International Journal of Automation and Computing 6(4), November 2009, 406-414
DOI: 10.1007/s11633-009-0406-y

A Neuro-genetic Based Short-term Forecasting Framework for Network Intrusion Prediction System

Siva S. Sivatha Sindhu[1,*] S. Geetha[2] M. Marikannan[3] A. Kannan[4]

[1]Department of Computer Science and Engineering, College of Engineering, Guindy, Annauniversity, Chennai 600025, India
[2]Department of Information Technology, Thiagarajar College of Engineering, Madurai 600015, India
[3]Department of Computer Science and Engineering, Institute of Road and Transport Technology, Erode 638316, India
[4]Department of Computer Science and Engineering, College of Engineering, Guindy, Annauniversity, Chennai 600025, India

Abstract: Information systems are one of the most rapidly changing and vulnerable systems, where security is a major issue. The number of security-breaking attempts originating inside organizations is increasing steadily. Attacks made in this way, usually done by "authorized" users of the system, cannot be immediately traced. Because the idea of filtering the traffic at the entrance door, by using firewalls and the like, is not completely successful, the use of intrusion detection systems should be considered to increase the defense capacity of an information system. An intrusion detection system (IDS) is usually working in a dynamically changing environment, which forces continuous tuning of the intrusion detection model, in order to maintain sufficient performance. The manual tuning process required by current IDS depends on the system operators in working out the tuning solution and in integrating it into the detection model. Furthermore, an extensive effort is required to tackle the newly evolving attacks and a deep study is necessary to categorize it into the respective classes. To reduce this dependence, an automatically evolving anomaly IDS using neuro-genetic algorithm is presented. The proposed system automatically tunes the detection model on the fly according to the feedback provided by the system operator when false predictions are encountered. The system has been evaluated using the Knowledge Discovery in Databases Conference (KDD 2009) intrusion detection dataset. Genetic paradigm is employed to choose the predominant features, which reveal the occurrence of intrusions. The neuro-genetic IDS (NGIDS) involves calculation of weightage value for each of the categorical attributes so that data of uniform representation can be processed by the neuro-genetic algorithm. In this system unauthorized invasion of a user are identified and newer types of attacks are sensed and classified respectively by the neuro-genetic algorithm. The experimental results obtained in this work show that the system achieves improvement in terms of misclassification cost when compared with conventional IDS. The results of the experiments show that this system can be deployed based on a real network or database environment for effective prediction of both normal attacks and new attacks.

Keywords: Genetic algorithm, intrusion detection system (IDS), neural networks, weightage calculation, knowledge discovery in databases (KDD), classification.

1 Introduction

Intrusion detection is an extremely essential part of today′s network-centric digital world. An intrusion into a computer system can be compared to a physical intrusion into a building by a thief. It is an entity gaining unauthorized access to resources. The unauthorized access is intended to steal or change information or to disrupt the valid use of the resource by an authorized user. Many mechanisms and technologies like firewalls, encryption[1−3], authentication, vulnerability checking, and access control policies can offer security but they are still susceptible to attacks from hackers who take advantage of system flaws and social engineering tricks. In addition, computer systems with no connection to public networks remain vulnerable to disgruntled employees or other insiders who misuse their privileges. This observation results in the fact that much more emphasis has to be placed on intrusion detection systems (IDSs).

1.1 Intrusion detection systems

Intrusion detection is the ability to determine that an intruder has gained, or is attempting to gain unauthorized access. An intrusion detection system is a tool used to make this determination. Intrusion detection generally takes one of two approaches: anomaly detection and signature recognition[1]. Signature recognition techniques store patterns of intrusion signatures, and compare those patterns with the observed activities for a match to detect an intrusion. Because signature recognition techniques are based on known patterns of intrusion signature, they cannot detect novel intrusions whose signature patterns are unknown. Anomaly detection techniques identify an intrusion when the observed activities in computer systems demonstrate a large deviation from the normal profile built on long-term normal activities.

Another classification of intrusion detection systems are host-based, network-based and distributed. A host-based system resides on a single host computer. It uses audit logs or network traffic records of a single host for processing and analysis. This type of system is limited in scope since it is only able to see its own host′s environment, and cannot detect simultaneous attacks against multiple hosts.

A network-based system is a dedicated computer, or special hardware platform, with detection software installed. It is placed at a strategic point on a network (like a gateway or sub network) to analyze all network traffic on that particular segment. This system can detect attacks against multiple hosts on a single subnet, but it usually cannot

Manuscript received December 24, 2008; revised February 20, 2009
*Corresponding author. E-mail address: sivathasindhu@gmail.com; sivatha127@yahoo.com

monitor multiple subnets at one time.

