Expert Systems with Applications 39 (2012) 129–141

Contents lists available at ScienceDirect

Expert Systems with Applications

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

Decision tree based light weight intrusion detection using a wrapper approach

Siva S. Sivatha Sindhu [a,*], S. Geetha [b,1], A. Kannan [a]

[a] Department of Computer Science and Engineering, Anna University, Chennai 600025, India
[b] Department of Information Technology, Thiagarajar College of Engineering, Madurai 625015, India

ARTICLE INFO

Keywords:
Intrusion Detection System
Misuse detection
Genetic algorithm
Neural network
Decision tree
Neurotree

ABSTRACT

The objective of this paper is to construct a lightweight Intrusion Detection System (IDS) aimed at detecting anomalies in networks. The crucial part of building lightweight IDS depends on preprocessing of network data, identifying important features and in the design of efficient learning algorithm that classify normal and anomalous patterns. Therefore in this work, the design of IDS is investigated from these three perspectives. The goals of this paper are (i) removing redundant instances that causes the learning algorithm to be unbiased (ii) identifying suitable subset of features by employing a wrapper based feature selection algorithm (iii) realizing proposed IDS with neurotree to achieve better detection accuracy. The lightweight IDS has been developed by using a wrapper based feature selection algorithm that maximizes the specificity and sensitivity of the IDS as well as by employing a neural ensemble decision tree iterative procedure to evolve optimal features. An extensive experimental evaluation of the proposed approach with a family of six decision tree classifiers namely Decision Stump, C4.5, Naive Baye's Tree, Random Forest, Random Tree and Representative Tree model to perform the detection of anomalous network pattern has been introduced.

1. Introduction

Conventional intrusion prevention strategies, such as firewalls, access control schemes or encryption methods, have failed to prove themselves to effusively protect networks and systems from increasingly sophisticated attacks and malwares. The Intrusion Detection Systems (IDS) turn out to be the proper salvage to this issue and have become a crucial component of any security infrastructure to detect these threats before they induce widespread damage.

The design and construction of IDS is subjected to many concerns including data collection, data pre-processing, intrusion recognition, reporting and response. Among these entities, intrusion recognition is highly indispensible. This component compares the audit data with the detection paradigms, which model the patterns of intrusive or innocuous behavior, so that both successful and unsuccessful intrusion attempts may well be identified and be contained.

Intelligent IDS is a dynamic defensive system that is capable of adapting to dynamically changing traffic pattern and is present throughout the network rather than only at its boundaries, thus helping to catch all types of attacks. The trivial factor that complicates such an IDS model construction is the demand for an automatically evolving system. The typical reasons that challenge this process are huge network traffic volumes, highly imbalanced data distribution, the difficulty to realize decision boundaries between normal and abnormal behavior, and a requirement for continuous adaptation to a constantly changing environment – of course the toughest factor.

Besides these, a very serious hazard that is imposed on the IDS is the ever growing audit database with patterns for each and every packet that passes the IDS, consequently, slowing up the capability of the IDS. More and more time is consumed as the database grows, and more and more false positives are generated, which defeats the purpose of IDS. This audit database is augmented with the real-time traffic data upon which the IDS get trained. Our observation has shown that this database contains irrelevant and redundant features and hence the IDS may be referred to as intense-IDS. This intensity impedes the training and testing process, consumes higher resource as well as offers poor detection rate. The optimum feature set needs to be identified by shredding the unnecessary features and also the candidate instance subset is to be identified for evaluation by deducing the redundant and noisy data patterns. The resulting audio log is populated with only the candidate instance subset and optimal feature subset, perhaps the very essential data points and hence the IDS may be referred as the lightweight IDS. The dimensionality of the reduced feature space (Mitra, Murthy, & Pal, 2002) is thus optimal and guarantees low false alarms and less computational cost. The proposed method

* Corresponding author. Mobile: +91 9626644151.
E-mail addresses: sivathasindhu@gmail.com (S.S. Sivatha Sindhu), sgeetha@tce.edu (S. Geetha), kannan@annauniv.edu (A. Kannan).
[1] Mobile: +91 9842550862.

0957-4174/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2011.06.013

employs wrapper based strategy where the feature selection process is wrapped inside the classifier. The resultant probability of the classifier's detection rate is used as feedback inputs to select the optimal features.

When confronted with these requirements, soft computing techniques have shown to offer promising solutions due to the high detection accuracy, fast processing times, ability to adapt and exhibit fault tolerance and especially resilience against noisy information. Humans are blessed with the characteristics of sensing that something is not right, detecting the anomalous patterns that differ from normal. It may be highly beneficial if this trait is injected into the machines, especially for the IDS model. Soft computing techniques impart the required artificial intelligence to IDS to make them self functioning as much as possible.

The proposed work is on developing advanced intelligent systems using ensemble soft computing techniques (Gaddam, Phoha Kiran, & Balagani, 2007) for intrusion detection. Integration of different soft computing techniques like neural network (NN), genetic algorithm (GA), and decision tree (DT) (Amor, Benferhat, & Elouedi, 2004; Benferhat & Tabia, 2005; Xiang & Lim, 2005) has lead to discovery of useful knowledge to detect and prevent intrusion on the basis of observed activity. Candidate instance subset is generated by removing the redundant and noisy records from the audit log. The GA component imparts the feature subset selection through a suitably framed fitness function. A neurotree paradigm which is a hybridization of neural network and decision tree is proposed for misuse recognition which can classify known and unknown pattern of attacks. The hybridization of different learning and adaptation techniques, overcome individual limitations and achieve synergetic effects for intrusion detection.

2. Related work

Shun and Malki (2008) presented a neural network-based IDS for detecting internet-based attacks on a computer network. Neural networks are used to identify and predict current and future attacks. Feed-forward neural network with the back propagation training algorithm was employed to detect intrusion. Sarasamma, Zhu, and Huff (2005) proposed a novel multilevel hierarchical Kohonen net to detect intrusion in network. Randomly selected data points form KDD Cup (1999) is used to train and test the classifier. The process of learning the behavior of a given program by using evolutionary neural network based on system-call audit data is proposed by Han and Cho (2006). The benefit of using evolutionary neural network is that it takes lesser amount of time to obtain better neural networks than when using conventional approaches. This is because they evolve the structures and weights of the neural networks simultaneously. They performed the experiment with the KDD intrusion detection evaluation data. Thomas and Balakrishnan (2009) addressed the problem of optimizing the performance of IDS using fusion of multiple sensors. The trade-off between the detection rate and false alarms highlighted that the performance of the detector is better when the fusion threshold is determined according to the Chebyshev inequality. A neural network supervised learner has been designed to determine the weights of individual IDS depending on their reliability in detecting a certain attack. The final stage of this data dependent fusion architecture is a sensor fusion unit which does the weighted aggregation in order to make an appropriate decision. The major limitation with this approach is it requires large computing power and no experimental results are available for their proposed approach. Linda, Vollmer, and Manic (2009) presented an IDS-NNM – Intrusion Detection System using Neural Network based Modelling for detection of anomalous activities. The major contributions of this approach are use and analyses of real network data obtained from an existing critical infrastructure, the development of a specific window based feature mining technique (Fayyad & Uthurusamy, 2002), construction of training dataset using randomly generated intrusion vectors and the use of a combination of two neural network learning algorithms namely the Error-Back Propagation and Levenberg–Marquardt, for normal behavior modelling. Koutsoutos, Christou, and Efremidis (2007) present a neural network classifier ensemble system using a combination of neural networks which is capable of detecting network attacks on web servers. The system can identify unseen attacks and categorize them. The performance of the neural network in detecting attacks from audit dataset is fair with success rates of more than 78% in detecting novel attacks and suffers from high false alarms rates. An ensemble combining the conventional neural network with a second module that monitors the server's system calls results in good prediction accuracy.

Comprehensibility, i.e., the explain-ability of learned knowledge is vital in terms of usage in reliable applications like IDS (Joo, Hong, & Han, 2003). The existing NN based IDS discussed in the literature lack comprehensibility and this is incorporated by means of extended C4.5 decision tree. Also a variation in activation function is proposed in order to reduce the error rate thus increasing the detection performance.

