Computer Communications 36 (2013) 303–319

Contents lists available at SciVerse ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Detection of distributed denial of service attacks using an ensemble of adaptive and hybrid neuro-fuzzy systems

P. Arun Raj Kumar *, S. Selvakumar

CDBR-SSE Project Laboratory, Department of Computer Science and Engineering, National Institute of Technology, Tiruchirappalli 620015, Tamil Nadu, India

ARTICLE INFO

Article history:
Received 4 July 2011
Received in revised form 12 July 2012
Accepted 12 September 2012
Available online 23 September 2012

Keywords:
DDoS
Neuro-fuzzy
Neural networks
Machine learning
Ensemble of classifiers

ABSTRACT

A DDoS attack is the most prevalent threat, viz., flooding the computing and communication resources in order to make the service unavailable for legitimate users, since a decade and continues to be threatening till date. Therefore, these critical resources must be protected against the DDoS attacks. The detection of DDoS attacks requires adaptive and incremental learning classifier, less computational complexity, and accurate decision making from uncertain information. Hence, the DDoS attacks could be detected using existing soft computing techniques such as fuzzy logic, neural networks, and genetic algorithms. Fuzzy logic has the advantage of interpreting the rules well but it suffers from the disadvantage of not able to acquire the rules automatically. The neural networks generalize the network well but they cannot interpret the rules. Genetic algorithm provides optimal solutions but the time complexity is high. Hybrid methods, Neuro-fuzzy and genetic fuzzy have been proposed to overcome the drawbacks of interpretability and manual rules acquisition. *In* this paper, adaptive and hybrid neuro-fuzzy systems were proposed as subsystems of the ensemble. Sugeno type Adaptive Neuro-Fuzzy Inference System (ANFIS) has been chosen as a base classifier for our research as Mamdani type ANFIS is not suitable for real time due to its high computational complexity and non-adaptiveness to extract exact knowledge from the dataset. Single classifier makes error on different training samples. So, by creating an ensemble of classifiers and combining their outputs, the total error can be reduced and the detection accuracy can be increased. Improvement in the performance of ANFIS ensemble is the focus of this paper. Our proposed DDoS classification algorithm, NFBoost, differs from the existing methods in weight update distribution strategy, error cost minimization, and ensemble output combination method, but resembles similar in classifier weight assignment and error computation. Our proposed NFBoost algorithm is achieved by combining ensemble of classifier outputs and Neyman Pearson cost minimization strategy, for final classification decision. Publicly available datasets such as KDD Cup, CAIDA DDOS Attack 2007, CONFICKER worm, UNINA traffic traces, and UCI Datasets were used for the simulation experiments. NFBoost was trained and tested with the publicly available datasets and our own SSE Lab[1] SSENET 2011 datasets. Detection accuracy and Cost per sample were the two metrics used to analyze the performance of the NFBoost classification algorithm and were compared with bagging, boosting, and AdaBoost algorithms. *From* the simulation results, it is evident that NFBoost algorithm achieves high detection accuracy (99.2%) with fewer false alarms. Cost per instance is also very less for the NFBoost algorithm compared to the existing algorithms. NFBoost algorithm outperforms the existing ensemble algorithms with a maximum gain of 8.4% and a minimum gain of 1.1%.

1. Introduction

Critical services in each infrastructure (wired, wireless, mobile, and sensor networks) are mostly accessed through web interface. These services are well protected by firewall and antivirus software as a first level of defense. With the rapid growth of mobile telecommunication technology, telecommuters are increasing nowadays for procuring the services such as online trading, online shopping, online gaming, online chatting, etc. Due to the advancement in internetworking technology, internet resources and services can be availed remotely in a distributed environment. Though it is beneficial for legal clients to access these services, it is also lucrative for the attackers to earn thousands of dollars weekly by shutting down the victim servers through compromised hosts.

In 1985, the vulnerability in TCP/IP protocol stack that the origin of the packet cannot be traced [46] was found. After 10 years, attackers routinely exploited this weakness to target the victims.

* Corresponding author. Tel.: +91 431 2503239; fax: +91 431 2500133.
E-mail addresses: park@nitt.edu (P. Arun Raj Kumar), ssk@nitt.edu (S. Selvakumar).

[1] All hosts in SSE Lab are connected through a backbone bandwidth of 1 Gbps and to different sites through a 2 Mbps MPLS VPN cloud.

0140-3664/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.comcom.2012.09.010

SYN flood attack at the attack rate of 200 packets/s was one among the several attacks in 1996. But, these attacks were not considered as high profile security alerts. In February 7, 2000, a massive flooding attack [20] happened at yahoo.com website besides the other commercial organizations such as CNN, e-Bay, Amazon, etc. The attack was high rate flooding attacks and was called as Distributed Denial of Service (DDoS) attack. DDoS attack is a coordinated attack on the availability of services of a single or multiple victim systems through many compromised secondary victims. Even after a decade, since its first massive attack in 2000, Internet is still vulnerable to massive DDoS attacks.

In 2008, the largest recorded DDoS attacks against a single target reached 40 Gbps, as against 24 Gbps reported in the year 2007. A sixth annual worldwide infrastructure security report by Arbor networks [8] on 2010 witnessed a sharp escalation in the scale and frequency of DDoS attacks (almost 100 Gbps) on the Internet. From the Arbor Networks report, it is evident that misconfigured DNS open recursors are the easiest way for the attackers to launch a DDoS attack on DNS servers. A semi-annual Web Hacking Incident Database (WHID) report by Trust wave [64] on December 2010 revealed that the DDoS attacks are on the rise followed by SQL injections and cross-site scripting (XSS) attacks.

Attackers utilized brute-force attack techniques to commit an inadvertent DDoS attack on the VoIP infrastructure resulting in service outages [64]. Botnet-driven and application-layer based DDoS attacks are on the rise in 2011. Hence, DDoS attacks are the most significant security threat that Internet Service Providers (ISPs) face. Revenue loss, network performance degradation, and service unavailability at critical time are some of the factors that motivated us to provide protection against DDoS attacks. The extensive growth of the mobile and fixed wireless Internet made DDoS attack detection a critical component of infrastructure protection mechanisms.

Misuse detection methods identify packets that match a known pattern or signature. However, these methods fail to detect unknown anomalies. An anomaly could be an old attack that has changed its pattern in an obtrusive way to avoid detection, or it could be a completely new form of attack. Anomaly detection methods are used to identify the traffic patterns that deviate from the modeled normal traffic behavior. The identified anomalies could be either an attack or normal traffic. However, modeling a normal traffic is not an easy task as new applications in the Internet are evolving continuously. Also, it is tedious to model attack behavior as there is high variability in attack features. Different applications exhibit different degree of burstiness and time correlation. Recently, mechanisms such as stateful firewall and Intrusion Prevention Systems (IPS) are used for the prevention of DDoS attacks. But, these mechanisms are more susceptible to DDoS attack as the state tables in firewall were overwhelmed by moderate size DDoS attack. To avoid stateful inspection method, a simple response packet confirmation mechanism is proposed in [29]. In [29], the focus is on Distributed Reflective Denial of Service (DRDoS) attacks using IP spoofing. Our proposal, NFBoost, considers the number of SYN errors and the number of connections to the same host during the specified time window to include the detection of DRDoS attack also. By deluging with traffic, the DDoS attacks mostly target web servers and deplete the server resources. DDoS attacks such as IRC flood, HTTP flood, SYN flood, UDP flood, and buffer overflow have been posing a serious threat [20] to such resource centers. In this paper, adaptive and hybrid neuro-fuzzy systems were used as subsystems of the boosting ensemble. Our proposed ensemble differs from the existing ensembles [15,28,56] in weight update distribution strategy, error cost minimization, and ensemble output combination method, but resembles similar in assigning classifier weights and error computation. Our proposal is to make automatic and intelligent message discard decisions based on neuro-fuzzy ensemble to result in fewer false alarms allowing more legitimate users to avail the service. The contributions of this paper include the following:

- Implementation of NFBoost algorithm for the classification of network traffic.
- Efficient false positive reduction using Neyman Pearson (NeP) Hypothesis.
- A classification accuracy of over
  - 98.2% when training and testing on the KDD cup dataset.
  - 98.8% when training and testing on the Mixed traffic dataset.
  - 99.2% when training and testing on the SSE Lab SSENET 2011 Dataset.

The rest of the paper is organized as follows: DDoS attack characteristics, DDoS attack datasets, Existing Feature Extraction methods, Soft Computing techniques for intrusion detection, Existing ensemble of neuro-fuzzy classifier methods, and Cost minimization mechanisms are discussed in Sections 2.1 through 2.6 respectively. Preprocessing steps are explained in Section 3.1. Proposed NFBoost algorithm is elucidated in Section 3.2. Comparison of NFBoost with existing ensemble methods are discussed in Section 3.3. Five experiments were conducted and the results are discussed in Sections 4.1 through 4.5 respectively. Section 5 concludes the paper.

2. Related work

2.1. DDoS attack

Simple tools that generated and sent packets from a single source to a single destination were the early DoS attack technologies used a decade ago. Later, sophisticated tools were used to launch single source attacks against multiple targets, multiple source attacks against single targets, and multiple source attacks against multiple targets. Today, the most common DoS attack type reported to the CERT/CC involves sending a large number of packets to a destination causing the consumption of resources (CPU and memory) and network bandwidth. Such attacks are commonly referred to as packet flooding (high rate) attacks. Flooding attacks may be categorized into high rate and low rate flooding attacks. Attacks such as quiet attack [5], low rate TCP Denial of Service (DoS) attack [2,3], and Reduction of Quality (RoQ) attacks [4,68] are classified into short lived TCP flows. Studies [4,68] show that less than 2% of the internet traffic only accounts for short lived flows and one third of the flows of internet traffic accounts for long lived flows. Low rate DoS attacks are detrimental to UDP (VoIP) traffic as well [67]. In this paper, the feature, number of UDP echo packets to a specified port is used for detecting the shrew and the RoQ attacks.

DDoS worm detection is generally done by analyzing the packet payload and its size. However, due to the presence of confidential data in payload, some countries do not recommend inspecting packet payload due to privacy violation. Also, attackers use randomly generated port numbers to conceal their identities. Hence, inspection on packet payload and transport protocol port numbers are not sufficient enough for accurate detection of DDoS attacks. The detailed analysis on DDoS attacks and available attack tools [9] show that the DDoS attack has the following characteristics:

- Source and Destination IP address and port numbers of the packets are spoofed.
- Window size, sequence number, and packet length are fixed during the attack.
- Flags in the TCP and UDP protocols are manipulated.
- Roundtrip time is measured from the server response.

Table 1
Comparison of normal access and DDoS attack.

Operation	Normal Activity	DDoS Attack
Large number of connections from multiple source ports to a single destination port of a server for downloading a file	Download manager E.g. Flashget	SYN flood
A single source port is used to connect to a lot of destination IP addresses for sharing files	P2P file sharing applications	Host scan
More number of http requests to popular web server	Flash crowd (Increase in the number of clients)	Increase in the request rate from small group of clients

- Routing table of a host or gateway is changed.
- DNS transaction IDs (reply packet) are flooded.
- HTTP requests are flooded through port 80.

Similarities between legitimate access and DDoS attacks [10] are shown in Table 1. For instance, Flashget [27] on a host (specific IP address) creates a large number of connections using multiple source ports to a single destination port of a server for downloading a file. This behavior is similar to that of SYN flood. However, the difference between SYN flood and Flashget is that the SYN flood will not complete the TCP 3-way handshakes with the target victim within the specified time period leading to a lot of SYN errors. Further, many P2P file sharing applications use a single source port to connect to a lot of destination IP addresses for sharing files. This looks similar to a host scan activity. Flash crowd refers to more number of clients with less number of requests. But, DDoS attack is generated by less number of clients each generating a huge request traffic rate. Thus, the features such as number of connections from the same host to the specific destination within the specified time window and number of connections having SYN errors within the specified time window have been used to differentiate the flooding attacks from abrupt changes of legitimate activity.

