divided into two groups,

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.

3.1.3 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.

3.2 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".

3.3 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.

3.4 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.

4. KDDCUP'99 AND NSL-KDD DATASETS

4. 1 KDD Cup'99 Dataset

This data set was derived from the 1998 DARPA Intrusion Detection Evaluation Program held by MIT Lincoln Labs. The dataset was created and simulated in a military network environment in which a typical U.S. Air Force LAN was subjected to simulated attacks. Raw TCP/IP dump data was gathered. The data is approximately 4 GB of compressed TCP dump data which took 7 weeks of network traffic and comprised about 5 million connection records. For each TCP/IP connection 41 various quantitative and qualitative features were extracted.KDD dataset is divided into training and testing records sets. The attacks include the four most common categories of attacks [4], [5] given as follows,

4.1. 1 Denial of Service Attacks (Dos)

It is an attack in which the attacker makes some computing or memory resource to busy or too full to handle legitimate requests, or denies legitimate users access to a machine. e.g. back, Neptune, land etc.

4.1.2 User to Root Attack (U2R)

It is a class of exploit in which the attacker starts out with access to a normal user account on the system and is able to exploit some vulnerability to gain root access to the system. e.g. loadmodule, perl, ps etc.

4.1.3 Remote to Login Attack (R2L)

This attack occurs when an attacker who has the ability to send packets to a machine over a network but who does not have an account on that machine exploits some vulnerability to gain local access as a user of that machine. e.g. ftpwrite, httptunnel, imap etc.

4.1.4 Probing Attack

It is an attempt to gather information about a network of computers for the apparent purpose of circumventing its security controls. e.g. nmap, satan, mscan etc.

4.2 NSL-KDD Dataset

NSL-KDD is a dataset suggested to solve some of the inherent problems of the KDD'99 dataset [5], [6].The NSL-KDD dataset has the following advantages over the original KDD'99 dataset.

i) It is not include redundant records in the training set, so the classifiers will not be biased towards more frequent records.

ii) 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.

iii) 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.

iv) 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.

v) 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 I
Statistics of redundant records in the KDD Training Dataset [5]

	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 II
Statistics of redundant records in the KDD testing Dataset [5]

	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 I and Table II shows the statistics of the redundant records in the KDD Cup'99 training and testing datasets

5. EXPERIMENTAL ANALYSIS

The system is developed on Pentium IV Computer with 2.6 GHz and 1 GB RAM, using JDK 1.6. We utilize the KDDCup'99 and NSL-KDD datasets [4], [6] to test the Network Intrusion Detection System Using AdaBoost-based machine learning algorithm. We have taken 10% training and testing data from the KDDCup'99 data to test the system. The results are given in figure 4.We have taken 20 % of the NSL-KDD training dataset as input to train the system and NSL-KDD testing dataset to test the system. The results are given in figure 5.The comparative performance of other learning algorithms with AdaBoost algorithm on KDDCup'99 and NSL-KDD datasets are given in figure 4 and figure 5. Figure 6 illustrates the comparative performance of the system on KDDCup'99 and NSL-KDD datasets. Two indices are commonly used to judge the accuracy of a network intrusion detection system. One is detection rate (DR) [10],

$$DR = \frac{DetectedAttacks}{AllAttacks} \qquad (12)$$

And the other is false alarm rate:

$$\text{False Alarm Rate} = 1 - \frac{DetectedNormals}{AllNormals} \qquad (13)$$

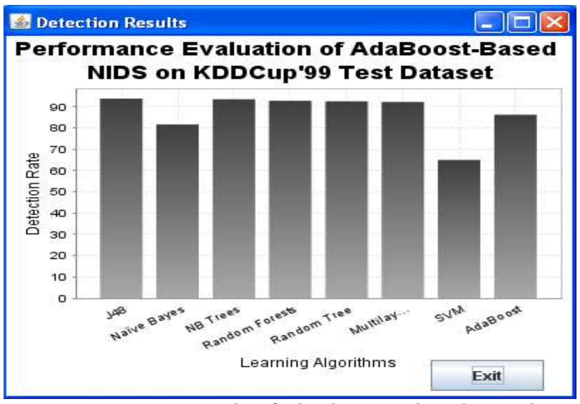

Fig. 3. Detection results of other learning algorithms with AdaBoost-based NIDS on the KDDCup'99 test data.

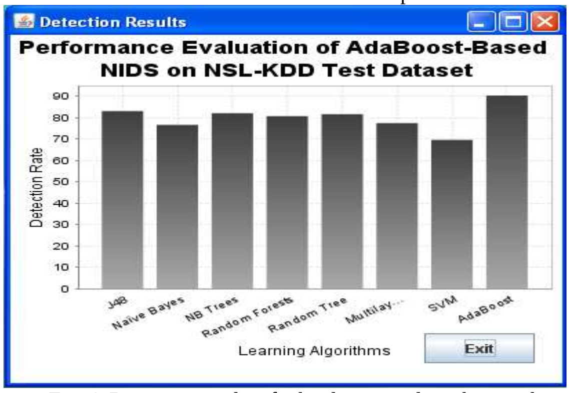

Fig. 4. Detection results of other learning algorithms with AdaBoost-based NIDS on the NSL-KDD test data.