Prema Rajeswari and Kannan (2008) discusses a rule based approach using enhanced C4.5 algorithm for intrusion detection in order to detect abnormal behaviors of internal attackers through classification and decision making in networks. The enhanced C4.5 algorithm derives a set of classification rules from KDD data set and then the generated rules are used to detect network intrusions in a real-time environment. An intrusion detection based on the AdaBoost algorithm is proposed by (Weiming Hu, Wei Hu, & Maybank, 2008). In this algorithm, decision stumps are used as weak classifiers and decision rules are provided for both categorical and continuous features. They combined the weak classifiers for continuous attributes and categorical attributes into a strong classifier. The main advantage of this approach is that relations between these two different types of features are handled naturally, without any type conversions between continuous and categorical attributes. Additionally they proposed a strategy for avoiding overfitting to improve the performance of the algorithm. Yasami and Mozaffari (2009) presents a host based IDS using combinatorial of *K*-Means clustering and ID3 decision tree learning algorithms for unsupervised classification of abnormal and normal activities in computer network. The *K*-Means clustering algorithm is first applied to the normal training data and it is partitioned into K clusters using Euclidean distance measure. Decision tree is constructed on each cluster using ID3 algorithm. Anomaly scores value from the *K*-Means clustering algorithm and decisions rules from ID3 are extracted. Resultant anomaly score value is obtained using a special algorithm which combines the output of the two algorithms. The threshold rule is applied for making the decision on the test instance normality. Performance of the combinatorial approach is compared with individual *K*-Means clustering, ID3 classification algorithm and the other approaches based on Markovian chains and stochastic learning automata.

Unlike existing decision tree based IDS discussed above the generated rules fired in this work are more efficient in classification of known and unknown patterns because the proposed neurotree detection paradigm incorporates neural network to preprocess the data in order to increase the generalization ability. But the existing decision tree based approaches discussed in the literature lack generalization and so the ability to classify unseen pattern is reduced.

3. Design of proposed system

The proposed system has four phases

Phase I - Preprocessing of network traffic pattern (removal of redundant data)
Phase II - Feature extraction (GA)
Phase III - Post-processing (normalization)
Phase IV - Classification of traffic patterns (neurotree)

3.1. Preprocessing of network traffic pattern

The major weakness with KDD data set is the presence of redundant records. The occurrence of redundant instances causes the learning algorithm to be biased towards frequent records and unbiased towards infrequent records. As the percentage of records for R2L class is very less in original KDD dataset the learning algorithm is unbiased towards R2L records due to the redundant and enormous records present in class like DoS. These redundant records are removed in order to improve the detection accuracy.

3.2. Rationale for the choice of GA in feature extraction

3.2.1. Individual's encoding

In order for a GA (Stein, Chen, Wu, & Hua, 2005) to efficiently search optimal features from such large spaces, careful attention has to be given to both the encoding chosen and the fitness function. In this work, there is a natural encoding of the space of all possible subsets of a feature set, namely, a fixed-length binary string encoding in which the value of the ith gene {0, 1} indicates whether or not the ith feature (i = 1 to 41) from the overall feature set is included in the specified feature subset. Thus, each individual chromosome in a GA population consists of fixed length, i.e., 41-bit binary string representing some subset of the given feature set. The advantage of this encoding is that a standard representation and a well understood GA can be used without any modification.

3.2.2. Fitness function

Each member of the current GA population represents a competing feature subset that must be evaluated to provide fitness feedback to the neurotree. This is achieved by invoking neurotree with the specified feature subset and a set of training data (which is condensed to include only the feature values of the specified feature subset). The neurotree produced is then tested for detection accuracy on a set of unseen evaluation data. We aim to enhance the detection accuracy of the IDS which is indirectly achieved by maximizing the sensitivity and specificity of the classifier. Hence, this knowledge is imparted into the IDS through the fitness function of the GA module. As a result the fitness function is formulated as

$$\text{Fitness} = \alpha * (1/\text{Count of Ones}) + \beta * \text{Sensitivity} + \gamma * \text{Specificity} \quad (1)$$

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (2)$$

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}} \quad (3)$$

Fitness of a chromosome is evaluated based upon the sensitivity and specificity from the validation dataset and number of features present in a chromosome. Here TP and TN are the number of records correctly classified in normal and abnormal classes respectively. Similarly FP and FN are the number of records incorrectly classified in normal and abnormal classes respectively. Count of ones is the number of ones present in the chromosome. If two subsets attain the same performance, while having different number of features, the subset with fewer features have been chosen. Among specificity, sensitivity and number of features, number of features is of less concern, so more weightage to specificity (γ = 0.4) and sensitivity (β = 0.4) is given than number of features (α = 0.2) to be selected.

3.2.3. Genetic operators

The other genetic operators like crossover, mutation and selection applied are that of the general simple GA's i.e., uniform crossover, simple mutation and tournament selection.

3.2.4. Algorithm for feature selection

GA_Feature_Selection(){
Input: Encoded binary string of length n (where n is the number of features being passed), number of generations, population size, crossover probability (Pc), mutation probability (Pm).
Output: A set of selected features that maximize the sensitivity and specificity of IDS.
1. Initialize the population randomly with the size of each chromosome to be 41.
Each gene value in the chromosome can be '0' or '1'. A bit value of '0' represent the corresponding feature is not present in chromosome and '1' represent the feature is present.
2. Initialize α = 0.2, β = 0.4, γ = 0.4, N (total number of records in the training set), Pc and Pm.
3. for each chromosome in the new population{
a. Apply uniform crossover and mutation operator to the chromosome with the specified probability Pc and Pm.
b. Evaluate fitness = $\alpha * (1/\text{Count of Ones}) + \beta * \text{Sensitivity} + \gamma * \text{Specificity}$}
4. If (Current_fitness − Previous_fitness < 0.0001) then Quit
5. Select the top best 50% of chromosomes into new population using tournament selection.
6. If number of generations is not reached, go to line 3.}

The above is an abstracted description of the algorithm execution. As a whole, the execution of the combined GA and neurotree algorithm is a wrapper approach. Each iteration results in a decision tree. After n iterations, a series of trees will be obtained, the best of which could be used to generate rules. The tree with the highest sensitivity and specificity is identified to be the best tree. Thus best set of features are extracted (Benferhat & Tabia, 2005; Liu & Yu, 2005) based on sensitivity and specificity values. The performance of the IDS is compared with a family of decision tree classifiers namely Decision Stump, C4.5, Naive Baye's Tree, Random Forest, Random Tree and Representative Tree model.

3.3. Post-processing of resulting feature vector

In the proposed wrapper approach based IDS the post-processing of the resulting feature vectors is responsible for preparing the following analysis by providing normalization as well as format conversions on the feature vectors. This step is introduced to make the approach more flexible and allow for different analysis or classification approaches. Before proceeding to evaluate the performance of the classifier, the discrimination capability of the proposed features is to be analyzed. The experiment involves formation of different dataset with different number of class types and with various feature vectors generated by different feature extraction algorithm ranging from best first to rank search. Hence

the feature set formed has to be normalized before feeding them into the classifier for training, to provide a uniform semantics to the feature values. A set of normalized feature vectors as per the data smoothing function given in Eq. (4)

$$x_{New} = \frac{x_{Current} - MIN}{MAX - MIN} \tag{4}$$

where, x is the numerical value of the attribute, *MIN* is the minimum value for the attribute that x belongs to, and *MAX* is the maximum values for the attribute that x belongs to.

3.4. Rationale for the choice of neurotree as classifier

The learning steps of a neurotree detection methodology are as follows. First, a network structure is defined with a fixed number of inputs, hidden nodes and outputs. Second, an algorithm neurotree is chosen to realize the learning process. This algorithm uses bagging approach which generates multiple training datasets from the original KDD dataset and then trains multiple neural networks (back propagation) named neural network ensemble. The predicted results produced by these neural networks are combined based on the voting algorithm. Then, the trained NN ensemble is employed to generate a new training set through replacing the desired class labels of the original training examples, with those output from the trained NN ensemble. Some extra training examples are also generated from the trained NN ensemble and added to the new training set. Finally, an enhanced C4.5 decision tree is grown from the new training set. The fusion of improved NN and enhanced C4.5 provides higher detection rates on unseen samples than the system only applies C4.5 to IDS.