Real challenge lies in distinguishing the flooding attacks from abrupt burst in legitimate traffic. In [69], three distance metrics such as Sibson, Jeffrey, and Hellinger were used to discriminate the flash crowds from DDoS attacks but the Sibson distance emerged as the best metric. Two flows with two different probability distributions were considered. Similarity between flows was computed besides the difference between the flows measured by 3 metrics. Threshold was used in the 'measured distance' metric to decide if it is a DDoS attack or flash crowd. A detection accuracy of 65% only could be achieved in [69]. In [72], proactive tests were conducted to identify and isolate the malicious traffic after successful TCP connection establishment.

2.2. DDoS attack dataset

The most significant challenge for an evaluation of intrusion detection algorithm is the lack of appropriate public DDoS attack datasets. Since 2000, the three publicly available datasets for intrusion detection are the DARPA/Lincoln Labs packet traces, the KDD Cup dataset [33] derived from DARPA traces, and Cooperative Association for Internet Data Analysis (CAIDA) [16,17]. The DARPA dataset contains multiple weeks of network activity from a simulated air force network, generated in 1998 and refined in 1999. The data is not synthetic and does not reflect contemporary attacks. There were criticisms on DARPA data [58]. The KDD Cup dataset was created by processing the tcpdump portions of the 1998 DARPA Intrusion Detection System (IDS) Evaluation dataset. CAIDA dataset [16] consists of DDoS attack dataset 2007, which can be availed by user request. DDoS attack dataset from CAIDA consists of one hour of anonymized traffic traces from a DDoS attack on August 4, 2007.

The reason for the lack of public DDoS attack datasets is that the deep inspection of network traffic reveals highly sensitive information such as confidential communications, user's network access patterns, etc. Any breach of such information can be disastrous for both the organization and service providers. Solution is to create synthetic data through simulation, emulation, or anonymization of shared real time datasets. Evaluation of detection algorithm using a simulated dataset may not produce accurate results and cannot be compared with real time dataset results. Traffic generated in a laboratory network differs from the aggregate traffic seen in upstream routers. Conclusions drawn from analyzing a small environment cannot be generalized to settings of larger scale [58]. Hence, our proposed algorithm is compared with the existing algorithms using our lab generated dataset (emulation) and publicly available datasets.

2.3. Real time feature extraction

Features are statistical characteristics derived from the collected dataset. Statistical features are used to identify and classify the network traffic. It can be performed offline or online. Online identification of statistical features has to be performed early by observing first few packets of the flow. Feature calculations in real time incur high computational costs. So, selection of features from the collected network traffic plays a vital role in DDoS attack pattern classification. KDD cup dataset contain 41 features. In [45], 248 features were given and 1 feature was used to describe the class (normal or attack). Computation of all the 248 features [12] took approximately two days on a dedicated system area network. More number of features led to better accuracy. But, computation of more number of features in real time causes more overhead and time consuming. So, less number of features is suitable for better pattern classification in real time. Feature extraction methods based on per flow analysis are expensive, not scalable, and thus prohibitive for large scale networks. A single packet does not offer much information. But, by processing the continuous flow of the packets, additional characteristics of network activity can be obtained between hosts. Six features as shown in Table 2 have been selected in this paper for accurate detection of DDoS attacks.

Our objective is to differentiate the DDoS attack and normal traffic. DDoS attacks are time based and connection based. Time based attacks are referred to as bursty attacks and connection based attacks are referred to as pulsing zombie attacks [37]. DDoS attacks such as TCP SYN flood, HTTP flood, UDP flood, P2P Botnet traffic, etc., are high rate flooding attacks. These attacks mostly consist of spoofed source address and contain half-open connections. The 'number of SYN errors' feature has been used to identify the incomplete TCP 3 way handshake connections. Most of the attacks target the victim servers through legitimate ports such as 80, 53, 443, etc. Hence, the 'number of requests' feature from clients over a time window was used to monitor the legitimate ports.

Feature extraction [31] is classified into two stages: (i) Feature construction (ii) Feature selection. Constructing the features is either integrated into the modeling process or into the preprocessing stage which includes standardization, normalization, etc. Feature selection is divided into filter methods and wrapper methods [48]. In filter methods, selection is based on distance

Table 2
List of features.

S. No.	Feature Description
1	Number of UDP echo packets to a specified port.
2	Number of connections to the same host during specified time window.
3	Number of ICMP echo reply packets from the same source
4	Number of connections having SYN errors using the same service during specified time window
5	Variance of time difference between two consecutive packets
6	Ratio of incoming SMTP packets and outgoing SMTP packets

and information measures in the feature space. In wrapper methods, selection is based on classifier accuracy. In this paper, wrapper method has been used for the feature selection.

2.4. Soft computing methods

Soft computing techniques such as neural networks, genetic algorithm, fuzzy logic, and hybrid approaches are the intrusion detection methods in recent research. Neural network consists of processing elements called neurons. These neural networks are designed to learn a new pattern, new association, and new functional dependencies. Advantage of neural network is better generalization capability. Several Machine Learning (ML) algorithms have been proposed [25,30,53,59,66] for DDoS attack detection. Disadvantages of neural network include greater computational burden and overfitting. A Genetic algorithm is a search heuristic that mimics the process of natural evolution. Genetic algorithms belong to the larger class of evolutionary algorithms (EA), which generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover. In [36], a genetic clustering algorithm for intrusion detection was proposed and from the results, the optimal number of clusters and high performance rates were obtained. Drawback of genetic algorithm is computational intensity. It searches for a solution from a broad spectrum of possible solutions, rather than where the results would normally be expected. Fuzzy logic allows decision making with estimated values under incomplete or uncertain information, especially for the system that is tedious to derive the mathematical model. In [60], a method of incremental mining is proposed so that fuzzy association rules can be implemented in a real-time network IDS. Creation of fuzzy sets from the input network packet data, membership functions for fuzzy variables, and the application of genetic algorithm to identify the best rules are proposed in [24]. In [35], the Hidden Markov Model (HMM) is enhanced to fuzzy HMM (FHMM) where fuzzy similarity measures replaced probabilistic measures.

The main drawback of soft computing methods is the lack of interpretability [43]. Hybrid methods have been proposed to overcome the drawbacks of each of these methods. A neural network can approximate a function, but it is impossible to interpret the result in terms of natural language. Fuzzy logic systems, which can reason with imprecise information, are good at explaining their decisions but the rules used by these systems to make decisions are not acquired automatically. These limitations have been our motivation for the creation of intelligent hybrid systems where these neural network and fuzzy logic systems are combined to overcome the limitations of individual techniques. Hence, Neuro-fuzzy method has been used in this paper.

2.5. Ensemble of neuro-fuzzy classifiers–motivation

Neuro-fuzzy systems have advantages over neural networks in learning all its membership function parameters by the hybrid backpropagation algorithm. To improve their accuracy, ensemble of neuro-fuzzy classifiers can be created using the existing ensemble algorithms such as bagging [15], boosting [56], and AdaBoost [28] algorithm. Accuracy obtained from an ensemble of classifiers will be higher than that obtained from a single classifier. Single classifier makes error on different training samples. So, by creating an ensemble of classifiers and combining their outputs, the total error can be reduced and the detection accuracy can be increased. There are two main components in all ensemble systems [51], viz., a strategy to build an ensemble that is as diverse as possible and the combination of outputs of classifier for the accurate classification decisions.

Each classifier decision boundary has to be uniquely different from others. To achieve diversity, ensemble of classifiers can be constructed by manipulating training data, feature sets, and injecting randomness. For the construction of the ensemble by manipulating training data, the entire dataset is divided into subsets and each classifier is trained with each subset. In order to construct the ensemble by manipulating input feature sets, it is divided into smaller feature subsets and each classifier is trained with the different feature subset of the same dataset. Another method to construct the ensemble is by randomly initializing the parameters such as weights, etc., and training with different parameter values at different times. As the number of features selected for training in this paper is six which is less, the ensemble construction by feature set is not suitable [48]. The advantage of constructing an ensemble by manipulating training data is that the generated hypothesis performs fairly well even when there are only small changes in traffic data. So, ensemble construction by manipulating training data was chosen, as it would correctly detect the deviations, if any, however small it be. The ensemble classifier approach [26] completed in between 3 to 4 hours, but the monolithic model took about 3 weeks or 504 hours to generate the 22 un-ordered rulesets for each order. In other words, the monolithic classifier approach is about 150 times computationally more expensive than the ensemble approach.

Existing ensemble of neuro-fuzzy methods for intrusion detection are shown in Table 3. From Table 3, it is evident that detection rate is less and false alarm threshold is high for several methods. More number of features (41 features) has been used in [64] for intrusion detection which consumes more time and increase in cost. None of the methods in Table 3 have considered cost reduction in misclassification. All the existing methods in Table 3 have similar kind of detection design (neuro-fuzzy model), but differ in boosting performance methods such as bagging, boosting, and AdaBoost. In this paper, NFBoost is proposed to boost the performance of Neuro-fuzzy classifier.

Classifier combination is divided into two categories:

- Classifier selection, where each classifier is trained to become an expert in some local area of the total feature space.
- Classifier fusion, where all classifiers are trained over the same feature space.

Classifier outputs can be combined by methods such as Weighted Mean (WM) [32], Majority Voting [51], Weighted Majority Voting (WMV) [51], Gating, stacking, etc. In this paper, popular ensemble methods such as bagging [15], boosting [56], and AdaBoost [28] are compared with our proposal, NFBoost. The fusion methods such as majority vote and the weighted majority vote require low computational resources, whereas other fusion methods such as gating, stacking, etc., require high computational resources. These computational requirements are unacceptable for the problem having large number of samples. In gating network, the weight is computed as a function of both the input and the hypothesis [42]. In stacking or combiner [22,49], a separate classifier is learnt

Table 3
Comparison of existing ensemble of neuro-fuzzy classification algorithms.

S. No.	Methods used	Application	Advantages	Drawbacks
1.	Ensemble of ANFIS and genetic algorithm to optimise the structure of fuzzy system [63]	Intrusion Detection (KDD cup 99 dataset)	Detection rate is high	• False positives are more • Obsolete dataset • More number of features
2.	ANFIS and bagging [18]	20 datasets, but no intrusion detection datasets	Bagging with mean defuzzified,high detection accuracy, and feature reduction	Detection rate is not high
3.	Ensembles of ANFIS [13]	Mental activity database	Hierarchical classification	Detection rate is not high
4.	Neuro-fuzzy and bagging [40]	Glass type identification	Detection rate is high	Number of classifiers are more
5.	ANFIS and adaBoost [34]	PIMA diabetes and breast Cancer	Merging rule sets from several neuro-fuzzy systems to form a single rule base	No reduction in misclassification cost
6.	Ensemble of neuro-fuzzy classifier and multi label voting algorithm [21]	Visual arts data mining, IRIS, Glass dataset	Performance is better compared to existing algorithms	Detection accuracy is less
7.	Ensemble of neuro-fuzzy classifier, adaboost, and Rough sets [39]	PIMA Indian Diabetes problem	Classification in case of missing or unknown features and less number of features	• Detection accuracy is less • Misclassifications are more
8.	From ensemble of fuzzy classifiers to single fuzzy rule base classifier (Mamdani NFS) [38]	Breast cancer and glass type identification	Detection accuracy is high	• Computationally not efficient • Not suitable for real time systems
9.	Ensemble of relational neuro-fuzzy + adaBoost [52]	PIMA Indian Diabetes problem and Monk 2	Detection accuracy is 83.2% and 100%	Low classification accuracy for binary classifications
10.	Fusion methods using ensemble ofRBFNN for network intrusion detection [1]	KDD CUP Dataset	Detection accuracy is high and less false alarm. Computation faster.	• Complexity is high in handling large dataset • Separate classifier for each attacks • Predicts only known intrusions not new attacks

to combine the predictions of individual classifiers. Stacking or combiner learns a meta-level hypothesis on the correlations of predictions of individual hypotheses to the true label.