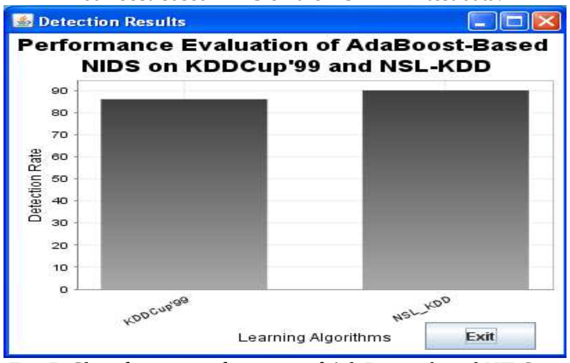

Fig. 5. Classification performance of AdaBoost –based NIDS on KDDCup'99 and NSL-KDD test data.

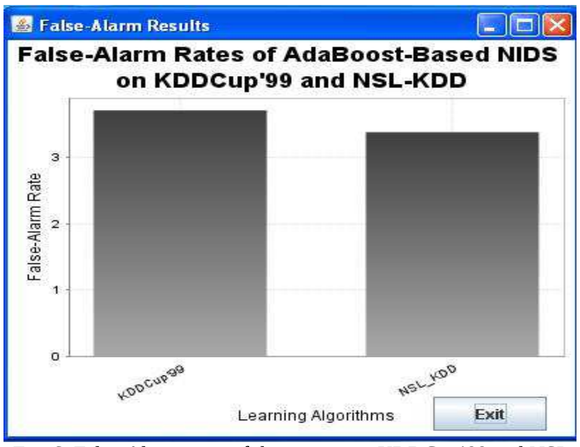

Fig. 6. False-Alarm rates of the system on KDDCup'99 and NSL-KDD datasets.

The above figures show the classification accuracy of various learning algorithms and the AdaBoost-based NIDS on KDDCup'99 and NSL-KDD test dataset. The initial classification results of AdaBoost-based NIDS on KDDCup'99 and NSL-KDD test dataset are 86.27% and 90.00% respectively, which are considerable with other learning algorithms. Figure 7 shows the classification performance of AdaBoost-based NIDS over KDDCup'99 and NSL-KDD test dataset .Due to the redundant records in the KDDCup'99 the system performance is biased and it gives only 86.27% of detection rate, whereas the performance of the system is increased to 90.00% detection rate on the NSL-KDD dataset. Hence NSL-KDD is the good benchmark dataset to test the intrusion detection systems which gives the real performance of the systems without biasing.

6. JUSTIFICATION DIFFERENCE

1. The performance of AdaBoost-based NIDS is improved over the NSL-KDD dataset. The detection rate of the system over the KDDCup'99 dataset is 86.27%, which is improved to 90.00% over the NSL-KDD dataset, which proves that NSL-KDD dataset is a good benchmark dataset to test the intrusion detection systems.
2. The false-alarm rate of the system is also improved over the NSL-KDD dataset. The false-alarm rate over the KDDCup'99 is 3.71 %, which is decreased to 3.38% over the NSL-KDD dataset as shown in figure 6.
3. The number of records in the training and testing sets of NSL-KDD dataset are reasonable i.e. 125973 records in the training set and 22544 records in the testing set, which makes it affordable to run the system on the complete set without the need to randomly select a small portion.

7. CONCLUSION

This paper deals with the initial design of the Network Intrusion Detection System based on AdaBoost-based machine learning algorithm. We have developed this system using Java and the KDDCup'99 and NSL-KDD intrusion dataset. Our initial experimental results are considerable as compare with other learning algorithms evaluated on the KDDCup'99 and NSL-KDD test dataset. The system shows the better results of detection rate and false-alarm rate on the NSL-KDD dataset as compare to the KDDCup'99 dataset.

REFERENCES

[1] S Chebrolu, A. Abraham and J.P. Thomos,"Feature 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]KDDCup 1999 Data, http://www.kdd.ics.uci.edu/databases/kddcup9 9 /kddup99.html, 1999.

[5] M. Tavallaee, E. Bagheri, W. Lu, and A. Ghorbani, “A Detailed Analysis of the KDD CUP 99 Data Set”, Second IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA), 2009.

[6] “Nsl-kdd data set for network-based intrusion detection systems.” Available on: http://nsl.cs.unb.ca/NSL-KDD/, March 2009.

[7] Elkan, Charles, “Results of the KDD’99 classifier learning”, SIGKDD Explorating, 2000.

[8] L. G. Valiant, “A theory of the learnable”, Communications of the ACM, 27(11):1134–1142, November 1984.

[9] 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.

[10] 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.

[11] Dorothy E.Denning,”An, “An Intrusion-Detection Model”, IEEE Transactions on Software Engineering, Volume SE-13, No. 2, February 1987, pp. 222-232.

[12] G. Vigna and R. A. Kemmerer, “NetSTAT: A network-based intrusion detection approach”, in Proceedings of Computer Security Applications Conference, December. 1998, pp. 25– 34.

[13] W. Lee, S. J. Stolfo and K. Mok, “A data mining Framework for building intrusion detection Models”, in Proceedings of IEEE Symposium on Security and Privacy, May 1999, pp. 120–132.

[14] K. Sequeira and M. Zaki, “Admit: Anomaly-Based Data Mining for Intrusions”, in Proceedings of the eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM Press, 2002, pp. 386.395.