3.4.1. Neural network ensemble

Neural networks have been widely used in anomaly intrusion detection as well as in misuse intrusion detection. There are two approaches for implementing neural network for misuse detection (Cannady, 1998). The first approach incorporates the neural network module into the existing or modified expert system. This approach uses the neural network to preprocess the incoming data for suspicious activities and forwards them to the expert system. This enhances the efficiency of the detection system. The second approach employs the neural network as a stand alone system to detect intrusion. In this method, the neural network process data from the network audit logs and analyzes it for intrusion detection. The proposed detection system employs the earlier approach and it has three steps. In the first step it collects data randomly from the KDD dataset for each NN and constructs a training dataset using bagging approach. Bagging and boosting (Bauer & Kohavi, 1998; Zhou & Jiang, 2004) are the two widely applied techniques for combining multiple classifiers in order to improve the prediction accuracy. These techniques aggregate multiple hypotheses produced by the same learning algorithm (neural network) invoked over different distributions of KDD dataset. They generate a classifier with a smaller error on the dataset as it combines multiple hypotheses which individually have a huge error. Bagging (Bauer & Kohavi, 1998) refers to ***B***ootstrap ***AGG***regat***ING.*** In bagging if a single classifier is unstable i.e., it has high variance, the aggregated classifier (Zhou & Jiang, 2004) (neural ensemble) has a smaller variance than a single base classifier. Boosting generates a series of neural networks whose training data sets are determined by the performance of the previous networks. Training instances that are wrongly predicted by the previous networks will play major roles in the training of the later networks. This results in high error rate if the initial neural network trained produces false prediction results. Also bagging is more suitable if the induced classifier is unstable. As neural network and decision tree are unstable classifiers bagging approach is employed in the proposed work. Bagging creates a training dataset by sampling with replacement 'n' times from the dataset containing 'n' records. The created dataset have some duplicated records and some of the records are not selected from original dataset. These unpicked records are placed in the testing set. As the KDD data set have large number of records, about 36.8% of the original records is placed in the testing set.

Proof

Consider there are 'n' records in the dataset then the probability of particular record being picked is $1/n$. Therefore the probability of record not being picked is $1 - 1/n$. As these records are picked 'n' times then the chance that particular record not being picked is $(1 - 1/n)^n \approx e^{-1} = 0.368$ for $n > 20$, where e is the base of natural logarithms (2.7183). From this it can be concluded that around 36.8% of the original unique records are placed in the testing set and about 63.2% of unique records are placed in the training set. Some of the records are repeated in training set and the total size of the generated training set is same as that of the original dataset.

In the second step, each of the neural networks is trained using training dataset generated to identify the network pattern based on feature vector. Some extra training examples are also generated from the trained NN ensemble and added to the training set.

In the final stage the trained neural network ensemble is employed to generate a new training set through replacing the desired class labels of the original training examples, with those output from the trained NN ensemble. The network pattern is identified based on the predicted output from each of the NN using voting algorithm. Voting algorithm chooses the class label receiving most number of votes as the final output of the ensemble.

3.4.2. Analysis of error rate of neural network

During the learning phase the weights in the NN is modified. In conventional NN, all weights are modified based on the error value. The error value is calculated by calculating the difference between the actual output and the predicted output. The error value is then propagated back from the top layer to lower layers to be used at these layers to modify connection weights. In this work, improved NN is employed in order to reduce the error rate (i.e., false positive and false negative error) of IDS, so as to increase its performance.

To achieve this weight values are adjusted based on the ratio of false positive and false negative error. The main aim of this analysis is to find the relationship between individual error rates and to find optimal weight values for w_1 and w_2 so as to reduce the total error. False negative error occurs when a normal pattern is classified as an attack type whereas false positive error occurs when an attacker is recognized as a legitimate person and allowed to enter the organization network and access the available assets. This may cause greater damage to the organization as the attacker who is recognized as a legitimate user could destroy various resources and may cause various security incidents. Therefore reducing false positive error is very essential when compared to false negative error. Thus false positive error is given higher weightage when compared to false negative error. Here normal class is considered as a positive class and anomaly classes are considered as negative class.

$$\text{Error rate} = w_1 * \text{false positive rate} + w_2 * \text{false negative rate} \tag{5}$$

where w_1 and w_2 are weight values for false positive and false negative error rate respectively. Best results are obtained when $w_1 = 0.8$ and $w_2 = 0.2$.

3.4.3. Extended C4.5

Decision tree induction algorithms have been applied in various fields. Some of the decision tree algorithms are ID3, C4.5 (Quinlan, 1986; 1993) and C5.0. C4.5 is an extension of the basic ID3 algorithm. The proposed system utilizes enhanced C4.5 which is

an improvement over C4.5. The ID3 and C4.5 algorithm utilizes the information theoretic approach in classifying a network traffic pattern. The decision tree is initially created from the pre-classified dataset. Each instance is defined by values of the attributes. A decision tree consists of nodes, edges and leaves. A node of a decision tree identifies an attribute by which the instance is to be partitioned. Every node has a number of edges, which are labeled according to a potential value of edges and a probable value of the attribute in the parent node. An edge links either two of the nodes in a tree or a node and a leaf. Leaves are labeled with class labels for classification of the instance. Information gain is calculated for each of the attribute. The best attribute to divide the subset at each stage is selected using the information gain of the attributes. According to the values of these attributes the instances are divided. If the value of attributes is nominal then a branch for each value of the attribute is formed, but if it is numeric a threshold value is determined and two branches are created. This procedure is recursively applied to each partitioned subset of the instances. The procedure ceases when all the instances in the current subset belong to the same class. The concept of information gain tends to favor attributes that have a large number of values. For example, if there are set of records T and an attribute X that has a distinct value for each record, then Info(X,T) is 0, thus Gain(X,T) is maximal. To overcome this extended C4.5 algorithm is used which employs gain ratio instead of information gain which takes into account the potential information from the partition itself. Extended C4.5 deals with continuous attributes and missing attributes which helps in improving the computation efficiency. To categorize an unknown instance, one starts at the root of the decision tree and follows the branch indicated by the result of each test until a leaf node is arrived. The name of the class at the leaf node is the resulting classification.

4. Research framework

4.1. Functional framework of wrapper based IDS

In the proposed wrapper based approach, after neurotree construction in phase I, the analysis of specificity and sensitivity is presented in phase II as in Fig. 1. The purpose of this phase is to analyze the relationship between the detection rate and error rate, and find the optimal features which minimize the false alarm rates for IDS. This objective is incorporated into the model via the fitness function of the GA component that is employed for feature selection. The selected features which are significant are then used for restructuring and improving the neurotree model. The improved model is used for real time detection as in phase 2. The performance of the IDS is optimized when the total errors are minimized. Hence the objective of the fitness function is to minimize the false alarm rates. In order to prevent over-fitting and to give more exploration to the system, our proposed fitness evaluation framework considers the problem as a three-goal objective function: maximize the sensitivity, and maximize the specificity and minimize the number of features.

4.2. Proposed neurotree algorithm

Algorithm: Neurotree algorithm
Input: Network audit data from MIT Lincoln Labs
T = {(x1, y1), (x2, y2), ... , (xn, yn)}, extra data ratio μ, trials of bootstrap sampling B, number of records in the training set 'n'
Output: Decision tree C4.5 DT