By combining intelligent learning systems, the ensemble model accuracy is always improved, comparing to single-model solutions. Our algorithm differs from existing algorithms in two ways, viz., achieving diversity of the classifiers and combining the classifier outputs through Weighted Mean (WM) [51] and Majority Voting (MV) [51].

2.6. Cost minimization

From Table 3, it is evident that none of these methods considered reducing the cost of incorrect classifications. Some of the methods achieve high detection accuracy, but false positives are more. Using ensemble of neuro-fuzzy classifiers, only few methods use Intrusion detection datasets for classification as shown in Table 3. Several other methods tested their proposed ensemble methods with different applications and different feature sets.

Our aim is to achieve high detection accuracy with less false positives for binary class problems. In [1], high detection accuracy of 100% for multi-class problems was achieved, but it failed to achieve the same for binary classifications. Hence, it is evident that methods used for multi-class problems cannot be applicable for binary classification problems.

Our proposed NFBoost algorithm, inspired in part by AdaBoost, differs in weight update distribution and post-training step process. In this paper, error cost reduction occurs during training and also during validation (post-training step). During training, the incorrectly classified instances are checked for false positives. If these instances are misclassified and detected as false positive (normal traffic is misclassified as attack), more weightage is given to these instances during next iteration. As a post-training step, by inputting validation data to the trained ensemble, the optimum threshold of false positives for each classifier in the ensemble is determined using Neyman Pearson (NeP) Hypothesis. The relative cost of any misclassification is extremely high compared to many other machine learning applications. A false positive requires spending expensive analyst time examining the reported incident only to eventually determine that it reflects benign underlying activity. In this paper, Neyman Pearson hypothesis is used for cost-minimization strategy.

In [23], a classification system using ensemble of neural network classifiers for each class has been considered and used the classification error cost minimization technique in the intrusion-detection problems for multi-class problems using Bayesian probability analysis. The differences between dCMS [23] and our Neyman Pearson cost minimization method are shown in Table 4.

2.7. Existing traceback mechanisms

Detection techniques detect the attack traffic. The exact origin of the attack must be known to prevent the further attacks. Hence, the challenge in DDoS attack detection is in identifying the attack source. By tracing back to the source IP address, reflectors or compromised hosts may be found. But, the real attacker is not found. The security expert spends time in tracing the real source address and at times it demands more time as the perpetrator spoofs the source IP address. Traceback mechanisms [6,7,14,44,55,57,70,71,73] have been proposed to trace the real source of the attackers in order to take actions against them and to circumvent the attack at the point nearest to its source in order to prevent the further attacks.

In Probabilistic Packet Marking PPM [55], routers mark the packets probabilistically with either the router IP address or the edges of the path that the packet traverses to the router. Due to more number of combinations required to rebuild a fragmented edge ID, the reconstruction of such an attack graph is computationally intensive. Furthermore, the approach results in a large number of false positives. In order to gain the correct attack path with 95%

Table 4
Comparison of dCMS and NeP.

dCMS	NeP –Cost Minimization
Prior knowledge of class probabilities	No prior knowledge of class probabilities
Ensemble construction by feature Set	Ensemble construction by training data
Multi-class problems	Binary classification problems
Bayesian probability analysis for determining optimum threshold	Neyman Pearson hypothesis for determining optimum threshold

accuracy, 294,000 packets are required. But, in Deterministic Packet Marking (DPM) [7], zero false positive with 99% accuracy is obtained with only 8 packets. In PPM, full path is required. But in DPM, address of the ingress router is enough for the construction of attack graph. Flexible-DPM [67] adopts a flexible mark length strategy to make it compatible to different network environments. It also adaptively changes its marking rate according to the load of the participating router by a flexible flow-based marking scheme. FDPM has low false positive rates.

Recently, a new hybrid IP traceback scheme, RIHT [44], with efficient packet logging and a fixed storage requirement for each router in packet logging without the need to refresh the logged tracking information, has been proposed. Zero false positive and false negative rates have been achieved in attack path reconstruction. Modified-DPM is used for the DDoS attack traceback in IPv6 networks. In IPv6, destination options header fields are used for storing the ingress router address. A detailed comparison on some of the existing traceback mechanisms with respect to their working principle, advantages, and drawbacks is given in our published work [9].

3. Proposed NFBoost based ensemble system

The proposed NFBoost based Ensemble system consists of the following two stages:

- Preprocessing.
- Classification.

3.1. Preprocessing

The input to the preprocessing is the network traffic and the output of this stage is normalized dataset. The block schematic of preprocessing is shown in Fig. 1. The preprocessing stage consists of feature extraction and normalization. Preprocessing refers to the process of extracting information about packets from network traffic for the construction of new statistical features. The preprocessing steps are explained as follows:

- Let 'x' be the input vector of dimension 'n', such that $x = [x_1, x_2, x_3, \ldots x_n]$. The variables x_i of the input vector are the original features.
- Let 't_x' be a vector of transformed features of dimension 't_n'.

A receiver process running in promiscuous mode captures all incoming packets and stores in data storage server. The captured network traffic is given as input to the tcptrace [62] tool for feature extraction. Tcptrace tool extracts the features from the captured traffic. The data is stored as a set of traffic flows, with each instance being qualified by a set of features. Each instance is expressed in vector space model.

The statistical properties such as mean, standard deviation, and variance are computed for the extracted six features. These features quantify the behavioral characteristics of a connection in terms of number, type of various data items with respect to time. Hence, theses features are called as statistical real time features. The extracted features and its values are input to the normalization module. Normalization is a process of ensuring that each attribute value in a database is suitable for further querying and free from certain undesirable characteristics. Hence, each variable is normalized in the range [0,1] to eliminate the effect of scale difference. As the feature consists of both continuous and discrete values, it is normalized using min-max normalization such that the training dataset contain values between [0,1]. These values are used as inputs for machine learning algorithms. The attributes are scaled to the range [0,1] using (1), where '$i(t)$' denotes the value of the feature, 'min(i)' denotes the minimum value, and 'max(i)' denotes the maximum value. Thus, data available for the classifier are real numbers between 0 and 1.

$$i_{\mathrm{norm}(t)} = \frac{i(t) - \min(i)}{\max(i) - \min(i)} \tag{1}$$

3.2. Proposed NFBoost classifier

Block schematic of the proposed NFBoost classifier is shown in Fig. 2. The neuro-fuzzy experts, NF_1 to NF_T in Fig. 2, act as classifiers. The defuzzification strategy for each expert identifies the most significant fuzzy set of the output as the predicted class by Mean-Of-Maximum (MoM). In the MoM defuzzification method, the scaled membership function is identified with the greatest degree of membership and the value for that membership function is determined.

The determined value is the mean of the numerical values corresponding to the degree of membership at which the membership function was scaled. Neuro-fuzzy has been combined into ensembles to improve accuracy and robustness. Ensemble of classifiers is created when trained with normal and attack data subsets. The output of ensemble of classifier is combined by Weighted Mean algorithm. The final classification decision is chosen by Majority Voting, where each ensemble vote is included. The proposed NFBoost classification algorithm consists of the following two steps as shown in Fig. 3:

- Training.
- Testing.

3.2.1. Training

Dataset consisting of both attack and normal traffic is used for training and testing. Number of samples in each class is selected randomly. The inputs to the algorithm are as follows:

- Training dataset
- Adaptive Neuro-Fuzzy Inference System (ANFIS) as supervised base classifier
- Number of classifiers and Number of classes

Initially, the instance weights in the dataset are assigned uniformly. The distribution of the samples as input to the supervised

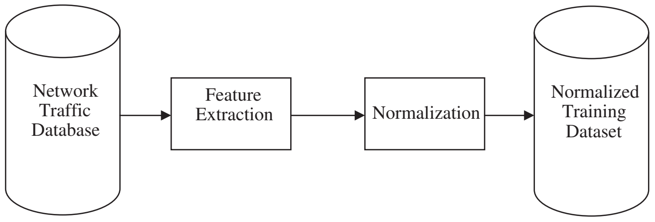


Fig. 1. Preprocessing Steps.

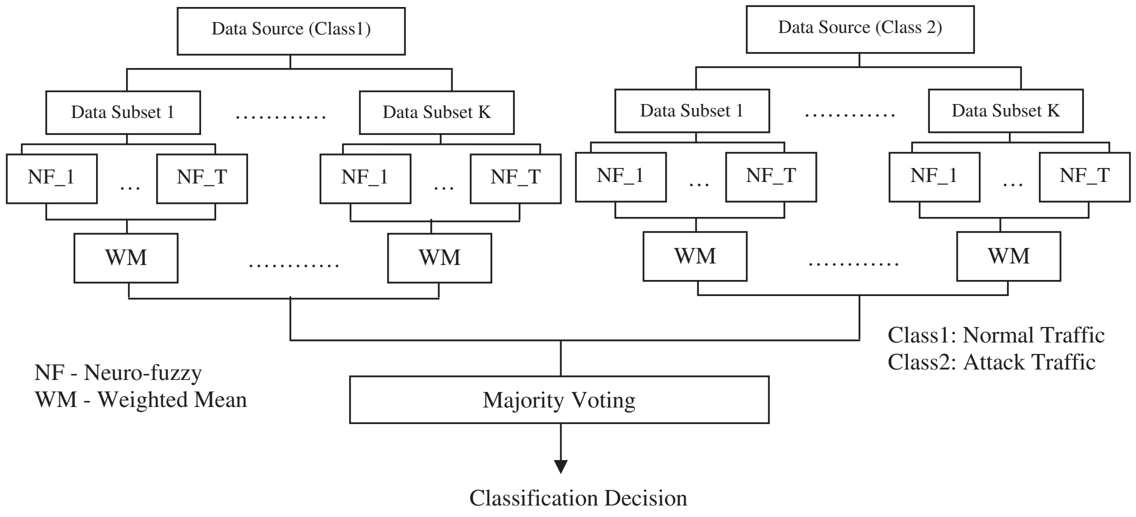


Fig. 2. Block schematic diagram of NFBoost classifier.

base classifier is given in (2), where ‘n’ is the total number of instances in the data subset and $d_t(i)$ is the distribution of the classifier ‘t’. The weights of all instances in the data subset are summed up (resulting in unity during initialization). Hence, instances are selected equally likely from the data subset. Initially, false alarm threshold is set to 0.5. So, the classifiers failing to achieve 50% accuracy are dropped.

$$d_t(i) = \frac{1}{n} \tag{2}$$

Samples from class1 are chosen from a dataset$_1$. The dataset$_1$ is split into ‘k’ subsets as shown in Fig. 2. Each subset is given as input to the base classifier, ANFIS [54], NF_1 to NF_T. Sugeno type fuzzy model is used as it has advantages such as computationally efficient (suitable for real time) and adaptive to extract exact knowledge from the dataset than Mamdani fuzzy model. ANFIS is a method for tuning an existing rule base with a learning algorithm based on a collection of training data. ANFIS is trained with subset1 to obtain hypothesis $h_{t.}$ The structure of the neuro-fuzzy system is shown in Fig. 4. The fuzzy subsystems are learnt by gradient learning and initialized by Fuzzy C-Means (FCM) clustering algorithm. In Fig. 4, number of nodes in Input layer (N) is the product of number of inputs and membership functions (MFs) for each input.