Distributed systems allow detection software modules to be placed throughout the network with a central controller collecting and analyzing the data from all the modules. This provides a robust mechanism for detecting intrusions across several subnets and several hosts. However, it requires a dedicated computer to act as the central controller; centralization can make it vulnerable to attack.

1.2 Neuro-genetic

Neural networks are universal function approximators that can map any nonlinear function without a prior assumption about the data. In a misuse prediction system, the network profile may change over time. To handle dynamic profiles, learning algorithms are required to track network behavior and adapt to a dynamically changing concept. Unlike traditional statistical models, neural networks are data-driven, non-parametric weak models, and they let the data speak for themselves. Therefore, neural networks are less susceptible to the model misspecification problem than most of the parametric models and are more powerful in describing the dynamics of network behavior time series than traditional statistical models. In addition, neural network methods scale up much better than linear statistical models as the size and complexity of the learning task grows.

In general, the learning steps of a neural network are as follows. First, a network structure is defined with a fixed number of inputs, hidden nodes and outputs. Second, an algorithm is chosen to realize the learning process. We employed genetic algorithm (GA)[4, 5] in the learning phase of the neural networks; since it is one of the most powerful tools for searching in large search spaces. However, it imposes few mathematical constraints in the shape of the function to optimize. The fusion of GA and artificial neural network (ANN) is not only able to select "optimal feature set" but also figures out "optimal weight values" for ANN forecasting.

1.3 Proposed system framework

This work aims at establishing a framework to apply neuro-genetic algorithm in order to detect abnormal behavior of the user in network based systems. The intelligent system component (see Fig. 1) is trained with the network audit data and a knowledge base is built upon the intelligence gained. The intelligent system component has two stages: learning stage and detection stage. The learning stage makes up the knowledge base for the neuro-genetic IDS (NGIDS) system. The training is accomplished using GA and it utilizes a data set containing normal and abnormal samples. It consists of preprocessing, genetic feature selector, and data set formation. The entry point into the NGIDS system would be via the detecting stage. This would constantly monitor the network traffic in the environment. Whenever any traffic pattern enters into the system, the identification and authentication unit identifies it and that information is passed to the preprocessor. The necessary traffic parameters are selected using genetic feature selection algorithm and given as input to neuro-genetic classifier. The neuro-genetic classifier analyzes the traffic pattern and takes appropriate actions based on the knowledge gained. A detailed description of this component is provided in Section 4.

The IDS is subjected to operate in a dynamically changing environment[6]. Newer types of attacks are evolving out so often.

Whenever the false prediction rate of the system is increasing, the neuro-genetic algorithm is run automatically and the newer attacks are categorized, respectively. It is sufficient that these newer signatures alone are fed into the system for learning. The new solutions are integrated into the existing systems without disturbing its performance.

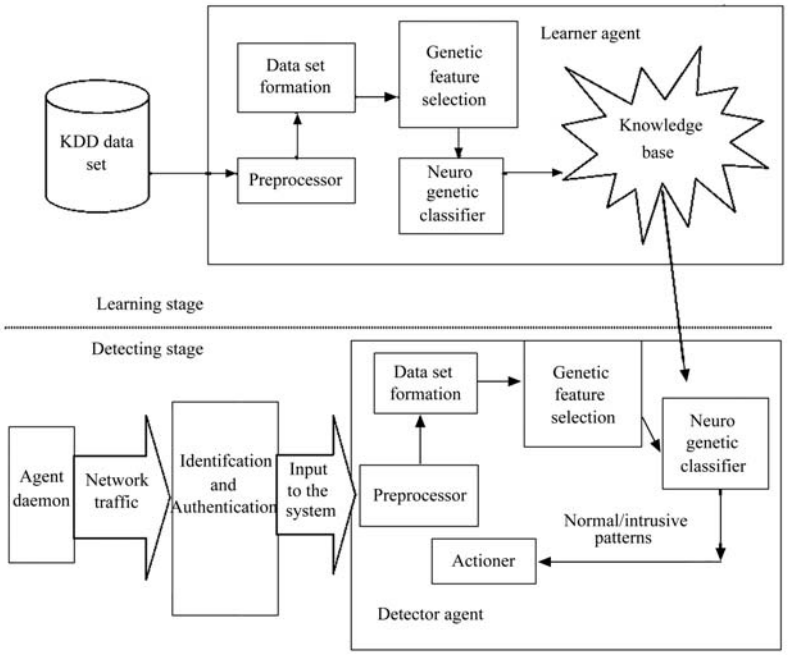

Fig. 1 Architecture of intelligent system component neuro-genetic classifier engine

2 Related work

An IDS is usually working in a dynamically changing environment, which forces continuous tuning of the intrusion detection model, in order to maintain sufficient performance.