1. Train the neural network (NN) from T via Bagging. Call the procedure NN* = *Bagging* (T, NN, B)
2. Initialize generated training set $T' = \phi$
3. Process the original training set with the trained NN* and classify an instance x_i by counting votes for which NN*(x_i) represents the class with most votes
   For (i = 1 to n)
   Begin
   a. Replace the class label (y_i) with those output from the neural ensemble $y'_i = \mathrm{NN}^*(x_i : (x_i, y_i) \in T)$
   b. Add the new samples generated to the generated training set $T' = T' \cup \{(x_i, y'_i)\}$
End
4. Generate extra training data from the trained NN*
   For (j = 1 to $\mu \times n$)
   Begin
   a. Randomly generate attribute vector x'_j = Random() and feed it into NN* and add the outcome y'_j to the attribute vector as the last label. $y'_j = \mathrm{NN}^*(x'_j)$
   b. Append the attribute vector (x'_j, y'_j) to the generated training set $T' = T' \cup \{(x - j', y'_j)\}$
   End
5. Read records from the neural ensemble (NN*) generated training set T'
6. Tokenize each record and store it in an array
7. Determine whether the attribute is discrete or continuous
8. If (discrete attribute)
   a. Find the probability of occurrence for each value for each class
   b. Find the entropy I(P) = −(p1 ∗ log (p1) + p2 ∗ log (p2) + · · · + pn ∗ log(pn))
   c. Calculate the information gain *Gain* (X, T') = *Info* (T') − *Info* (X, T')
   d. Compute $GainRatio(X, T') = \frac{Gain(X,T')}{SplitInfo(X,T')}$
   where *SplitInfo*(X, T') is the information due to the split of T' on the basis of the value of the categorical attribute X. Thus *SplitInfo*(X, T') is $I\left(\frac{|T'_1|}{|T'|}, \frac{|T'_2|}{|T'|}, \ldots, \frac{|T'_m|}{|T'|}\right)$ where $(T'_1, T'_2, \ldots T'_m)$ is the partition of T' induced by the value of attribute X
   Else
   a. For continuous attributes the values are sorted and the GainRatio for each partition is found
   b. The partition with the highest GainRatio is considered
9. Construct the extended C4.5 DT with the highest GainRatio attribute as the root node and values of the attribute as the arc labels
10. Repeat steps 7 to 9 until categorical attributes or the leaf nodes are reached
11. Derive rules following each individual path from root to leaf in the tree
12. The *condition* part of the rules is built from the label of the nodes and the labels of the arcs: the *action* part will be the classification (eg. Normal, Smurf etc)
Procedure *Bagging* (T, NN, B)
Begin
for (i = 1 to B)
{
   a. Generate new training set of size 'n' with replacements for each 'B' trials, T_i = bootstrap sample from T;

(*continued on next page*)

b. Generate a classifier NN_i for each training set. Call procedure *NeuralNetwork*(T_i)
}
Form ensemble neural network classifier NN^* by aggregating the '*B*' classifiers
Return trained neural ensemble NN^*
End

Procedure *NeuralNetwork* (TrainingSet T_i)
Begin
1. Get input file T_i for training
2. Read records from T_i
3. Train the network by specifying the number of input nodes, hidden nodes, output nodes, learning rate and momentum
4. Initialize weights and bias to random values
5. Calculate output for each node
Node input = $\sum$(weight+output of previous layer cells) + Bias value of nodes
Node output = $1/(1 + \exp(-(\text{Node input})))$
Repeat until final output node is reached
/*Back propagating the errors*/
6. Calculate Error rate (ER) = E (FP, FN)
Therefore, $ER = (W_1^* FP + W_2 * FN)$ where FP is false positive rate, FN is false negative rate, W_1 and W_2 are their respective weight values.
7. Output cell error=Logistic function derivative * Error rate where logistic function derivative = $dF(x)/dx$
$=1/(1 - \exp(-x)) * (1 - (1/(1 - \exp(-x)))) *$ Error rate
8. Hidden Cell error = Logistic function derivative * Sum of(output layer cell error * weight of output layer cell connection)
/* Adjusting weights and bias*/
9. Net weight = Current Weight between hidden layer and output + (output cell error * hidden layer cell value * learning rate)
10. Net Bias value = Current bias Value + (learning rate * output cell error)
11. Training is completed
12. Return trained neural network
End

4.3. Performance measurement indices

Performance of IDS is evaluated using the following indices

- Detection rate (DR) - Ratio between number of anomaly (normal) correctly classified and total number of anomaly (normal).
- Error rate (ER) - Ratio between number of anomaly (normal) incorrectly classified and total number of anomaly (normal).
- True positive (TP) - Classifying normal class as normal class.
- True negative (TN) - Classifying anomaly class as anomaly class.
- False positive (FP) - Classifying normal class as an anomaly class.
- False negative (FN) - Classifying anomaly class as a normal class.

These are good indices of performance, as they measure what percentage of intrusions the system is able to detect and how many incorrect classifications are made in the course of action.

Other metrics include

- Kappa statistics- Used in assessing the degree to which two or more raters, examining the same data, agree when it comes to assigning the data to categories.
- Mean Absolute Error (MAE) - Average over the verification sample of the absolute values of the differences between forecast and the corresponding observation.
- Root Mean Squared Error (RMSE) - Quadratic scoring rule which measures the average magnitude of the error. Measures the difference between forecast and corresponding observed values, are each squared and then averaged over the sample. Finally, the square root of the average is taken.
- Relative Absolute Error (RAE) - Similar to the relative squared error in the sense that it is also relative to a simple predictor, which is just the average of the actual values.
- Relative Squared Error (RSE) - Calculates the total squared error and normalizes it by dividing the total squared error of the simple predictor.

5. Test scenario

The proposed model is developed using Java and the neurotree algorithm proposed is implemented as per the framework. The stepwise procedure is as follows. The input to the system is given as an attribute-relation file format (ARFF) file. The dataset is created using the name specified in \@relation". The attributes are specified under \@attribute" correspondingly specifying the type of attribute and instances specified under \@data" are retrieved from the ARFF file and then they are used for training by the classifier. This procedure is followed for test set also. The classifier was trained and evaluated by using the preprocessed dataset, formed from the whole KDD dataset (Tavallaee, Bagheri, Lu, & Ghorbani, 2009).

5.1. Test objectives

Objective #1: To investigate the impact of proposed feature extraction algorithm on the performance of IDS.
Objective #2: To identify the set of misuse sensitive features and to find out the discriminative power of selected features.
Objective #3: To evaluate the impact of proposed neurotree classification algorithm on the proposed framework.
Objective #4: To investigate the detection capability of neurotree with dataset containing 23 partitions namely normal and specific attack type classes such as Neptune, Back, Smurf, Buffer_overflow etc.
Objective#5: To investigate the impact of training and test dataset on detection accuracy.
Objective#6: To investigate the impact of proposed neurotree algorithm with various rule base approaches.

5.2. Preparation of test dataset

Regardless of the detection paradigm used, it is also vital to use relevant and essential data in order to build and verify network Intrusion Detection Systems. Unfortunately, for various reasons, IDS is a field with a lack of good quality data. Therefore we rely on the data set compiled for the 1999 KDD intrusion detection contest, by Massachusetts Institute of Technology's (MIT) Lincoln Labs (KDD Cup, 1999). The main reason for using this data set is that we need relevant data that can be easily shared with other researchers and developers, allowing them to duplicate and improve our results. It is considered as a standard benchmark