Membership function is defined in the range of 0 to 1 which maps each point in the dataset to any one of the subranges classified as low, medium, and high. Membership function chosen is Triangular membership function. Number of nodes in rules, Normalization, and Defuzzification layer is equal to the generated rules in the fuzzy base. To model the target system, a training dataset of input/output pair ($x1$,$x2$,$x3$,$x4$, $x5$,$x6$,y) is required. Hence, the input/output are mapped adaptively by ANFIS through MFs, rules, precise, and consequent parameters. Hybrid learning method (back propagation algorithm + least mean square algorithm) is used to update the Fuzzy Inference System (FIS) parameters. The training process continues till the desired least mean square is achieved. In Rules layer as shown in Fig. 4, T-norm operator is used. Firing strength of each rule is calculated based on the membership functions.

In normalization layer, firing strength of a rule (r_1) is normalized to the sum of all rule’s firing strength using (3). Similarly, the firing strength of the other rules is normalized. In Fig. 4, every node in rules layer is labeled as N to indicate the normalization of the firing levels.

$$r_{11} = \frac{r_1}{r_1 + r_2 + \cdots + r_n} \tag{3}$$

Defuzzification is the process of converting the degrees of membership of output linguistic variables into numerical (crisp) values. In defuzzification layer, the output of the neuron is the product of the normalized firing level and the individual rule output of the corresponding rule. A single node in output layer computes the overall system output as the sum of all incoming signals using (4), where y_i is the individual rule output.

$$h_t = \sum_{i=1}^{n} r_{ii} y_i \tag{4}$$

Thus, after the defuzzification process, the hypothesis h_t is generated. The classifier error is computed using (5).

$$\varepsilon_t = \frac{\sum_{i=1}^{n}[h_t(x_i) \neq y_i]}{n} \tag{5}$$

If the error is not less than the false alarm threshold, the hypothesis is dropped. Otherwise, the normalized error is computed using (6) and the weight is assigned to the classifier using (7).

$$\beta_t = \frac{\varepsilon_t}{1 - \varepsilon_t}, \quad 0 < \beta_t < 1 \tag{6}$$

$$w_t = \log \frac{1}{\beta_t} \tag{7}$$

After the assignment of the weight to classifier, the distribution of the samples is updated using distribution update rule as given in (8).

$$d_t + 1(i) = d_t(i) \times \left\{ \begin{array}{ll} \beta_{tt}, & \text{if } h_t(x_i) = y_i \\ 1, & \text{if } h_t(x_i) \neq y_i \text{ and if } y_i \in \Omega_1 \\ \mu, & \text{if } h_t(x_i) \neq y_i \end{array} \right\} \tag{8}$$

The training distribution is updated by including the cost of misclassification on successive iterations. The updated distribution samples are given as input for the next iteration. Similarly, the process is repeated for T iterations. Number of classifiers is selected using cross-validation method. Cross-validation is a popular method of manipulating training data to subdivide the training data into ‘k’ disjoint subsets and to reconstruct training sets by leaving out some of the subsets.

```
Input:
    For each dataset DS_j, where j = 1 and 2, representing Normal and Attack
        • Training Data 'TD_j' of size 'N' with correct labels
        • Adaptive Neuro-fuzzy Inference System as Supervised algorithm base classifier
        • Number of iterations or classifiers (T)
        • Number of Classes (L) = {Ω_1, Ω_2}

Initialize:
        • μ = 0.5              // False Alarm Threshold
        • L = 2                // Number of Classes (Normal, Attack)
        • $d_t(i) = \frac{1}{n}$     // Summing up all instance weights result in unity
Training:
    Do for j = 1... L
        1. Choose samples from class 'j' and form Data Source DS_j
        2. Split DS_j into 'k' subsets (S_1, S_2,…..………, S_k)
            Do for each m = 1…..k              // Number of Data Subsets
                Do for t = 1……T                // Number of Classifiers
                    a. Train S_m by ANFIS and obtain hypothesis h_t
                    b. Compute error of h_t:
                       $\varepsilon_t = \frac{\sum_{i=1}^{n}[h_t(x_i) \neq y_i]}{n}$
                    c. If $\varepsilon_t$ > μ, then drop hypothesis and go to step 2.a
                       Else, add the classifier h_t to the Ensemble 'E_m'.
                    d. Compute normalized error:
                       $\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}\ \ 0 < \beta_t < 1$
                    e. Assign weight to the Classifier:
                       $w_t = \log\frac{1}{\beta_t}$
                    f. β_tt = β_t * μ
                    g. Update Distribution of Instances:
                       $d_{t+1}(i) = d_t(i) \times \begin{Bmatrix} \beta_{tt}, & \text{if } h_t(x_i) = y_i \\ 1, & \text{if } h_t(x_i) \neq y_i \text{ and if } y_i\ \varepsilon\ \Omega_1 \\ \mu, & \text{if } h_t(x_i) \neq y_i \end{Bmatrix}$
                End
                    h. Obtain Composite Hypothesis:
                       $H_j^{\,m} = \arg\ mean \sum_{t=1}^{n}\left(\log\frac{1}{\beta_t^{\,m}}\right)$
            End
    End

Testing:
    Given an unlabeled instance 'X'
        A. Evaluate the ensemble 'E_m' of each data subset for particular class on 'X'.
        B. Obtain composite hypothesis for each subset by Weighted Mean.
        C. Find the class based on the threshold value.
        D. Obtain the total vote received by each class.
        E. Choose the class that has the highest total vote as the final classification decision.
```

Fig. 3. Proposed NFBoost classification algorithm.

The instances can be classified correctly or incorrectly. If the instances are classified correctly, the weight of these instances is reduced aggressively such that these instances will not be considered during next iteration of training. If the instances are incorrectly classified, there are two possibilities: (i) normal traffic classified as attack (false positive) and (ii) attack classified as normal traffic (false negative). A closer look at the distribution update rule in (8) reveals that the assignment of weight to false positive instances increases than false negative instances and correct classifications. Such an increase in weight for false positive instances has been thoughtfully designed and introduced by us in our proposal in this paper with an objective of reducing false positives and increasing detection accuracy. Therefore, NFBoost raises the weights of instance misclassified by h_t for successive iterations. Consequently, if the weight exceeds 0.5, it drops the hypothesis, indicating the classifier is bad. Also, the number of classifiers is reduced as more weight has been assigned to false positive instances during training.

Each hypothesis in the ensemble is trained on the same dataset with different distribution. All hypothesis generated thus far are then combined using Weighted Mean algorithm. The working of Weighted Mean algorithm can be illustrated with an example as follows: Let us assume five classifiers as $(c1, c2, c3, c4, c5)$. Weights are assigned to the classifiers during training using (8). Let the weights of five classifiers be $(0.5, 0.7, 0.6, 0.5, 0.7)$. For a given instance 'X', after defuzzification, let the output of the five classifiers be $(0.4, 1, 0.5, 0.6, 0.7)$. Let the optimum threshold be assigned as 0.5. If the predicted output is less than 0.5, the instance belongs to class 1. If the predicted output is more than or equal to 0.5, the instance belongs to class 2. Using weighted mean approach, weights and outputs of each classifier are multiplied and mean is calculated. The mean value thus obtained is 0.39, which is less than 0.5. So, the test instance 'X' belongs to class 1.

The composite hypothesis is obtained where each hypothesis is assigned a weight inversely proportional to its normalized error. Therefore, hypothesis with less error is assigned a higher voting

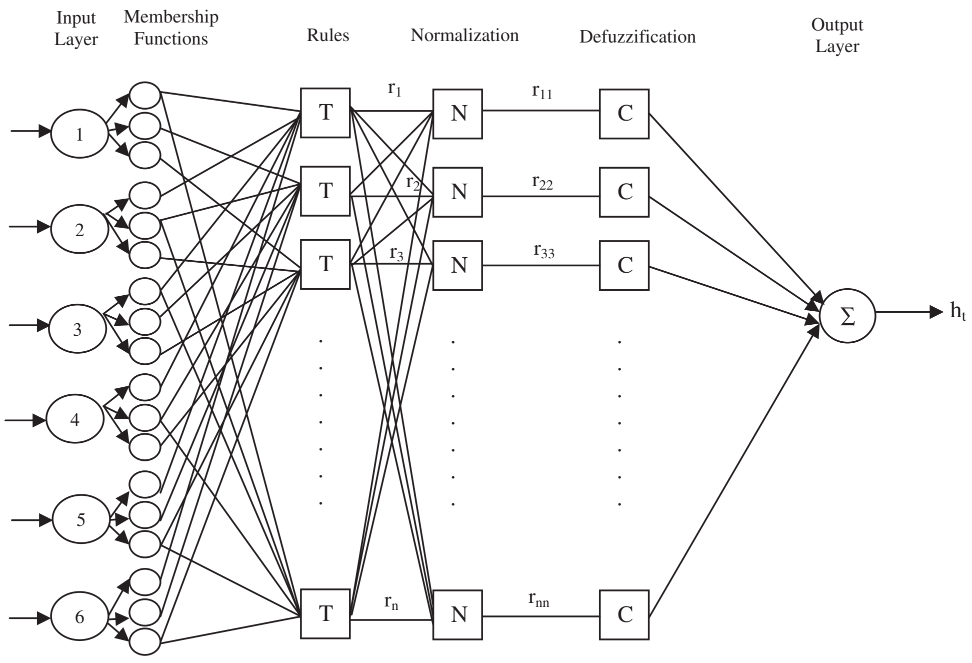


Fig. 4. Neuro-fuzzy System.

weight. Each classifier error is less than 0.5. Hence, the error of composite hypothesis will also be less than 0.5. The output of the classifiers trained by ANFIS algorithm is computed using (9).

$$f(x) = \sum_{i=1}^{r} W_t h_t(x) \tag{9}$$

3.2.1.1. Post training - cost minimization. Neyman Pearson (NeP) hypothesis is useful in situations where different types of error have different consequences. Such applications include fraud detection, spam filtering, machine monitoring, target recognition, and disease diagnosis [19]. In most applications, class probabilities differ for training and testing data. But NeP does not assume prior knowledge of class probabilities. It automatically balances model complexity and training error. It is used to increase the detection accuracy rate with known maximum false alarm threshold 'μ' where μ is in the range [0,1].

The objective of using Neyman Pearson approach in this paper is to reduce the classification error costs. The cost minimization in NeP [11] has the following steps:

- Calculation of total number of samples belonging to each class
- Calculation of misclassified samples belonging to each class
- Calculation of the optimum threshold and display of final classification decision (Attack / No Attack)

3.2.1.2. Calculation of total number of samples for each class. The total number of samples from class j is calculated using equation (10).

$$\mathbf{n}_j = \sum_{i=1}^{n} I\{Y_i = j\} \tag{10}$$

Let $Z_n = \{X_i, Y_i\}$ where $i = 1,\ldots,$n be a collection of 'n' independent and identically distributed samples of $Z = (X,Y)$. A learning algorithm is a mapping function, h_n: $Z_n \to H(X,Y)$, where $H(X,Y)$ is the set of all classifiers. h_n is a rule for selecting a classifier based on training sample. 'I' denotes the indicator function. n_j is the total number of samples for class 'j'.

3.2.1.3. Calculation of total number of misclassified samples for each class. The efficiency of classifier may be measured in terms of the metric, the total number of misclassified instances, which is calculated using Eq. (11).

$$R_j(h) = 1/n_j \sum_{i=Y_i=j} I\{h(i) \neq j\} \tag{11}$$

$R_j(h)$ denotes the false positives corresponding to $j = 0, j = 1$. n_j is the total number of samples for class 'j'. I is the indicator function which outputs either 0 (if the sample is correctly classified as class j) or 1 (if the sample is not classified as class j). The number of false positives is calculated by dividing the sum of misclassified samples by the total number of samples, n_j.