A number of soft computing based approaches have been proposed for detecting network intrusions[7−11]. Soft computing refers to a group of techniques that exploit the tolerance for imprecision, uncertainty, partial truth, and approximation to achieve robustness and low solution cost. The principal constituents of soft computing are fuzzy logic (FL), ANNs, probabilistic reasoning (PR), and GAs. Despite different soft computing based approaches having been proposed, the possibilities of using the techniques for intrusion detection are still under-utilized.

Xiao et al.[8] narrated a k-means as a clustering algorithm applied to intrusion detection. However, with the deficiency of global search ability it is not satisfactory. Particle swarm optimization (PSO), which has high global search ability, is combined with k-means algorithm (PSO-KM) in this paper. Experiment over network connection records from KDD CUP 1999 data set was implemented to evaluate the proposed method. A Bayesian classifier was trained to select some fields in the data set. The experimental results clearly showed the outstanding performance of the proposed method. However, the method faced limitations in feature selection phase and the reliability of the system greatly depended on the selected features.

Bridges and Vaughn[10] developed a method that integrated fuzzy data mining techniques and GAs to detect both network misuses and anomalies. In most of the existing GA based IDSs, the quantitative features of network audit data are either ignored or simply treated, though such features are often involved in intrusion detection. This is because of the large cardinalities of quantitative features. The authors proposed a way to include quantitative features by introducing fuzzy numerical functions. Their preliminary experiments showed that the inclusion of quantitative features and the fuzzy functions significantly improved the accuracy of the generated rules. In this approach, a GA was used to find the optimal parameters of the fuzzy functions as well as to select the most relevant network features. Li[11] proposed a GA-based method to detect anomalous network behaviors. Both quantitative and categorical features of network data were included when deriving classification rules using GA. The inclusion of quantitative features may lead to increased detection rates. However, no experimental results are available yet. Triantafyllopoulos and Pikoulas[12] used Bayesian forecasting model to detect future network intrusions against systems. However, their method is unable to build a reliable prediction model when complexity or non-linearity is present in a data set.

Fang[13] described network intrusion detection based on pattern recognition even without plenty of prior knowledge about earlier intrusive signatures. A novel ensemble-learning algorithm for fuzzy classification rules is presented. The fuzzy antecedent is adjusted based on the combination of ensemble learning and induction-enhanced particle swarm optimization for intrusion detection. Tuning the distribution of training instances and joining the distribution factor in computing the fitness function take the collaboration of rules into account during rule generation phase. This method possesses good generalization ability and high classification rate.

Neuro-genetic forecasting model makes periodic short-term forecasts, since long-term forecasts cannot accurately predict an intrusion[9].

3 Neuro-genetic forecasting model

3.1 Training procedure

The purpose of the training process is to adjust the input and output parameters of the ANN model, so that the mean absolute percentage error (MAPE) measure is minimized. Training of the feed-forward neural network models is usually performed using gradient-based learning algorithms. A gradient descent algorithm is used to adopt the weights based on comparison between the desired and the actual network response to a given input. This gradient descent has several drawbacks: it is dependent on the shape of the error surface and sensitivity to the starting points (values of the usually randomly initialized weights)[9]. Most often, the error surface becomes trapped to local minima, usually not meeting the desired convergence criterion. The termination at a local minimum is a serious problem while the neural network is learning. In other words, such a neural network is not completely trained. Another issue where care must be taken is "the susceptibility to over-fitting". However, GAs offer an efficient search method for complex (i.e., possessing many local optima) spaces to find nearly global optima. Thus, its ability to find a better sub-optimal solution or have a higher probability to obtain the global optimal solution makes it one of the preferred candidates to solve the learning problem.

3.2 Training with GA

Section 3.3 shows the proposed algorithm. The parameters of the neural network are tuned by GA. A randomly initialized population (P) with 200 genotypes is considered. We ran GA for 100 generations with the same population size. The best model was found after 72 generations. First, two parents are selected from P by the method of tournament selection. Then, a new offspring is generated from these parents using arithmetic crossover and non-uniform mutation operators[9], which are governed by the probabilities of crossover and mutation, respectively. These operators decrease the training time to a great extent and help in locating an optimum solution[14, 15] relatively faster. In this work, the probability of crossover is 0.6 and the probability of mutation is 0.1. These probabilities are chosen by trial and error through experiments for good performance. The new population thus generated replaces the current population. The above procedures are repeated until a certain termination condition is satisfied. The termination condition is that the algorithm stops when a predefined number of generations have been processed or the change of the fitness values between the current and the previous iteration is less than 0.001.