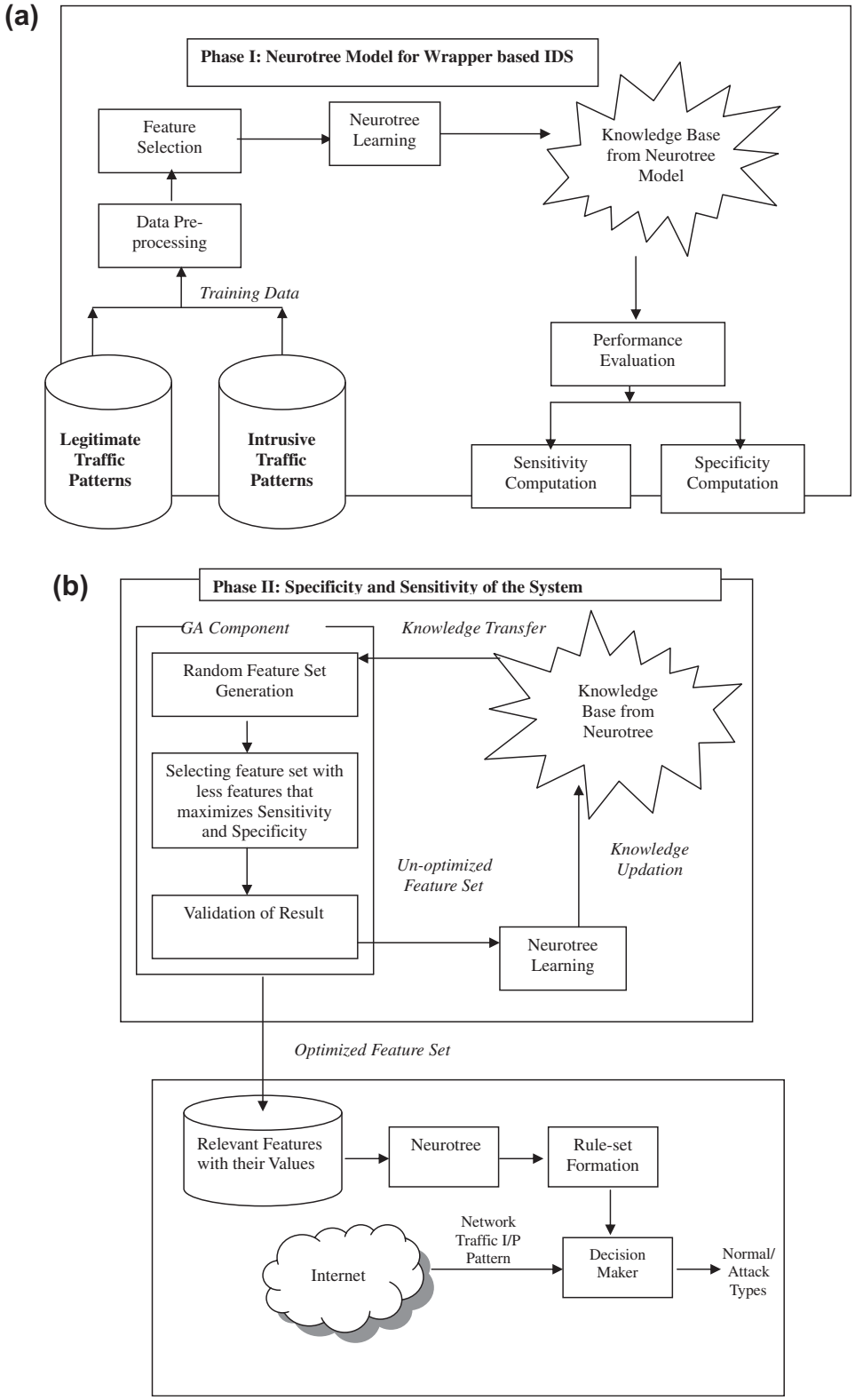

Fig. 1. (a). Framework for the Phase 1 of the proposed wrapper based IDS. (b). Framework for the Phase 2 of the proposed wrapper based IDS.

for intrusion detection evaluations. In this dataset, forty-one attributes that usually characterize network traffic behavior compose each record. The record pattern may be normal or of attack type. There are totally 22 different types of attacks reported and these attacks fall into four main categories. They are DoS, U2R, R2L and Probe.

5.2.1. Influence of feature vector

In this experiment, the traffic patterns of various feature vectors are selected to see the influence on detectability. Initially, to extract relevant features using proposed method, dataset is formed using 41 attribute of feature vector and 22 different attack types. These relevant features extracted are written into a file with their

Table 1
Distribution of records in training and test dataset before and after redundancy removal.

Class	Training set		Test set			
	Total no. of records in KDD Cup 99	No. of unique records after redundancy removal	Total no. of records in KDD Cup 99		No. of unique records after redundancy removal	
			Known	Unknown	Known	Unknown
Normal	972781	812814	60593	-	47911	-
DoS	3883370	45927	223298	6555	5741	1717
Probe	41102	11656	2377	1789	1106	1315
R2L	1126	995	5993	10196	2199	555
U2R	52	52	39	189	37	163

corresponding data values for each and every class. This is done for the below mentioned feature extraction algorithm in section.4 (used for comparison purpose) and the proposed algorithm. The result for proposed feature extraction is good when compared to other algorithms. This satisfies objective#1 and objective#2.

5.2.2. Influence of neurotree classifier

To test the performance of the proposed method, our training and test dataset consists of network traffic patterns from 10% of KDD dataset (KDD Cup, 1999). It formed a network traffic database containing diverse data with 22 different attack types and normal class with selected feature vector to satisfy objective#4. Cross-validation is applied to the dataset formed for estimating the generalization error based on re-sampling and to estimate how accurately the intrusion detection paradigm performs in real time.

Experiments are conducted using stratified 10-fold cross-validation. In stratified10-fold cross-validation the sample of data instances are split into 10 approximately equal partitions such that the mean response value is approximately equal in all the partitions i.e., each partition contains roughly the same proportions of the all types of class labels present in the original dataset. After partitioning 9/10 of dataset is used for training i.e., 9 partitions are used for performing analysis called training set and 1/10 dataset is used for testing called validation set or testing set. This procedure is repeated 10 times and the overall error rate is calculated by taking average of error rates on each partition i.e., the final output is the average result of these ten folds. After ten-fold cross-validation we found that TP rate and FP rate of classes like imap, phf, perl, spy, and multihop are nil as the number of records in these class types are less than 10 and therefore these records are not present in most of the partition. This satisfies objective#3.

5.2.3. Detection with mismatch between training and test dataset

Here we evaluate the performance by using different types of attacks in training and testing. Denote the training set and the test set as *TR* and *TE* respectively. First, we created *TR* by including anomaly records of 22 attack types and some normal traffic patterns. On the other hand, *TE* is constituted of anomaly traffic patterns of 22 attack types and normal patterns that are not present in the training dataset. Essentially, *TR* and *TE* were made disjoint. We also evaluated the detection performances in presence of other mismatches. In the second case, we interchanged the roles of *TR* and *TE* to form another two sets. The experimental results are listed in Table 4, which reveals that the average classification rate is 98.38%.

5.3. Preprocessing – redundancy check

In this analysis, redundant records present are replaced by a single copy. After redundancy check it is found that most of the redundant records are present in the anomaly class than normal class. About 75% redundant records are present in both training and test dataset. Some invalid records are found in the original KDD and it is removed. This process assists the neurotree learner from biasing towards frequent records.

Tables 1 and 2 shows the number of records present in the KDD Cup 99 dataset and the number of records obtained after redundancy removal for both training and test dataset. It is inferred that

Table 2
Distribution of Records for specific class in Training and Test Dataset before and after Redundancy Removal.

S.No	Specific class types	Class	Total no. of samples		Unique samples	
			Train	Test	Train	Test
1	Normal	Normal	972781	60593	812814	47911
2	Smurf	DoS	2807886	164091	2646	665
	Neptune	DoS	1072017	58001	41214	4657
	Back	DoS	2203	1098	956	359
	Teardrop	DoS	979	12	892	12
	Pod	DoS	264	87	201	41
	Land	DoS	21	9	18	7
3	Satan	Probe	15892	1633	3633	735
	Ipsweep	Probe	12481	306	3599	141
	Portsweep	Probe	10413	354	2931	157
	Nmap	Probe	2316	84	1493	73
4	Warezclient	R2L	1020	0	890	0
	Guess_passwd	R2L	53	4367	53	1231
	Warezmaster	R2L	20	1602	20	944
	Imap	R2L	12	1	11	1
	Ftp_write	R2L	8	3	8	3
	Multihop	R2L	7	18	7	18
	Phf	R2L	4	2	4	2
	Spy	R2L	2	0	2	0
5	Buffer_overflow	U2R	30	22	30	20
	Rootkit	U2R	10	13	10	13
	Loadmodule	U2R	9	2	9	2
	Perl	U2R	3	2	3	2
	Total		4898431	292300	871444	56994

Table 3
List of features selected by various feature selection algorithms.