3.2.1.4. Finding optimum threshold. A Receiver Operating Characteristic (ROC) curve is plotted against false positive (x-axis) and detection accuracy (y-axis). The line $y = x$ shows random guessing (half of the samples are misclassified). If the misclassifications are more than 50%, then the performance of Intrusion Detection system (IDS) will be poor. So, the default false positive decision threshold is chosen as 0.5. From the list of classifiers producing false positives, the classifier with the fewer false positives are considered as optimum threshold. Optimum threshold is calculated using Eq. (12).

$$h^* = \arg\min\{R_j(h) : h \in H_0\} \quad \text{where} \quad H_0 = \{h \in H : R_0(h) \leqslant \mu\}. \tag{12}$$

New instance classification decision is based on the new optimum threshold h^*. h^* is optimal with respect to classes of randomized tests/classifiers. Thus, the classification accuracy is improved by combining the outputs of different classifiers. After determining optimum thresholds for both the classes, the class corresponding to the output that exceeds its threshold is selected as the classification

Table 5
Comparison of existing ensemble of classification algorithms with proposed algorithm.

Algorithm / Features	Bagging	Boosting	AdaBoost	NFBoost
Prior knowledge of data distribution	Not required	Not required	Required	Required
Method used to combine classifiers	Majority Voting	Three-way Majority Vote	Weighted Majority Voting	Weighted Mean
Number of classifiers	Given as input	Three	Given as input	Cross Validation
Data subset for training operation	Draws randomly some fraction from the Dataset	Creates informative data subset	Draws from the updated data distribution	Draws from the updated data distribution
Classification error (cost) minimization	No	No	No	Yes (Neyman Pearson approach)
Weight updation to false positive instances	No	No	No	Yes
Drawbacks	Works for small subset	Limited to binary classification problems. Sensitive to noise and outliers	Prior knowledge of data distribution needed before generating hypothesis. Frequent retraining needed	Large amount of network traffic data is required
Advantages	Simple to implement with good performance	Most informative dataset provided to each classifier	Capable of handling multi-class and regression problems	No retraining of classifiers. Adaptive and Incremental Learning

decision. Voting methods such as simple Majority Voting, Weighted Majority Voting, etc., provide the solution in case if multiple outputs exceed their thresholds.

3.2.2. Testing

An unlabeled instance '*X*' is given as input to the trained ensemble. The ensemble of each subset of particular class is evaluated on given instance '*X*'. The ensemble of each subset outputs a class label by weighted mean approach. The class that receives the highest total is the final classification decision.

3.3. Comparison of NFBoost with the existing ensemble algorithms

Comparison of the proposed NFBoost classification algorithm with existing ensemble algorithms such as bagging, boosting, and AdaBoost is shown in Table 5. Bagging, for bootstrap aggregating, is an ensemble method for improving unstable estimation or classification schemes. Boosting algorithm creates three weak classifiers. If first two classifiers agree on the same class, that class is the final classification decision. If these two classifiers disagree, then class chosen by the third classifier is the final decision. The three classifiers are combined through a three-way majority vote. Prior knowledge of data distribution is required for Adaboost algorithm. In Adaboost, the hypotheses are generated by training a weak classifier, using instances drawn from an iteratively updated distribution of the training data. The distribution update ensures that instances misclassified by the previous classifier are included in the training data of the next classifier. Majority Voting is used to combine the ensemble of classifier outputs for bagging and Adaboost.

From Table 5, it is seen that our proposed NFBoost differs from AdaBoost in classification error minimization and in instances weight update method. The false positive instances are assigned more weights than the other misclassified instances. In the next iteration of training, the false positives are classified correctly by the trained network as higher weight instances were given priority in training. Also, in post-training step process, optimum threshold for each class was found by the NeP method and final classification depends on the new optimum threshold.

Our earlier work, RBPBoost [11], proposed an ensemble of neural classifier for DDoS attack detection. Comparison of RBPBoost and NFBoost are shown in Table 6. From Table 6, it is evident that NFBoost is adaptive and learns new classes of data. False positive cost reduction is not considered during training in RBPBoost, but NFBoost reduces the false positive cost during all training iterations. But, Neyman Pearson hypothesis has been chosen in both RBPBoost and NFBoost as false positive cost minimization technique in post-training.

The problem of adaptive learning differs significantly from incremental learning. The assumption of incremental learning is that class labels are fixed, but data points about these given classes are assimilated one at a time. However, in our case, a new or a few completely new classes of data become available and the model is required to be able to classify them without compromising its ability to classify existing classes. The generalization error decreases with increasing margin, and NFBoost maximizes these margins by increasing the confidence of the ensemble's decision. Since the base model learning algorithm is required to have an error less than 0.5, it is guaranteed to classify correctly the previously misclassified training instances.

4. Experimental results

Three experiments were carried out on publicly available datasets such as KDD cup (1999) dataset [33], Conficker dataset [17], UNINA dataset [65], SSE Lab Dataset, and CAIDA DDoS dataset [16] using MATLAB and the results are discussed in Sections 4.1, 4.2, and 4.3. The KDD cup (1999) dataset, though old, have been used to test our algorithm against the existing ones in the literature. Further, DDoS attack traffic was generated in our lab to test the efficiency of our proposed NFBoost algorithm with real attack traces. Confusion matrix and cost matrix are shown in Table 7 and Table 8. In this paper, it has been assumed that the cost of false positive is four times higher than false negative as shown in Table 8. As the objective of our proposed algorithm is to prevent denying access to legitimate users, the false alarm cost was assumed higher. For the simulation experiments conducted, detection accuracy was calculated using (13).

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \tag{13}$$

True positive (TP) = Number of samples correctly predicted as attack class

False positive (FP) = Number of samples incorrectly predicted as attack class

Table 6
Comparison of RBPBoost and NFBoost.

RBPBoost	NFBoost
Neural classifier as base classifier	Neuro-fuzzy classifier as base classifier
Incremental learning	Adaptive and incremental learning
Ensemble output combined using weighted majority voting and weighted product rule	Ensemble output combined using weighted mean and majority voting
No assignment of weights to instances	Assignment of weights to instances
No false positive cost reduction during training and NeP cost reduction during Post-Training	False positive cost reduction during every iteration of training and post-training using NeP cost reduction

True negative (TN) = Number of samples correctly predicted as normal class

False negative (FN) = Number of samples incorrectly predicted as normal class.

The metric cost function was used to facilitate performance comparison of the existing ensemble methods with our proposed NFBoost algorithm. Cost function is based on the number of samples that are misclassified. It was calculated using (14). λ is the parameter for cost difference between false alarm and miss. The value of λ was set as 4 for the experiments. Any value could be assumed for λ. The cost function for various bagging, boosting, AdaBoost, and NFBoost algorithms are calculated using (14). The ensemble with least cost function emerges out as the best detection system.

$$\text{Cost} = (1 - \text{Detection Accuracy rate}) + \lambda(\text{False positive rate}) \tag{14}$$

4.1. Experiment 1 – KDD cup dataset

KDD cup dataset is a standard set of data collected through the 1998 DARPA intrusion detection evaluation program at the MIT Lincoln Labs. The data set includes a wide variety of intrusions simulated in a military network environment. It is a 41-feature database and grouped into three groups: intrinsic (9), traffic (13), and content (19) features. There are five classes, viz., four different types of networks attacks such as Denial of Service (DoS), Probe, UserToRoot (U2R), RootToLocal (R2L), and one normal traffic. As our work is towards the detection of DDoS attacks, Probe, U2R, and R2L are not included in the dataset used for training and testing. KDD cup dataset used consisted of 4898,430 connection records. Removing the duplicate records, the remaining traffic consisted of 812,813 normal connection records and 247,267 DoS connection records. Among these, 300,000 records comprising of both normal and DoS attack traffic were used for training and testing the proposed and existing algorithms.

Using all 41 features present in KDD cup dataset could result in a huge Intrusion detection model, which could be an overhead for online detection. Hence, number of features was reduced. Out of 41 features, 6 features such as Protocol type, Service, Number of connections from the same host, number of unauthorized root accesses, number of connections that have SYN errors, and number of connections that have REJ errors have been considered in this paper due to our focus on the detection of DDoS attacks. Further, these feature values were normalized using SOM toolbox in MATLAB. The normalized values were given as inputs to the Neuro-fuzzy Classifier for training, validation, and testing. The experiment was repeated with random splits of training, validation, and testing data, and the results obtained were summarized in Table 9.

Table 7
Confusion matrix.

	Normal	Attack
Normal	TN	FP
Attack	FN	TP

Table 8
Cost matrix.

	Normal	Attack
Normal	0	4
Attack	1	0

Our NFBoost algorithm was trained with KDD cup dataset. KDD cup dataset contains DoS attacks such as back, land, Neptune, pod, smurf, and teardrop. From KDD cup dataset, attacks such as smurf and land were not considered for training. Hence, these attack traffic (unseen samples) were given as input during testing in order to test the capability of our NFBoost algorithm. The trained ensemble was tested and the simulation results confirm that the detection rate was high and able to detect the new attacks.

Our proposed classification algorithm with cost minimization strategy is compared with existing ensemble methods [15], [56], and [28] in terms of cost per sample on KDD datasets and it is inferred that the trained network is able to generalize well and detect the new traffic patterns. From Table 9, it can be seen that the detection accuracy is 98.2% with 1.7% false positives. Comparing the False positive rate of NFBoost with and without cost minimization, it can be seen that NFBoost with cost minimization yields an improvement of 39.28% $\left(\frac{2.8-1.7}{2.8} \times 100\right)$ over NFBoost without cost minimization. Further, it is evident that NFBoost with cost minimization achieves a maximum gain of 6.4% and a minimum gain of 2.1% in detection accuracy. Moreover, it can be inferred from Fig. 5 that cost per instance is less for NFBoost with cost minimization than the other methods.

4.2. Experiment 2 – mixed traffic

The second experiment was conducted for a mixed traffic by obtaining data from different sources to train the classifiers where each classifier becomes an expert in particular application classification. Information coming from different data sources makes the ensemble a completely versatile classifier. Due to the availability of different data sources, data fusion was realized. The number of training and testing instances used in the experiment is shown in Table 10. The following publicly available datasets were used for training the classifier:

- Conficker
- CAIDA
- UNINA

The Conficker [21] dataset contains data from the UCSD Network Telescope for three days between November 2008 and January 2009. The first day (21 November 2008) covers the onset of the Conficker A infection. On the second day, 21 December 2008, only Conficker A was active, and during the third and final day both Conficker A and B was active. The dataset contains 68 compressed

Table 9
Simulation results of classification algorithms for kdd dataset.

Algorithm	Detection accuracy (%)	False positive (%)
Bagging	91.8	6.7
Boosting	92.7	5.8
AdaBoost	94.6	3.5
NFBoost	96.1	2.8
NFBoost + Cost Minimization	98.2	1.7

Table 11
Simulation results of classification algorithms for mixed dataset.

Algorithm	Detection accuracy (%)	False positive (%)
Bagging	90.4	8.1
Boosting	90.6	6.2
AdaBoost	93.7	4.8
NFBoost	97.2	4.6
NFBoost + Cost Minimization	98.8	1.9

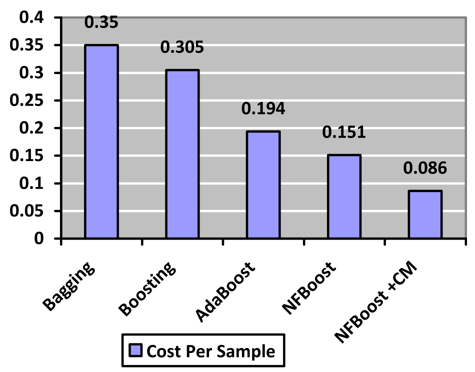


Fig. 5. Cost Per sample for existing ensemble classifiers and NFBoost.