3.3 Neuro-genetic algorithm overview

Algorithm 1. Anomaly detection using ensemble neuro-genetic algorithm.

Input. Network audit data, number of generations and population size.

Output. A trained NN that classifies the intrusion from normal behavior.

Step 1. Preprocess the dataset: normalization and weightage calculation.

Step 2. Genetic_Feature_Selection().

Step 3. Create input layer with the required number of neurons.

Step 4. Create hidden layer(s).

Step 5. Create output layer with the required number of neuron objects.

Step 6. Initialize $count = 0$, $fitness = 0$, $W1 = 0.7$, $W2 = 0.3$, number of generations as required.

Step 7. Generation of initial population. The chromosome of an individual is formulated as a sequence of consecutive genes, each one coding an input parameter.

Step 8. Assign random weights for each link in the neural network.

Step 9. Find output of all output neuron objects.

Step 10. Calculate MAPE, sum squared error (SSE), mean squared error (MSE), and the fitness value. The genotypes are evaluated on the basis of the fitness function.

$$SSE = \sum_{i=1}^{n} (target_i - predicted_i)^2$$

$$MSE = \frac{\sum_{i=1}^{n} (target_i - predicted_i)^2}{n}$$

$$MAPE = \frac{\sum_{i=1}^{n} \text{abs}(target_i - predicted_i)^2}{n}$$

$$fitness = (W1 \cdot sensitivity) + (W2 \cdot specificity).$$

Step 11. If abs(previous fitness – current fitness)<0.001 then quit.

Step 12. If (previous fitness < current fitness value) then store current weights.

Step 13. $count = count + 1$.

Step 14. Selection: two parents are selected by using tournament selection.

Step 15. For each chromosome in the new population, apply crossover operator to the chromosome, and apply mutation operator to the chromosome.

Step 16. End for

Step 17. If (number of generations > *count*), then go to Step 8.

Genetic_Feature_Selection() {

Algorithm 2. Feature set selection using genetic algorithm.

Input. Encoded binary string, number of generations, and population size, cross over probability, mutation probability.

Output. A set of selected features.

Step 1. Initialize the population randomly.

Step 2. $W1 = 0.7$, $W2 = 0.3$.

Step 3. N = total number of records in the training set.

Step 4. For each chromosome in the new population, apply uniform crossover operator to the chromosome with a probability of Pc, and apply mutation operator to the chromosome with a probability of Pm.

Step 5. Evaluate $fitness = W1 \cdot accuracy + W2 \cdot zeros$.

Step 6. Select the top best 60% of chromosomes into new population using tournament selection operator.

Step 7. If number of generations is not reached, then go to Step 4. }

3.4 Evaluate the individuals using fitness function

The objective of the fitness function is to minimize the prediction error. In order to prevent over-fitting and to give more exploration to the system, we have changed the fitness evaluation framework. The optimization problem is a two-goal objective function: maximize the *sensitivity*, and maximize the *specificity*. In this paper, we use the intrusion prediction system evaluation metrics to calculate the fitness of a chromosome instead of the prediction error with classical fitness function. The fitness of a chromosome for the normal class is evaluated according to the following equations:

$$TP = \sum_{i=1}^{p} predicted(normaldata_i) \tag{1}$$

$$TN = \sum_{i=1}^{q} [1 - predicted(abnormaldata_i)] \tag{2}$$

$$FP = \sum_{i=1}^{q} predicted(abnormaldata_i) \tag{3}$$

$$FN = \sum_{i=1}^{p} [1 - predicted(normaldata_i)] \tag{4}$$

$$sensitivity = \frac{TP}{TP + FN} \tag{5}$$

$$specificity = \frac{TN}{TN + FP} \tag{6}$$

$$fitness = (W_1 \cdot sensitivity) + (W_2 \cdot specificity) \tag{7}$$

where TP, TN, FP, and FN are the true positive, true negative, false positive, and false negative values for the testing data respectively, *predicted* is the predicted value of the training class, p and q are the number of normal and abnormal samples in the training data set used by each chromosome respectively, W_1 and W_2, are the assigned weights for each rule characteristic respectively, $normaldata_i$ is the subset of normal training patterns, and $abnormaldata_i$ is the subset of abnormal training patterns. The weights $W1$ and $W2$ are used to control the balance between the two terms and have the default values of $W1 = 0.7$ and $W2 = 0.3$. We performed tests with random values for the fitness function weights in order to give more importance to the specificity and sensitivity metrics. Sensitivity is much more crucial than specificity in IDSs. Thus, we preferred to choose the weights accordingly, and the weights are scaled such that the weights sum to one. The range of the fitness

function of neural network is [0,1]. The set of equations to calculate the fitness for the abnormal class can be obtained by changing abnormal to normal in previous equations. We use the same fitness function as in the previous simplification step for evaluating the population.