S.No	Algorithm	#Features selected	Features selected	Detection rate
1	BestFirst + ConsistencySubsetEval	11	duration, service, src_bytes, dst_bytes, count, srv_count, dst_host_srv_count, dst_host_diff_srv_rate, dst_host_same_src_port_rate, dst_host_srv_diff_host_rate, dst_host_serror_rate	97.01
2	GeneticSearch + CfsSubset Eval	20	duration, protocol_type, service flag, src_bytes, dst_bytes, wrong_fragment, num_failed_logins, logged_in, count, srv_serror_rate srv_rerror_rate, same_srv_rate,diff_srv_rate, dst_host_count, dst_host_srv_count, dst_host_diff_srv_rate, dst_host_srv_diff_host_rate, dst_host_serror_rate, dst_host_srv_serror_rate	98.16
3	GeneticSearch + ConsistencySubsetEval	20	duration, service, src_bytes, wrong_fragment, hot, num_failed_logins, root_shell, num_root, is_host_login, srv_count, serror_rate srv_serror_rate, srv_rerror_rate, same_srv_rate, srv_diff_host_rate, dst_host_srv_count, dst_host_same_src_port_rate, dst_host_srv_diff_host_rate, dst_host_rerror_rate, dst_host_srv_rerror_rate	97.86
4	GreedyStepwise + CfsSubsetEval	9	flag, src_bytes, wrong_fragment, hot, num_access_files, diff_srv_rate, dst_host_diff_srv_rate, dst_host_same_src_port_rate, dst_host_srv_diff_host_rate	97.97
5	Ranker + ChiSquaredAttributeEval	36	src_bytes, dst_bytes, service, wrong_fragment, flag, dst_host_srv_diff_host_rate, dst_host_diff_srv_rate, count, diff_srv_rate, srv_serror_rate, hot,dst_host_serror_rate, srv_count,same_srv_rate serror_rate, dst_host_srv_count, dst_host_same_srv_rate dst_host_same_src_port_rate, dst_host_srv_serror_rate, rerror_rate protocol_type, dst_host_rerror_rate, dst_host_count num_compromised, dst_host_srv_rerror_rate, logged_in, srv_rerror_rate, land, srv_diff_host_rate, duration, num_failed_logins, root_shell, is_guest_login, num_file_creations, num_access_files, num_root	98.13
6	RankSearch + CfsSubsetEval	22	protocol_type,service,flag,src_bytes,dst_bytes,land,wrong_fragment, hot, num_failed_logins, logged_in, num_compromised, serror_rate, srv_serror_rate, rerror_rate, same_srv_rate, diff_srv_rate,dst_host_same_srv_rate,dst_host_diff_srv_rate,dst_host_same_src_port_rate ,dst_host_srv_diff_host_rate,dst_host_serror_rate , dst_host_srv_serror_rate	98.4
7	RankSearch + ConsistencySubsetEval	34	duration, protocol_type, service, flag, src_bytes, dst_bytes,land,wrong_fragment, hot, num_failed_logins, logged_in, num_compromised, root_shell, num_file_creations, is_guest_login, count, srv_count, serror_rate, srv_serror_rate, rerror_rate, srv_rerror_rate, same_srv_rate, diff_srv_rate, srv_diff_host_rate, dst_host_count, dst_host_srv_count, dst_host_same_srv_rate, dst_host_diff_srv_rate, dst_host_same_src_port_rate,dst_host_srv_diff_host_rate, dst_host_serror_rate, dst_host_srv_serror_rate, dst_host_srv_rerror_rate	98.15
8	Proposed feature selection	16	protocol_type, service, flag, src_bytes, dst_bytes, wrong_fragment, hot, logged_in, srv_count, serror_rate, same_srv_rate, dst_host_diff_srv_rate, dst_host_same_src_port_rate, dst_host_srv_diff_host_rate, dst_host_srv_serror_rate, dst_host_rerror_rate	98.38
9	Nil	41	All features in KDD dataset	98.47

Table 4
Evaluation metrics comparison of NN*, extended C45 and neurotree.

Evaluation metrics	NN*	Extended C4.5	Neurotree
Kappa statistic	0.945	0.9432	0.9729
Mean absolute error	0.0029	0.0073	0.0018
Root mean squared error	0.0542	0.0515	0.0364
Relative absolute error	5.366%	13.313%	3.3432%
Root relative squared error	32.8434%	31.1735%	22.0407%
Correctly classified instances	96.76%	96.66%	98.38%
Incorrectly classified instances	3.23%	3.34%	1.61%

there is a large reduction in number of records for DoS attack when compared to other classes. Additionally, there are no duplicate records present in the training set for U2R attack.

5.4. Feature extraction – objective#1

A number of feature selection algorithms are proposed by various authors. The purpose of this work is to examine the various existing attribute selection algorithms in terms of detection accuracy and to compare those algorithms with the proposed algorithm. Out of the total 41 network traffic features, used in detecting intrusion, some features will be potential in revealing evidence of anomalies. Therefore the predominant features are extracted from the 41 features based on specificity and sensitivity metrics.

5.4.1. Attribute evaluators

Attribute evaluator is used for ranking all the features according to some metric. Various attribute evaluators available in WEKA (Weka, 2008) are used in this work which includes CfsSubsetEval, ChiSquaredAttributeEval, ConsistencySubsetEval, InfoGainAttributeEval and GainRatioAttributeEval.

- CfsSubsetEval: Evaluates the worth of a subset of attributes by considering the individual predictive ability of each feature along with the degree of redundancy between them. Subsets of features that are highly correlated with the class while having low intercorrelation are preferred.
- ChiSquaredAttributeEval: Evaluates the worth of an attribute by computing the value of the chi-squared statistic with respect to the class.
- ConsistencySubsetEval: Evaluates the worth of a subset of attributes by the level of consistency in the class values when the training instances are projected onto the subset of attributes.

- GainRatioAttributeEval: Evaluates the worth of an attribute by measuring the gain ratio with respect to the class.

GainR (Class, Attribute) = (H(Class) − H(Class|Attribute))/ H(Attribute).

- InfoGainAttributeEval: Evaluates the worth of an attribute by measuring the information gain with respect to the class.

InfoGain (Class, Attribute) = H(Class) − H(Class|Attribute).

5.4.2. Search methods

These methods search the set of all possible features in order to find the best set of features. Five search methods which includes BestFirst, GeneticSearch, GreedyStepwise, Ranker and RankSearch available in weak are used in this work for comparison purpose.

- BestFirst: Searches the space of attribute subsets by greedy hillclimbing augmented with a backtracking facility. Setting the number of consecutive non-improving nodes allowed controls the level of backtracking done. Best first may start with the empty set of attributes and search forward, or start with the full set of attributes and search backward, or start at any point and search in both directions (by considering all possible single attribute additions and deletions at a given point).
- GeneticSearch: Performs a search using the simple genetic algorithm.
- GreedyStepwise: Performs a greedy forward or backward search through the space of attribute subsets. May start with no/all attributes or from an arbitrary point in the space. Stops when the addition/deletion of any remaining attributes results in a decrease in evaluation. Can also produce a ranked list of attributes by traversing the space from one side to the other and recording the order that attributes are selected.
- Ranker: Ranks attributes by their individual evaluations. Use in conjunction with attribute evaluators (ReliefF, GainRatio, Entropy etc).
- RankSearch: Uses an attribute/subset evaluator to rank all attributes. If a subset evaluator is specified, then a forward selection search is used to generate a ranked list. From the ranked list of attributes, subsets of increasing size are evaluated, ie. The best attribute, the best attribute plus the next best attribute, etc.... The best attribute set is reported. RankSearch is linear in the number of attributes if a simple attribute evaluator is used such as GainRatioAttributeEval.

Various combination of features selection are tried and they include BestFirst + ConsistencySubsetEval, GeneticSearch + CfsSubsetEval, GeneticSearch + ConsistencySubsetEval, GreedyStepwise + CfsSubsetEval, Ranker + ChiSquaredAttributeEval, RankSearch + CfsSubsetEval, RankSearch + ConsistencySubsetEval, Ranker + InfoGainAttributeEval and Ranker + GainRatioAttributeEval gave the same result as that of Ranker + ChiSquaredAttributeEval. GreedyStepwise + ConsistencySubsetEval gave the same result as that of BestFirst + ConsistencySubsetEval .The details of the combinations and the features selected by each combination and their visualization is described Table 3, Figs. 2 and 3. The proposed algorithm performs the genetic search method to select the distinguishing features. The GA parameters are: Chromosome length = 41 (one gene per network traffic feature), Population Size = 100; Crossover Probability = 0.7 and Mutation Probability = 0.001. It has been found that the best set of features is selected within 1000 generations by the genetic algorithm. Finally, 16 salient features as in Table 3 are selected by the proposed genetic algorithm and the classification was based on these predominant features. Fig. 2 proves the findings of Breiman, Friedman, Olshen, and Stone

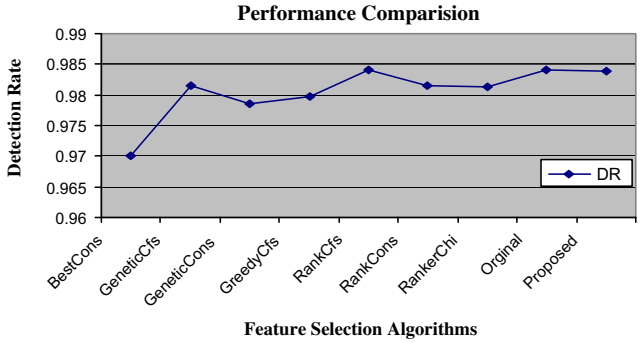

Fig. 2. Performancecomparisons of various feature extraction algorithms.