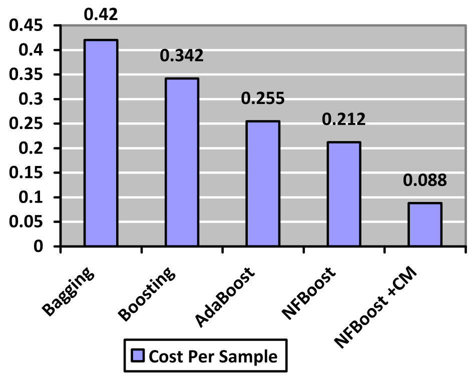


Fig. 6. Cost per sample for existing ensemble classifiers and NFBoost.

pcap files each containing one hour of traces. The total size of the dataset for all the three days is 69 GB. The pcap files only contain packet headers; payload has been removed. Out of the 68 compressed pcap files, 20 compressed pcap files (only attack traffic) from three days were used for training and testing.

CAIDA DDoS Attack 2007 [16] dataset contains approximately one hour of anonymized traffic traces from a DDoS attack on August 4, 2007. This type of denial-of-service attack attempts to block access to the targeted server by consuming computing resources on the server and by consuming all of the bandwidth of the network connecting the server to the Internet. The one-hour trace is split up in 5-min pcap files. The total size of the dataset is 5.3 GB. Three 5 min pcap files out of one hour trace were used for training, validation, and testing. Only attack traffic to the victim and responses to the attack from the victim were included in the traces. DDoS Attack traffic consists of Ping ICMP flood, TCP SYN flood, and HTTP requests. Non-attack traffic has been removed. Traces in this dataset have been anonymized using CryptoPAn prefix-preserving anonymization using a single key. The payload has been removed from all packets.

The UNINA dataset [65] refers to traffic traces captured by passively monitoring incoming traffic at the WAN access router at University of Napoli "Federico II". D-ITG [65] is a traffic generation and active measurement software architecture, which replicates several network traffic profiles and performs packet-level measurements. Traffic traces were captured from real networks using Planet Lab [50]. The link observed was a link at 200 Mbps connecting the University of Napoli "Federico II" network to the rest of the Internet.

These traces are in tcpdump format. Out of the 3 traces for port 80 traffic, one trace of one hour duration has been chosen in our experiment. Packet lengths are variable as each packet contains full TCP headers including optional headers (e.g., MSS). The traffic related to TCP port 80 generated by clients inside the network of University of Napoli, Federico II to the outside world were used for training and testing. Similarly, traces collected during September 2005, related to TCP port 25 (SMTP traffic) generated by clients inside the network of University of Napoli, Federico II to the outside world were used for training, validation, and testing. Out of the 3 traces of SMTP port 25 traffic, one trace has been chosen in our experiment.

The test data is not from the same probability distribution as the training data, and it includes specific attack types not in the training data. This made the classification task more realistic. From Table 11, it can be seen that the detection accuracy is 98.8% with 1.9% false positives. Comparing the False positive rate of NFBoost with and without cost minimization, it can be seen that NFBoost with cost minimization yields an improvement of 78.26% over NFBoost without cost minimization. Further, it is evident that NFBoost with cost minimization achieves a maximum gain of 8.4% and a minimum gain of 1.6% in detection accuracy. Moreover, it can be inferred from Fig. 6 that cost per instance is less for NFBoost with cost minimization than the other methods.

4.3. Experiment 3

Experiment 3 was conducted in an emulated SSE testbed in our lab as shown in Fig. 7. The environment that has been used for generating network traffic was similar to Planet Lab. Target machine was in Site 4. Client machines in all sites were used to generate both normal and attack traffic. In a single client machine, many virtual clients were created with spoofed and random IP addresses to

Table 10
Training and testing instances used in experiment 2.

Datasets	Total number of connections	Number of training instances	Number of testing instances
CAIDA Conficker	1812,685	7280	2720
CAIDA DDoS 2007	409,073	630	270
UNINA (port 80 and port 25)	1214,485	5040	2160

generate both attack and normal traffic. The exact real attack traffic data in the recent past, viz., Twitter, Total Choice hosting Network, Egyptian government websites, etc., attack environments was not made publicly available due to confidential, privacy, and legal reasons. This motivated us to create our own datasets. So, normal and attack traffic data were created in our Smart and Secure Environment Project Laboratory. For normal traffic, video conferencing application (15 min UDP traffic), random legitimate requests from virtual clients to web server (HTTP traffic), and file sharing application (FTP traffic), were generated in SSE testbed. A category of traffic that is now constantly increasing in Internet is peer-to-peer video streaming. Hence, traffic generated by the video conferencing application has been considered. The traffic falls in the UDP based application category. SSE Testbed provides a flexible envi-

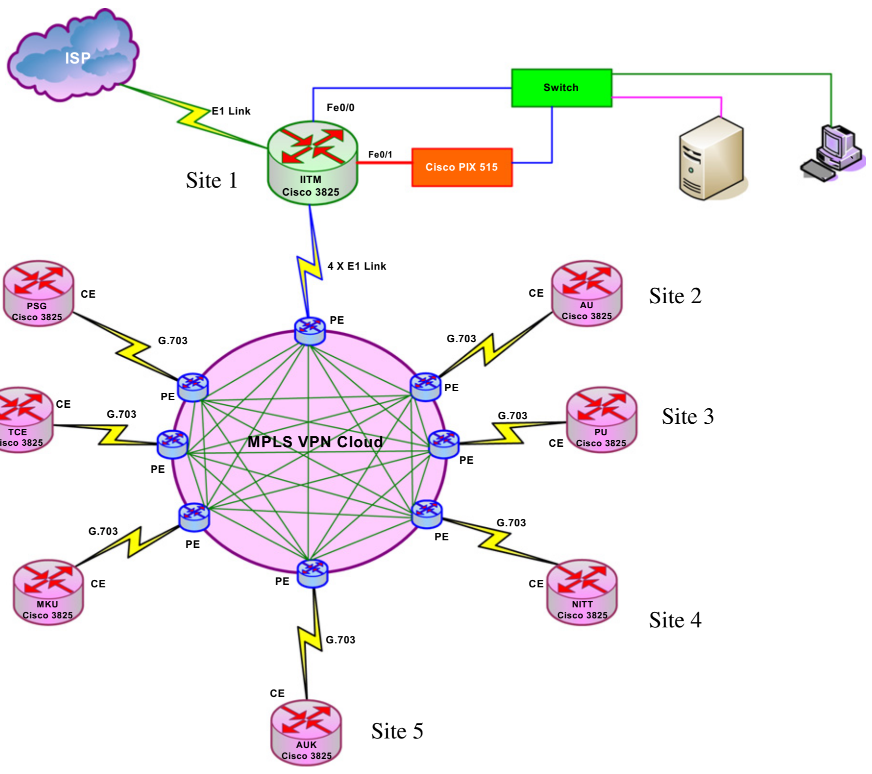
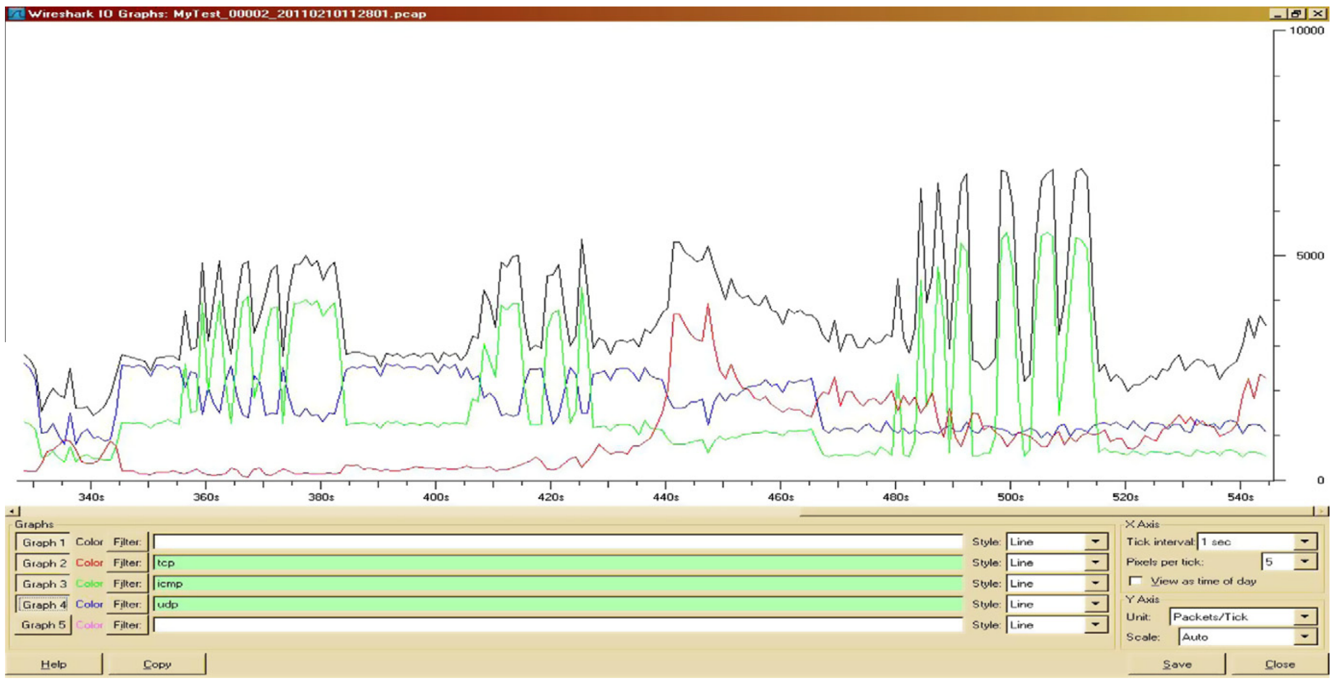


Fig. 7. SSE environment.



Graphs

Graph 1 Color Filter: Style: Line
Graph 2 Color Filter: tcp Style: Line
Graph 3 Color Filter: icmp Style: Line
Graph 4 Color Filter: udp Style: Line
Graph 5 Color Filter: Style: Line

X Axis
Tick interval: 1 sec
Pixels per tick: 5
View as time of day

Y Axis
Unit: Packets/Tick
Scale: Auto

Fig. 8. Attack traffic trace on 09.02.2011 at 11.30 a.m. (30 min) in SSENET.

ronment where researchers choose systems for attack from any one of our sites and test their proposed protocols or detection algorithms. Attack traffic generation were successively repeated in order to increase the amount of data collected. As the number of packets for analysis increases, the classification performance increases and the testing error reduces. Attacks such as SYN flood, UDP flood (DNS Cache Poisoning), ICMP flood, and HTTP flood were generated in SSE Testbed.

All client machines in the Intranet are connected through a backbone bandwidth of 1 Gbps and to different sites through a 2 Mbps MPLS VPN cloud. As Target (victim) server was available in Site 4, traces were collected in Site 4 in tcpdump format. For example, TCP port (80) traffic was generated by the clients in all sites including target server site. The traffic was collected during working hours. There were two types of traffic generated in the network, viz., legitimate and attack traffic. DDoS attack was launched on the Web server present in Site 4.

Using Metasploit framework [41], Netwag tool [47], and TFN2K program, DDoS attack traffic with varying packet lengths were generated. Packets were captured from the network using Wireshark tool, which is based on the Libpcap library and in capture mode, a filter was used to monitor traffic for WWW, TCP SYN, UDP, and IP as shown in Fig. 8. It is evident that there is an abrupt spurt in the network traffic over a period of 30 min. During the same period, normal traffic was generated from our virtual clients. Similarly next day over a period of 30 min, attack traffic were generated with different traffic rate. Tcpdump [61] and tcptrace [62] were used to extract the features from the captured packets of 2 s time window. Data obtained from feature extraction were used for training the NFBoost algorithm. Simulation results were tabulated in Table 12.