3.5 ANN implementation

In this work, the ANN is employed to learn the input-output relationship of an intrusion prediction model using the GA. NN (see Fig. 2) contains one input, one hidden, and one output layer. Each ANN has 9 input neurons corresponding to the 9 network audit data attributes selected using GA and the output layer has five neurons (one for normal and the others for each type of attack). The number of neurons in the hidden layer must be chosen carefully; if it is too small, the network may not be capable of adequately learning the behavior of the series; if it is too large, the network may become exceedingly specialized and thereby lose its generalizing capability. The number of hidden nodes of the traditional NN is chosen by Baum-Haussler rule through experiments for good performance.

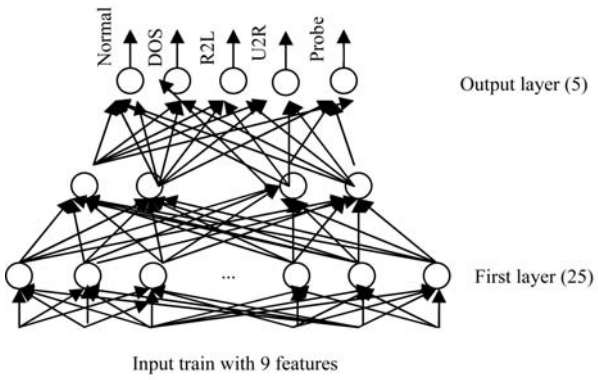

Fig. 2 NN structure for network intrusion prediction system

A rule of thumb, known as the Baum-Haussler rule, is used to determine the number of hidden neurons to be used:

$$N_{\text{hidden}} \leqslant \frac{N_{\text{train}} E_{\text{tolerance}}}{N_{\text{pts}} + N_{\text{output}}} \tag{8}$$

where N_{hidden} is the number of hidden neurons, N_{train} is the number of training examples, $E_{\text{toleracne}}$ is the error tolerance, N_{pts} is the number of data points per training example, and N_{outputs} is the number of output neurons. For comparison purposes, three-layered feed-forward ANN with a sigmoid function is used in this work and the back propagation algorithm is used for training the ANN. Java neural network simulator 1.1 (JNNS)[16] was used to implement this. The momentum and adaptive learning rate was 0.2 and 0.9, respectively. The number of iterations to train the neural network was 2000 for the classical back propagation neural network (BPN).

4 Experimental topology

4.1 Data source

For the experimental environment, it was difficult to obtain real-life datasets due to limitations of network size and limited external access. Unfortunately, usable datasets are rarely published as these involve sensitive information such as the network architecture, security mechanisms, and so on. Therefore, we rely on the Knowledge Discovery in Databases Conference (KDD 2009)[17] dataset (2004) provided by the Defense Advanced Research Projects Agency (DARPA) as this dataset contains several weeks of attack data and has been used to assess the performance of a number of IDS. It is considered as a standard benchmark for intrusion detection evaluations. In the KDD 2009 dataset, each record represents a TCP/IP network connection with a total of 41 features. Five classes of connections were identified, including normal network connections. In this dataset, forty-one attributes that usually characterize network traffic behavior compose of each record. The record pattern may be normal or of attack type. There are a total of 22 different types of attacks reported and these attacks fall into four main categories. They are DOS, U2R, R2L, and probe. The distribution of records is provided in Table 1.

Table 1 Distribution of data in KDD data set

Attacks data	Normal (%)	Probe (%)	DOS (%)	R2L (%)	U2R (%)
Training data	19.69	0.83	79.24	0.01	0.23
Test data	19.48	1.34	73.9	0.07	5.21

4.2 Data preprocessing

The essence of any successful applications lies at the recognition of a need for effective data preprocessing. The KDD raw data is converted into machine-readable form. The 41 features in the KDD data set are converted into a standardized numeric representation. The process involves the calculation of weightage value (W) for each of the categorical attributes using the weight calculation formula (9) so that the neuro-genetic algorithm can process data of uniform representation.

$$W = \frac{\#AttributeAbnormal}{\#Abnormal} \tag{9}$$

where $\#AttributeAbnormal$ is the number of abnormal records in which the attribute type is present and $\#\ Abnormal$ is the total number of abnormal records. For example, to find the weightage of the service HTTP, the number of abnormal records in which HTTP is present is divided by the total number of abnormal records therefore the data of uniform representation can be processed by the neuro-genetic algorithm. The resultant service weights (sample) are summarized in Table 2.