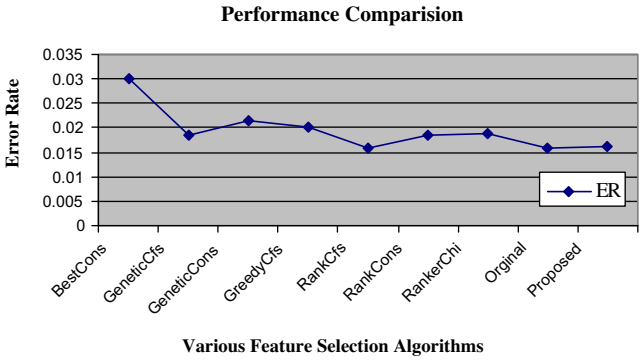

Fig. 3. Error rate for neurotree using various feature selection algorithms.

(1984) that detection accuracy is not much affected by the choice of attribute selection measure. Although the detection accuracy obtained by various algorithm like GeneticSearch + CfsSubsetEval, Ranker + ChiSquaredAttributeEval, RankSearch + CfsSubsetEval, RankSearch + ConsistencySubsetEval is near to the proposed algorithm, the number of features selected by this algorithm is less when compared to other algorithms. Thus the detection time of proposed feature selection algorithm is less when compared to other algorithms. However the number of features selected by BestFirst + ConsistencySubsetEval and GreedyStepwise + CfsSubsetEval is less when compared to other algorithms, their detection accuracy is reduced by approximately 2%.

5.5. Post-processing – objective#2

The discrimination ability of proposed feature vectors is analyzed to evaluate the performance of the neurotree classifier. For this the feature set formed has to be normalized before feeding into the classifier for training to achieve a uniform semantics to the feature values. A set of normalized feature vectors as per the data smoothing function. Therefore, the extracted features are normalized and written in two files. File#1 has records with five classes like DoS whereas File#2 has records formed with 23 different classes including normal. Additional, some more files are created with feature vectors extracted by other feature extraction algorithm.

5.6. Building and training classifier

The feed forward neural network is used in the proposed work due to its simple structure and easy realization. Back-propagation algorithm is used to train the network. The number of NN used for bagging is 10. Each NN consists of input layer with number of input nodes equal to the number of selected features and the output

layer based on number classes used. The output of each node is propagated from the input layer through the hidden layer to the output layer. The value of the output node ranges from 0 to 1. The output node with the highest value is chosen and its corresponding class is named as its output. Logistic activation function is adapted in hidden layer while linear activation function is employed in output layer. The maximum number of epochs for training NN is set as 1000. In order to avoid over-fitting, training of NN is stopped when the error value does not change in the consecutive five epochs. Extended C4.5 decision tree algorithm is constructed to classify the records. The input training records are given and the corresponding gain ratio for each of the attribute is calculated. The discrete and continuous attributes are identified from the input records. Then the tree is constructed based on the gain ratio. Following the each individual path in the tree, the rules are generated. The output of this module is decision tree. The portion of decision tree evolved is shown below

```
srv_count <= 299
| src_bytes <= 0
| | srv_count <= 5
| | | dst_host_srv_serror_rate <= 0.06
| | | | srv_count <= 1:0(16.0/1.0)
| | | | srv_count > 1:12(3.0/1.0)
| | | dst_host_srv_serror_rate > 0.06
| | | | dst_host_srv_serror_rate <= 0.83:9(3.0)
| | | | dst_host_srv_serror_rate > 0.83:7(5.0/1.0)
| | srv_count > 5
| | | dst_host_srv_serror_rate <= 0.4
| | | | dst_host_rerror_rate <= 0.92:6(19.0)
| | | | dst_host_rerror_rate > 0.92:2(35.0)
| | | dst_host_srv_serror_rate > 0.4:(1689.0)
| src_bytes > 0
| | src_bytes <= 8
| | | protocol_type <= 2
| | | | srv_count <= 3:0(5.0)
| | | | srv_count > 3:6(3.0)
| | | protocol_type > 2:8(20.0)
| | src_bytes > 8
| | | wrong_fragment <= 0
| | | | src_bytes <= 15876
| | | | | dst_host_rerror_rate <= 0.01
| | | | | | dst_host_srv_diff_host_rate <= 0.97:0(1515.0/11.0)
| | | | | | dst_host_srv_diff_host_rate > 0.97
```

6. Results and discussion

6.1. Evaluation of neurotree on proposed framework – objective#3

To demonstrate the increase in the detection performance, we compare the detection rate of the neurotree classifier with other classifiers like Extended C4.5 and neural network ensemble with modified activation function (NN*), which proves that our claim is valid. i.e., reduction of false alarm errors, increase the detection rates of the classifiers and reduce the false alarm rates. The performance comparison for the proposed system with Extended C4.5 and neural network ensemble with modified activation function (NN*) is presented in Table. 4. The detection and error rates obtained by various classifiers are shown in Figs. 4 and 5. Reduction of error is more important in IDS as mentioned in the previous section. It could be noted from Table 2 that the decrease in the mean absolute error, on an average, is 0.11% from 0.29% to 0.18%. Similarly, the root mean squared error has a drop by 1.78% from 5.42% to 3.64% on an average, relative absolute error has reduced by 2.0228% from 5.366% to 3.3432% and root relative squared error has reduced to 22.0407. We noted that there is a change in total error as the weight values w1 and w2 are changed. From experiment we found that false positive error was reduced considerably when w1 is assigned 0.8 and w2 is assigned 0.2. Thus the proposed activation function which is designed to reduce the error rates proves to be superior.

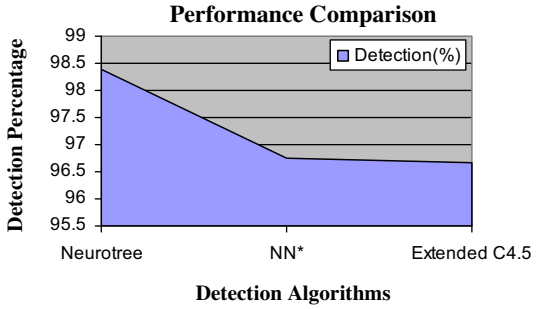

Fig. 4. Detection percentage comparison for neurotree, NN* and extended C4.5.

6.2. Investigation of detection capability with 23 classes – objective#4

In this experiment dataset is formed using 23 classes like smurf, back, normal etc and with 16 features selected using GA. In this testing phase the detection rate of imap, spy, perl etc is very low when compared to other classes. This is due to the presence very low number of traffic patterns for these classes. The overall detection rate is 98.4%. From this it can be concluded that neurotree detection paradigm performs better when specific attack types are provided. The detailed accuracy for each class is shown below.

TP Rate	FP Rate	Precision	Recall	F-Measure	Class
0.987	0.042	0.964	0.987	0.975	normal
0.318	0.001	0.778	0.318	0.452	warezclient
0.999	0.004	0.991	0.999	0.995	neptune
0.989	0.007	0.829	0.989	0.902	ipsweep
0.8	0.001	0.857	0.8	0.828	teardrop
0.911	0.001	0.953	0.911	0.932	satan
0.855	0.001	0.93	0.855	0.891	portsweep
0.983	0.001	0.95	0.983	0.966	smurf
0.8	0.001	0.933	0.8	0.862	nmap
0	0	0	0	0	warezmaster
0.074	0	0.667	0.074	0.133	back
0	0	0	0	0	land
1	0	1	1	1	pod
0.833	0	0.625	0.833	0.714	buffer_overflow
0	0	0	0	0	loadmodule
0	0	0	0	0	rootkit
1	0	0.833	1	0.909	guess_passwd
0	0	0	0	0	ftp_write
0	0	0	0	0	imap
0	0	0	0	0	spy
0	0	0	0	0	perl
0	0	0	0	0	multihop
0	0	0	0	0	phf

6.3. Evaluation of neurotree in detection performance – objective#6

To demonstrate the increase in the detection performance, we compare the detection rate of the proposed classifier with other six decision tree classifiers like Decision Stump, C4.5, Naive Baye's

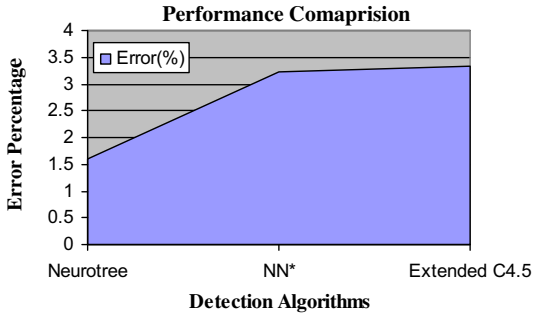

Fig. 5. Error percentage comparison for neurotree, NN* and extended C4.5.