From Table 12, it is evident that our proposed NFBoost algorithm with cost minimization strategy achieves high detection accuracy than the existing algorithms. Also, in classification, video conferencing application traffic and file sharing application traffic were detected correctly without any misclassifications. From Table 12, it can be seen that the detection accuracy is 99.2% with 1.2% false positives. Comparing the False positive rate of NFBoost with and without cost minimization, it can be seen that NFBoost with cost minimization yields an improvement of 57.14% over NFBoost without cost minimization. Further, it is evident that NFBoost with cost minimization achieves a maximum gain of 5.1% and a minimum gain of 3.7% in detection accuracy. Moreover, it can be inferred from Fig. 9 that cost per instance is less for NFBoost with cost minimization than the other methods. Neyman-Pearson (NeP) classification minimizes the miss rate while ensuring the false alarm rate to be less than a specified threshold. Thus from the experimental values, NeP is proven to be better suited for binary classification problems.

4.4. Observations from the experiments

In our simulation experiments, the dataset is split into training dataset and testing dataset. The testing dataset was served as

Table 12
Simulation results of classification algorithms for sse dataset.

Algorithm	Detection accuracy (%)	False positive (%)
Bagging	94.1	4.8
Boosting	94.6	4.2
AdaBoost	96.4	3.2
NFBoost	96.9	2.8
NFBoost + Cost minimization	99.2	1.2

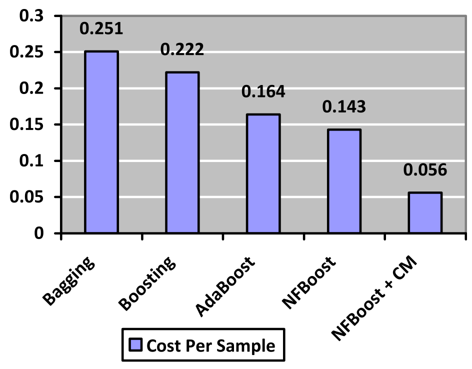


Fig. 9. Cost per sample for existing ensemble classifiers and NFBoost.

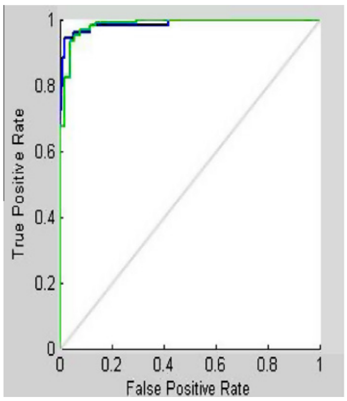


Fig. 10. ROC curve of NFBoost with cost minimization for experiment 1.

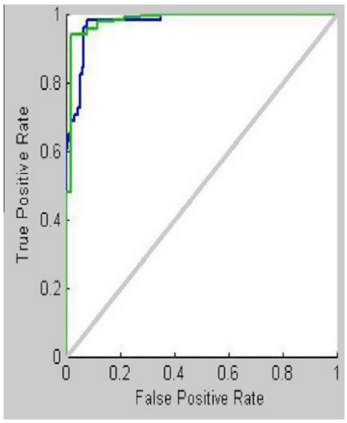


Fig. 11. ROC curve of NFBoost with cost minimization for experiment 2.

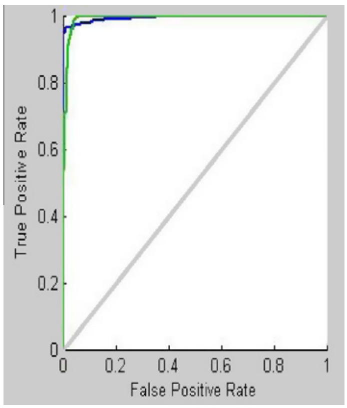


Fig. 12. ROC curve of NFBoost with cost minimization for experiment 3.

Table 13
Performance Comparison Of NFBoost With Existing Ensemble Methods In Terms Of Consuming Time.

Dataset	Algorithms	Preprocessing time (seconds)	Training time (seconds)	Testing time (seconds)
KDD cup dataset (3,00,000 instances)	Bagging	0.01	3.10	1.30
	Boosting	0.01	2.40	1.30
	AdaBoost	0.01	3.14	1.40
	NFBoost	0.01	3.22	1.40
	NFBoost + Cost Minimization	0.01	3.26	1.42
Mixed Traffic (18,000 instances)	Bagging	0.001	1.34	0.30
	Boosting	0.001	1.22	0.20
	AdaBoost	0.001	1.38	0.30
	NFBoost	0.001	1.36	0.30
	NFBoost + Cost minimization	0.001	1.42	0.36
SSE Lab (14,000 instances)	Bagging	0.001	1.14	0.20
	Boosting	0.001	1.60	0.12
	AdaBoost	0.001	1.26	0.24
	NFBoost	0.001	1.26	0.22
	NFBoost + Cost minimization	0.001	1.40	0.32

completely unseen samples to the trained ensemble. Testing dataset contains some attack types which were not included in the training dataset and poses a challenge to test the ability of NFBoost algorithm in detecting the new attack types. The performance of the ensemble of neuro-fuzzy classifiers was evaluated based on the testing dataset. From the simulation experiments, it was evident that the trained ensembles were able to detect new attacks.

Receiver Operating Characteristic (ROC) curve for all the experiments conducted were plotted against False positive rate and Detection accuracy rate as shown in Figs. 10–12. From these figures, it is evident that the True positive rate is high with fewer false alarms. Also, the testing curve is not visible in the ROC graph as the false positives are very less. Hence, only training and validation lines are visible in graphs. During training, the false positives were more and with successive rounds of boosting the network, the false positives were reduced gradually as can be seen in Figs. 10–12.

During learning, the instances that have not been properly learnt by the entire ensemble are assigned more weights. Hence, during next iteration, the instances that were assigned more weights or misclassified in the previous iteration were collected as a new dataset. A new instance of data is learnt incrementally to the trained ensemble as shown in Step 2 of NFBoost algorithm. The Learning ensemble learns the new data without forgetting the previously acquired knowledge. So, a new instance of data is learnt without discarding the existing classifier and retraining a new one. Hence, Ensemble once trained does not require retraining in real time, instead learning on its own every time using the traffic other than trained ones. It is evident from the experiments that the NFBoost algorithm classifies intrusions with more detection accuracy. Thus, the proposed framework is suitable for real-time scenario for detecting the new attacks.

4.5. Computational complexity of NFBoost

Preprocessing time includes the time spent in feature extraction and normalization. The training time depends on the number of times the classifier needs training which in turn depends on the mean square error between iterations reaching global minimum. Testing time includes the time spent in testing the unlabeled instances by weighted mean and majority voting algorithm.

Table 13 shows the performance comparison of the NFBoost in terms of consuming time obtained during the experiments. From Table 13, it can be seen that NFboost consumes the similar preprocessing time with the existing ensemble methods but consumes more training and testing time than the existing ensemble methods. Classification error cost minimization technique is proposed in NFboost technique but not in the existing ensemble methods.

Table 14
Computational complexity of NFBoost in different layers.

Layers	Number of nodes	NFBoost
Input	N	6
Membership function	$N \times P$	6×3
Rules	P^N	3^6
Normalization	P^N	3^6
De-fuzzification	P^N	3^6
Ouput (Sum)	1	1

So, the training and testing phases consume little more time than the existing ensemble methods.

Testing time is little high due to the ensemble output combination methods such as weighted mean and majority voting algorithm, but more detection accuracy and less false positives were achieved in NFBoost. The computational complexity of our proposed algorithm NFBoost is based on the time spent on preprocessing module and classification module. Time spent for classification includes training the ensemble of neuro-fuzzy classifiers and decision making process. Computational complexity of our proposed NFBoost is shown in Table 14. 'N' denotes the number of inputs (features) and 'P' denotes the number of membership functions. The computational complexity of testing process is $O(P^N N)$, where 'P^N' is the number of rules. The testing time is seen to increase if the number of inputs or rules is increased. In our proposed NFBoost, testing time is less due to less number of features and rules.

The increased performance of ensemble of neuro-fuzzy classifiers comes at a cost of increased complexity. This level of computational complexity is favorable in order to achieve high detection accuracy with few false positives. Also, the computational time was calculated on Intel 2.93 GHz, Core 2 Duo Processor, 2 GB RAM computer. Optimizing the antecedent part and consequent part of neuro-fuzzy system is different research area, which is not the focus in the paper. As ANFIS model, which has been used in this paper, employs gradient descent to adjust the membership function parameters, it is computationally less expensive and is significantly advantageous for larger networks. The speedup of NFBoost can be improved when ensemble of classifier is executed in parallel processors or more than one micro engine of a (network) processor. Thus, all the modules can be processed in parallel by different engines in order to reduce the overall processing time considerably.

5. Conclusion

Nowadays, Botnet based DDoS attack happens with legitimate traffic. Even with the help of stored attack signatures, it is tedious to detect the DDoS attacks. So, the challenge lies in discriminating the normal traffic and DDoS attack traffic. Neural networks fail sometimes with imprecise and noisy input data. Fuzzy logic extracts the comprehensible rules which cannot be extracted by neural network. An approach based on hybrid neuro-fuzzy system is proposed in this paper. Our contributions include a novel weight update distribution strategy, and our proposal differs from the existing methods in weight update distribution strategy, error cost minimization, and ensemble output combination method. The performance of the ensemble of neuro-fuzzy classifiers was evaluated based on the testing dataset which contains attack types not included in the training set. From the simulation experiments, it was evident that the trained ensembles were able to detect new attacks. Hence, the Learning ensemble learns the new data without forgetting the previously acquired knowledge, discarding the existing classifier, and retraining a new one.

Our proposed detection algorithm can deal with both discrete and continuous attributes in the database, which is practically useful for real time network datasets. The main objective in this paper is to provide an efficient false positive reduction technique to minimize the false alarms. NFBoost algorithm proposed in this paper demonstrates the use of Neyman Pearson approach as a post-training step to minimize the cost of misclassification errors. From the simulation results, it has been found that the NFBoost classification algorithm results in high detection accuracy of 99.2%. Further, it has been observed from the results that NFBoost algorithm outperforms the existing algorithms with a maximum gain of 8.4% and a minimum gain of 1.1%. Hence, our proposed NFBoost algorithm with less statistical features is suitable for real-time detection of novel and existing DDoS attacks.

Acknowledgements

The authors are grateful for the sponsorship of this research work provided by the Government of India, New Delhi, under the Collaborative Directed Basic Research (CDBR) in Smart and Secure Environment (SSE) Project.

References

[1] P. Aki, F. Chan, Daniel S. Yeung, Eric C.C. Tsang, Wing W.Y. Ng, "Empirical study on fusion methods using ensemble of RBFNN for network intrusion detection", ICMLC 2005, vol. 3930, 2005, pp. 682–690.

[2] Amey Shevtekar, Karunakar Anantharam, Nirwan Ansari, Low rate TCP denial-of-service attack detection at edge routers, IEEE Communications Letters 9 (4) (2005) 363–365.

[3] Amey Shevtekar, J. Stille, N. Ansari, "On the impacts of low rate DoS attacks on VoIP traffic," Security and Communication Networks, vol. 1, no. 1, pp. 45-56, February 2008.

[4] Amey Shevtekar, Nirwan Ansari, A router-based technique to mitigate reduction of quality (RoQ) attacks, Computer Networks 52 (5) (2008) 957–970.

[5] Amey Shevtekar, Nirwan Ansari, Is it congestion or a DDoS attack?, IEEE Communications Letters 15 (7) (July 2009) 546–548

[6] Andrey Belenky, Nirwan Ansari, IP traceback with deterministic packet marking, IEEE Communications Letters 7 (4) (April 2003) 162–164.

[7] Andrey Belenky, Nirwan Ansari, "On deterministic packet marking", Computer Networks, 51(10) (2007) 2677–2700. July 11.

[8] Arbor, Networks, "Worldwide infrastructure security report", vol.VI, November 2010.

[9] P. Arun Raj Kumar, S. Selvakumar, "A DDoS threat in collaborative environment – A survey on DDoS attack tools and Traceback mechanisms", in: Proceedings of IEEE International Advance Computing Conference (IACC'09) 2009, pp. 1275–1280.