Table 2 Weightage calculation

Service type	Calculated weight
ecr_I	0.8123
ftp_data	0.0027
eco_I	0.0025
Private	0.1692
Discard	0.0041
ftp	0.0012

4.3 Normalization

In order to achieve maximum accuracy, each numerical value in the data set is normalized over the range [0.05, 0.95] according to the following data smoothing method:

$$x = \frac{x - Min}{Max - Min} \tag{10}$$

where x is the numerical value, 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. In order to avoid difficulty in getting predicted outputs very close to the two endpoints, the data inputs were normalized to the range $[0.05, 0.95]$ instead of $[0, 1]$ before it is inputted to the network.

4.4 Genetic feature selection

Among the 41 attributes available in KDD dataset, the most contributing 9 attributes are selected using genetic feature selection algorithm. The selected features are listed in Table 3.

Table 3 Selected attributes

Number	Method	Selected attribute count	Selected attributes
1	Best first	18	2,3,5,6,8,12,23, 24,30,32,33,35, 36,37,38,40,41
2	Information gain	16	3,5,6,8,12,24,29, 30,32,34,35,36, 37,38,39,40
3	Greedy search	15	2,3,5,6,8,12,23, 31,32,33,35,36 38,39,40
4	By our approach	9	4,5,6,7,16,27,30,37,39

4.4.1 Encoding

Each network traffic pattern is represented as a vector of 41 features, which are the signatures of the respective network behavior. In our encoding scheme, the chromosome is a bit string whose length is determined by the number of selected features. Each feature is associated with one bit in the string. If the i-th bit is 1, then the i-th feature is selected, otherwise, that component is ignored. Each chromosome thus represents a different feature subset.

4.4.2 Initial population

The initial population is generated randomly. To explore subsets of different numbers of features, the number of $1'$s for each individual is generated randomly. Then, the $1'$s are randomly scattered in the chromosome.

4.4.3 Fitness evaluation

The goal of feature subset selection is to use fewer features to achieve the same or better performance. Therefore, the fitness evaluation contains two terms: accuracy from the validation data and number of features used. The performance of the classifier is estimated using a validation data set and used to guide the GA. Each feature subset contains a certain number of features. If two subsets achieve the same performance, while containing different number of features, the subset with fewer features is preferred. Between accuracy and feature subset size, accuracy is our major concern. Combining these two terms, the fitness function is given as

$$fitness = 0.7 \cdot accuracy + 0.3 \cdot zeros \tag{11}$$

where *accuracy* is the accuracy rate that an individual achieves, and *zeros* is the number of zeros in the chromosome. The accuracy ranges roughly from 0.7 to 1. The number of zeros ranges from 0 to 41 where 41 is the length of the chromosome (i.e., the second term assumes values in the interval 0 to 12.3 since $L = 41$ here). Overall, higher accuracy implies higher fitness. Also, fewer features used imply a greater number of zeros, and as a result, the fitness increases. It should be noted that individuals with higher accuracy would outweigh individuals with lower accuracy, no matter how many features they contain.

4.4.4 Crossover and mutation

In general, we do not know how the features depend upon each other. If dependent features are far apart in the chromosome, it is more probable that traditional 1-point crossover will destroy the schemata. To avoid this problem, uniform crossover is used. Mutation is a very low probability operator and just flips a specific bit. It plays the role of restoring lost genetic material. Our selection strategy is cross generational. Assuming a population of size N, the offspring double the size of the population, and we select the best N individuals from the combined parent offspring population using tournament selection.

4.5 Training and testing data

In order to conduct an experimental setting, 20% of each class was used for testing and the remaining 80% was used for training. The combined proportion of samples from the normal class and the denial of service (DOS) class are almost 99% of the data set, so a larger number of samples was kept for these two classes in the training data set. Each point of the training data is located in the 9-dimensional space, with each dimension corresponding to a selected feature of the data point.

5 Experimental results

The learning process is carried out under Java development kit (JDK) 1.3 for Windows XP on a Pentium IV 2 GHz CPU with 512 Mb of RAM. The back-propagation NN model was developed using Java[16], and GA was programmed in Java using Java genetic algorithms package (JGAP) by the authors.