Table 5
Performance evaluation of proposed classifier with six decision tree classifiers.

Classifiers	Detection percentage	Error percentage
Decision Stump	79.73	20.27
C4.5	92.1	7.9
Naïve Bayes	92.27	7.73
Random Forest	89.21	10.79
Random Tree	88.98	11.02
REP Tree	89.11	10.89
Proposed	98.38	1.62

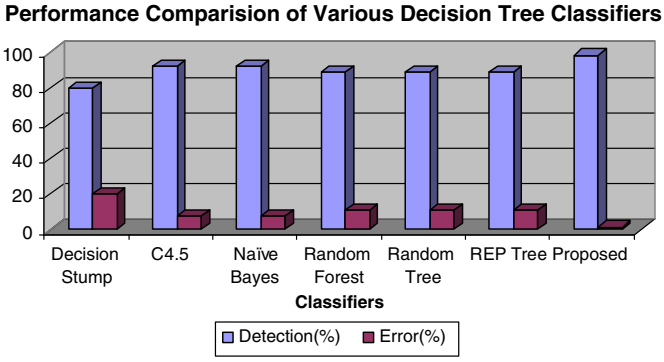

Fig. 6. Performance comparison of various decision tree classifiers with the proposed approach.

Tree, Random Forest, Random Tree and Representative (REP) tree model, which proves that our claim is valid. The performance comparison for the proposed system with six decision tree classifiers is presented in Table 5.

From Fig. 6, it has been inferred that detection rate of proposed approach is better in comparison with other approaches like Decision Stump, C4.5, Naïve Bayes, Random Forest, Random Tree and REP tree. This is due to the adapted fitness function which uses specificity and sensitivity as the metrics.

7. Discussion and conclusion

This work proposes lightweight IDS for multi-class categorization. The system is aimed at making improvements on existing work in three perspectives. Firstly, the input traffic pattern is preprocessed and redundant instances are removed. Next, a wrapper based feature selection algorithm is adapted which has a greater impact on minimizing the computational complexity of the classifier. Finally, a neurotree model is employed as the classification engine which imparted a detection rate of 98.4% which is superior to NN* and extended C4.5. Objective 4 summarizes the characteristics of our proposed method with various performance metrics like TP rate, FP rate, Precision, Recall and F-measure. It could be observed that the proposed system is better even when the dataset is presented with different number of classes. This justifies our claim that the proposed features and learning paradigm neurotree is a promising strategy to be applied on intrusion detection.

The main findings can be summarized as follows:

(1) The performance of the IDS is necessarily proportional to the training data, machine learning technique and the features selected to detect intrusion.
(2) The average classification rate (98.4%) for our proposed system is better to existing systems discussed in literature.
(3) The training and test database collects a larger number of normal and anomaly samples that were collected from KDD and are preprocessed to remove duplicate instances.
(4) The system is operated blindly and can detect specific attack type.
(5) The system is more suitable for multi-class classification problem which is a promising one.

References

Bauer, Eric, & Kohavi, Ron (1998). *An empirical comparison of voting classification algorithms: Bagging, boosting, and variants machine learning*. Kluwer Academic Publisher. 1–38.
Amor, Nahla B., Benferhat, S., Elouedi, Z. (2004). Naive Bayes vs decision trees in intrusion detection systems. In: Proceedings of the 2004 ACM symposium on Applied computing, *Cyprus*, pp. 420–424.
Benferhat, S. Tabia, K. (2005). On the combination of Naive Bayes and decision trees for intrusion detection. In: *IEEE International Conference on Computational Intelligence for Modelling, Control and Automation, 2005 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce*,1 (2006), pp. 211–216.
Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and regression trees*. Monterey, CA: Wadsworth.
J. Cannady. (1998). Artificial Neural Networks for Misuse Detection. In: *National Information Systems Security Conference*.
Fayyad, U. M., & Uthurusamy, R. (2002). Evolving data mining into solutions for insights. *Communications of the ACM, 45*, 28–31.
Gaddam, Shekhar R., Phoha Kiran, Vir V., & Balagani, S. (2007). *K*-Means+ID3: A novel method for supervised anomaly detection by cascading K-Means clustering and ID3 decision tree learning methods. *IEEE Transactions on Knowledge and Data Engineering, 19*, 3.
Han, S. J., & Cho, S. B. (2006). Evolutionary neural networks for anomaly detection based on the behavior of a program. *IEEE Transactions on Systems, Man and Cybernetics Part B: Cybernetics, 36*, 559–570.
Liu, H., & Yu, L. (2005). Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering, 17*, 491–502.
Joo, D., Hong, T., & Han, I. (2003). The neural network models for IDS based on the asymmetric costs of false negative errors and false positive errors. *Expert Systems with Applications, 25*, 69–75.
KDD Cup. (1999). Available on: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, 2007.
Koutsoutos, S., Ioannis T. Christou, Efremidis, S., A Classifier Ensemble Approach to Intrusion Detection for Network-Initiated Attacks, In: Proceeding of the international conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, *Information Retrieval and Pervasive Technologies*, 2007, pp.307–319.
Linda, O., Vollmer, T., Manic, M. (2009). Neural network based intrusion detection system for critical infrastructures. In: *Proceedings of IE EE international joint conference on Neural Networks, Georgia*, pp. 102–109.
Mitra, P., Murthy, C. A., & Pal, S. K. (2002). Unsupervised Feature Selection Using Feature Similarity. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 24*, 301–312.
Prema Rajeswari, L., & Kannan, A. (2008). An active rule approach for network intrusion detection with enhanced C4.5 Algorithm, I. *Journal of Communications, Network and System Sciences*, 285–385.
Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning, 1*, 81–106.
Quinlan, J. R. (1993). *C4.5: Programs for machine learning*. Springer.
Sarasamma, S., Zhu, Q., & Huff, J. (2005). Hierarchical Kohonen net for anomaly detection in network security. *IEEE Transactions on System, Man and Cybernetics, Part B: Cybernetics, 35*, 302–312.
Shun, J. Malki, H. A., Network Intrusion Detection System Using Neural Networks. In: *Proceedings of fourth IEEE International Conference on Natural Computation, ICNC'08, 2008*, pp. 242–246.

Stein, Gary, Chen, Bing, Wu, Annie S., & Hua, Kien A. (2005). Decision tree classifier for network intrusion detection with GA-based feature selection. *Proceedings of the 43rd annual Southeast regional conference* (Vol. 2, pp. 136–141). Georgia: ACM Publisher.

Tavallaee, M., Bagheri, E., Lu, W., Ali A. Ghorbani. (2009). A Detailed Analysis of the KDD CUP 99 Data Set. In: *Proceedings of IEEE Symposium on Computational Intelligence in Security and Defense Applications*.

Thomas, C., & Balakrishnan, N. (2009). Improvement in intrusion detection with ad vances in sensor fusion. *IEEE Transactions on Information Forensics and Security* (4), 542–551.

Weiming, Hu, Wei, Hu, & Maybank, S. (2008). AdaBoost based algorithm for network intrusion detection. *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics, 38*, 577–583.

Waikato environment for knowledge analysis (weka) version 3.5.7. Available on: http://www.cs.waikato.ac.nz/ml/weka/, 2008.

Xiang, C., & Lim, S. M. (2005). Design of multiple-level hybrid classifier for intrusion detection system. *IE EE Transaction on System, Man and Cybernetics, Part A: Cybernetics, 2*, 117–122.

Yasami, Y., & Mozaffari, S. P. (2009). *A novel unsupervised classification approach for network anomaly detection by K-Means clustering and ID3 decision tree learning methods*. *The Journal of Supercomputing*. Netherlands: Springer.

Zhou, Z.-H., & Jiang, Y. (2004). NeC4.5: Neural ensemble based C4.5. *IEEE Transactions on Knowledge and Data Engineering, 16*, 770–773.