[10] P. Arun Raj Kumar, S. Selvakumar, "Mathematical modeling of DDoS attack and defense – A survey" in: The 3rd International Conference on Computer Modeling and Simulation (ICCMS 2011), Mumbai, Maharashtra, January 7–9, vol. 2, 2011, pp. 85–89.

[11] P. Arun Raj Kumar, S. Selvakumar, "Distributed denial of service attack detection using an ensemble of neural classifier", in: International Journal of Computer Communications, Elsevier Publications, United Kingdom, Vol. 34, No. 11, 2011, pp. 1328–1341.

[12] T. Auld, A.W. Moore, S.F. Gull, "Bayesian neural networks for Internet traffic classification", IEEE Transactions on Neural Networks, 8(1) (2007) 223-239.

[13] D.R. Barbosa, Achanccaray, M. Vellasco, M.A. Meggiolaro, R. Tanscheit, "Mental tasks classification for a noninvasive bci application", in: 19th International Conference on Artificial Neural Networks, ICANN'09, Limassol, Cyprus, 2009.

[14] A. Belenky, N. Ansari, On IP traceback, IEEE Communications Magazine 41 (5) (July 2003) 142–153.

[15] L. Breiman, Bagging predictors, Machine Learning 24 (2) (1996) 123–140.

[16] The CAIDA, "DDoS Attack 2007", Dataset Paul Hick, Emile Aben, kc claffy, Josh Polterock. Available from <http://www.caida.org/data/passive/ddos-20070804dataset.xml>.

[17] CAIDA UCSD Network telescope, "Three days of conficker" – November 2008, Paul Hick, Emile Aben, Dan Andersen, kcclaffy. Available from <http://www.caida.org/data/passive/telescope-3days-conficker_dataset.xml>.

[18] J. Canul-Reich, L. Shoemaker, L.O. Hall, "Ensembles of fuzzy classifiers", in: IEEE International Fuzzy Systems Conference, 2007, London, http://dx.doi.org/10.1109/FUZZY.2007.4295345.

[19] Clayton Scott, Robert Nowak, "A Neyman Pearson Approach to statistical learning", Technical Report, TREE 0407.

[20] CNN, "DDoS attacks on Yahoo, Buy.com, eBay, Amazon, Datek, E Trade", CNN Headline News, 2000.

[21] Daniel Neagu, Shuai Zhang, Catalin Balescu, "A Multi-label Voting Algorithm for Neuro-fuzzy classifier ensembles with applications in visual arts data mining", in: Proceedings of 5th International Conference on Intelligent Systems Design and Applications, Wroclaw, Poland, 2005, pp. 245-250.

[22] David Wolpert, Stacked generalization, Neural Networks 5 (1992) 241–259.

[23] Devi Parikh, Tsuhan Chen, Data fusion and cost minimization for intrusion detection, IEEE Transactions on Information Forensics and Security 3 (3) (September 2008) 381–389.

[24] Y. Dhanalakshmi, I. Ramesh Babu, Intrusion detection using data mining along fuzzy logic and genetic algorithms, International Journal of Computer Science and Network Security 8 (2) (2008) 27–32.

[25] Dimitris Gavrilis, Evangelos Dermatas, Real time detection of distributed denial-of-service attacks using RBF networks and statistical features, Computer Networks 44 (5) (2005) 235–245.

[26] W. Fan, S. Stolfo, "Ensemble-based adaptive intrusion detection", in: SIAM International Conference on Data Mining, 2002.

[27] Flashget. http://www.flashget.com.

[28] Y. Freund, R.E. Schapire, Decision-theoretic generalization of on-line learning and an application to boosting, Journal of Computer and System Sciences 55 (1) (1997) 119–139.

[29] Hiroshi Tsunoda, Kohei Ohta, Atsunori Yamamoto, Nirwan Ansari, Yuji Waizumi, Yoshiaki Nemoto, "Detecting DRDoS attacks by a simple response packet confirmation mechanism", Computer Communications, 31(14) (2008) 3299–3306. September 5.

[30] Hoai-Vu Nguyen, Yongsun Choi, "Proactive detection of DDoS attacks using k-NN classifier in an Anti-DDoS Framework", in: International Journal of, Computer Systems Science and Engineering, 2008, pp. 247–252.

[31] Isabelle Guyon, Steve Gunn, Masoud Nikravesh, Lotfi. A. Zadeh, "Feature Extraction: Foundation and Applications", Physica-Verlag, Springer, Ch.1, 2006.

[32] Jane Grossman, Michael Grossman, Robert Katz, "The first systems of weighted differential and integral calculus", ISBN 0977117014, 1980.

[33] KDD data set, 1999; <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>.

[34] M. Korytkowski, R. Nowicki, L. Rutkowski, R. Scherer, "Merging ensemble of neuro-fuzzy systems", in: IEEE International Fuzzy Systems Conference, 2006, Vancouver, http://dx.doi.org/10.1109/Fuzzy.2006.1681971.

[35] Y. Li, Y. Ge, X. Jing, Z. Bo, "A new intrusion detection method based on fuzzy Hidden Markov model", in: The Proceedings of 3rd IEEE Conference on Industrial Electronics and Applications, Singapore.

[36] C.-C. Lin, M.-S. Wang, Genetic-clustering algorithm for intrusion detection system, International Journal of Information and Computer Security 2 (2) (2008) 218–234.

[37] X. Luo, R. Chang, "On a new class of pulsing denial-of service attacks and the defense", in: The Proceedings of Annual Network and Distributed System Security Symposium (NDSS), Feb. 2005.

[38] Marcin Korytkowski, Leszek Rutkowski, Rafal Scherer, "From ensemble of fuzzy classifiers to single fuzzy rule base classifier", in: ICAISC 2008, LNAI 5097, pp. 265–272.

[39] Marcin Korytkowski, Robert Nowicki, Rafal Scherer, Leszek Rutkowski, "Ensemble of rough neuro-fuzzy systems for classification with missing features", in: IEEE International Conference on Fuzzy Systems (FUZZ 2008), Hong-Kong, 2008, pp. 1745–1750.

[40] Marcin Gabryel, Marcin Korytkowski, Agata Pokropinska, Rafał Scherer, Stanisław Drozda, "Evolutionary learning for neuro-fuzzy ensembles with generalized parametric triangular norms", in: ICAISC 2010, Part I, LNAI 6113, 2010, pp. 74–79.

[41] Metasploit Framework. Available from <http://www.metasploit.org>.

[42] Michael Jordan, R. Jacobs, Hierarchical mixtures of experts and the EM algorithm, Neural Computation 6 (2) (1994) 181–214.

[43] T. Mitchell, Machine Learning, McGraw-Hill Education (ISE Editions), 1997.

[44] Ming-Hour Yang, Ming-Chien Yang, RIHT: a novel hybrid traceback scheme, IEEE Transactions on Information Forensics and Security 7 (2) (April 2012) 789–797.

[45] A.W. Moore, D. Zeuv, "Discriminators for use in flow-based classification", Intel Research Technical Report, 2005.

[46] R.T. Morris,"A Weakness in the 4.2BSD UNIX TCP/IP Software", Computer Science Technical Report No.117, AT&T Bell Laboratories, Murray Hill, NJ.

[47] Netwag Tool. Available from <http://ntwag.sourceforge.net/>.
[48] J.S. Park, K.M. Shazzad, D.S. Kim, "Toward modeling lightweight intrusion detection through correlation-based hybrid feature selection", Information Security and Cryptolo, in: First SKLOIS Conference, CISC 2005, Beijing, China, 2005, pp. 279–289.
[49] Philip Chan, Salvatore Stolfo, "Experiments on multi-strategy learning by meta-learning", in: International Conference on Knowledge and, Information Management, 1993, pp. 314–323.
[50] Planet Lab. Available from <http://www.planet-lab.org/>.
[51] R. Polikar, Ensemble based systems in decision making, IEEE Circuits and Systems 6 (September 2006) 21–45.
[52] Rafa Scherer, "Boosting ensemble of relational neuro-fuzzy systems", in: ICAISC 2006, LNAI 4029, 2006, pp. 306–313.
[53] Rasool Jalili, Fatema Imani–mehr, Morteza Amini, Hamid Reza shahriari,"Detection of DDoS attacks using statistical preprocessor and unsupervised neural networks", LNCS 2005, pp. 192–203.
[54] J.S. Roger Jang, ANFIS adaptive-network-based fuzzy inference system, IEEE Trans. Syst. Man Cybernetics 23 (1993) 665–685.
[55] S. Savage, D. Wetherall, A. Karlin, and T. Anderson, "Practical network support for IP traceback", in ACM SIGCOMM, 2000.
[56] R.E. Schapire, The strength of weak learnability, Machine Learning 5 (2) (1990) 197–227.
[57] A.C. Snoeren, C. Partridge, L.A. Sanchez, C.E. Jones, F. Tchakountio, Sch"single-packet IP traceback", IEEE/ACM Transactions Networking 10 (6) (2002) 721–734.
[58] R. Sommer, V. Paxson, "Outside the closed world: on using machine learning for network intrusion detection", in: Proceedings of the Symposium on Security and Privacy, 2010.
[59] Stefan Seufert, Darragh O Brein, "Machine learning for automatic defense against distributed denial of service attacks", in: Proceedings of IEEE International Conference (ICC) 2007, pp. 1217–1222.
[60] M.Y. Su, G.-J. Yu, C.-Y. Lin, A real-time network intrusion detection system for large-scale attacks based on an incremental mining approach, Journal of Computers and Security 75 (2009) 301–309.
[61] Tcpdump tool. Available from <http://www.tcpdump.org/>.
[62] TCPtrace tool. Available from <http://www.tcptrace.org/>.
[63] A.N. Toosi, M. Kahani, A new approach to intrusion detection based on a evolutionary soft computing model using neuro-fuzzy classifiers, Journal of Computer Communications 30 (2007) 2201–2212.
[64] Trustwave, "Web Hacking Incident Database (WHID) Semiannual Report", January 2011.
[65] UNINA traffic traces. Available from <http://www.grid.unina.it/software/ITG>, February 2007.
[66] Yang Xiang, Wanlei Zhou, "Mark-aided distributed filtering by using neural networks for DDoS defense", in: IEEE GLOBECOM, 2005, pp. 1701–1705.
[67] Yang Xiang, Wanlei Zhou, Minyi. Guo, Flexible deterministic packet marking: an ip traceback system to find the real source of attacks, IEEE Transactions on Parallel and Distributed Systems 20 (5) (May 2009) 1–14.
[68] Yanxiang He, Qiang Cao, Yi Han, Libing Wu, Tao Liu, "Reduction of quality (RoQ) attacks on structured peer-to-peer networks", in: IEEE International Symposium on Parallel & Distributed Processing, 2009, http://dx.doi.org/10.1109/IPDPS.2009.5161178.
[69] S.Yu, T. Thapngam, J. Liu, S. Wei, W. Zhou, "Discriminating DDoS flows from flash crowds using information distance", in: Proceedings of the 3rd IEEE International Conference on Network and System, Security (NSS'09), pp. 18–21, October 2009.
[70] Yuichi Uchiyama, et al. "Detecting and tracing DDoS attacks in the traffic analysis using auto regressive model" IEICE Transactions on Information and Systems, E87-D(12) (2004) 2635–2643.
[71] Zhiqiang Gao, Nirwan Ansari, IP traceback from the practical perspective, IEEE Communications Magazine 43 (5) (May 2005) 123–131.
[72] Zhiqiang Gao, Nirwan Ansari, Differentiating malicious DDoS attack traffic from normal TCP flows with proactive tests, IEEE Communications Letters 10 (11) (Nov. 2006) 793–795.
[73] Zhiqiang Gao, Nirwan Ansari, "A practical and robust inter-domain marking scheme for IP traceback", Computer Networks, 51(3) (2007) 732–750. February 21.