5.1 Error measurement

The quality of the forecasts is measured by the two widely accepted error metrics: SSE and MSE. Both accuracy measures calculate the squares of the errors; errors is larger than the one being amplified and those are less than the one being lessened. Tables 4 and 5 compare the SSE, MSE computed over the training dataset by the following predictors. Table 6 shows the fitness evaluated using the proposed model.

Table 4 Sum squared error (SSE)

Models	DOS	R2L	U2R	Probe	Normal	Average of SSE
Neuro-genetic	0.2671	0.2533	0.3176	0.4077	0.1673	0.2826
BPN	0.369	0.858	0.921	1.192	0.764	0.8208

Table 5 Mean squared error (MSE)

Models	DOS	R2L	U2R	Probe	Normal	Average of MSE
Neuro-genetic	0.010622	0.011147	0.01325	0.018746	0.006154	0.0119838
BPN	0.0357	0.036	0.038	0.0484	0.0318	0.0379

Table 6 Fitness value

Models	DOS	R2L	U2R	Probe	Normal	Average of $fitness$
Neuro-genetic	0.9861	0.9865	0.9572	0.9533	0.9924	0.9751

5.2 Accuracy/error trends

The graph in Fig. 3 shows the error trend curves obtained during training and testing. From the nature of the curves it is obvious that the neuro-genetic model is intelligent enough to learn the relationships between the inputs and the respective outputs.

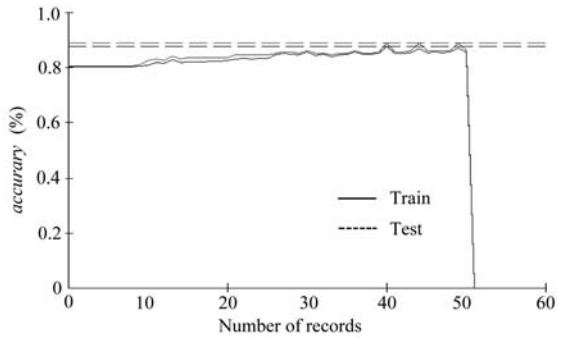

Fig. 3 Accuracy curves obtained during training and testing

5.3 Predicted versus desired graphs

The graphs in Figs. 4–8 compare the predicted versus desired values of our network during validation. They are for the normal behavior and for the four classes of attacks. It is inferred that probing attack is a bit challenging, i.e., the output from our model (predicted) is not tracing the desired output defined by the KDD data set.

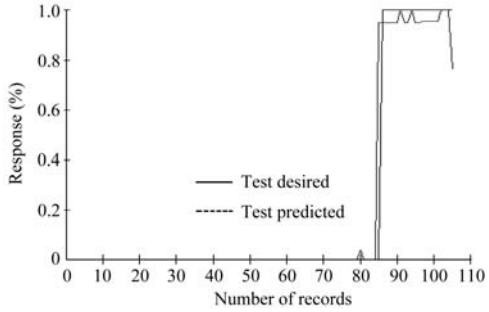

Fig. 4 Desired versus predicted for R2L

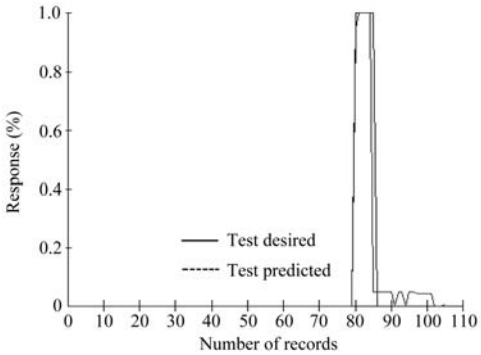

Fig. 5 Desired versus predicted for U2R

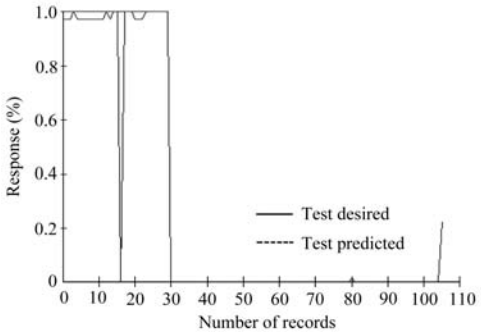

Fig. 6 Desired versus predicted for normal

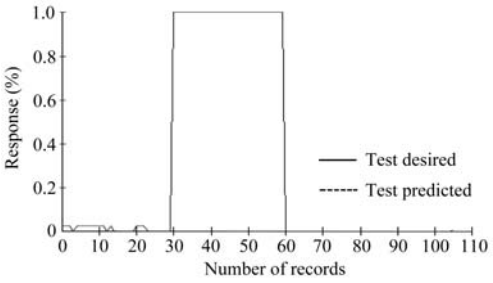

Fig. 7 Desired versus predicted for DOS

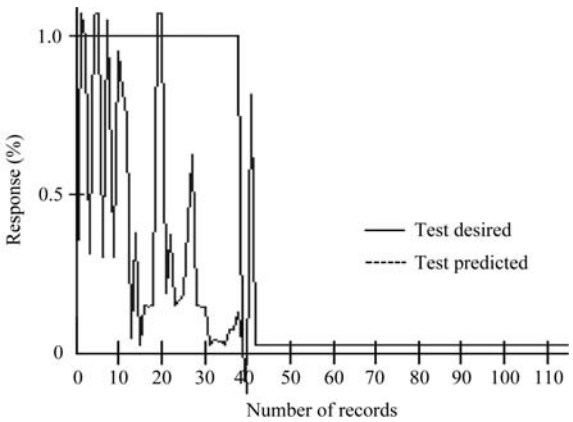

Fig. 8 Desired versus predicted for probe

5.4 Receiver operating characteristic (ROC) analysis

The intrusion detection method presented in this paper is evaluated using a receiver operating characteristic (ROC)

analysis, which has been widely applied in the field of intrusion detection. There are four elements defining the cost function of an intrusion detection system: TP is the true positive rate (the percentage of malicious events (intrusions) that are correctly identified as intrusive events) and FP is the false positive rate (the percentage of legitimate events that are incorrectly reported as intrusive events). TN occurs when a normal event is reported as a normal event. FN occurs when an intrusive event is signaled as a normal event. An ROC curve displays the tradeoff between the pairs of false alarm rate and detection rate (FP, TP) on the testing data.

We generated the average ROC curve for the 23 user patterns (22 different types of attack and 1 norma1). In an ROC curve the detection rate (*sensitivity*) is plotted as a function of the false alarm rate (100-*specificity*) for different cut-off points. Each point on the ROC plot represents a sensitivity/specificity pair corresponding to a particular prediction signal threshold. A test with perfect discrimination (no overlap in the two distributions) has an ROC plot that passes through the upper left corner (100% *sensitivity*, 100% *specificity*). Therefore, the closer the ROC plot is to the upper left corner, the higher the overall accuracy of the test. Fig. 9 shows the ROC curves of the BPN neural network and the neuro-genetic ensemble by plotting pairs of false alarm rate and detection rate. Since the ROC curve of the neuro-genetic is nearer to the upper-left corner than the ROC curve of the BPN test, the neuro-genetic ensemble model performs better than the BPN method in intrusion detection with respect to both false alarm rate and hit rate. Detection rate for neuro-genetic ensemble model is listed in Table 7.

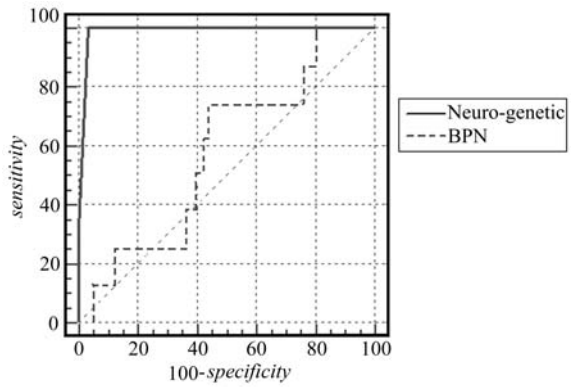

Fig. 9 ROC characteristics of BPN and neuro-genetic ensemble

Table 7 Detection rates for neuro-genetic ensemble model

Data set	Detection rate	False positive alarm	False negative alarm
DOS	0.968	-	0.032
R2L	0.97	-	0.03
U2R	0.885	-	0.115
Probe	0.6873	-	0.3127
Normal	0.998	0.002	-

6 Conclusions

Because computer networks are continuously changing, it is difficult to collect high-quality training data to build intrusion detection models. In this paper, rather than focus on building a highly effective initial detection model, we propose to improve a detection model dynamically after the model is deployed when it is exposed to new data. In our approach, the detection performance is fed back into the detection model, and the model is adaptively tuned.

In this paper, an automatically evolving anomaly intrusion detection system is constructed using neuro-genetic ensemble and its performance is tested on a set of benchmark Massachusetts Institute of Technology (MIT) Lincoln labs data. Our results show that the proposed model is a promising detection technique, outperforming the traditional BPN algorithm in the benchmark test. It is also concluded that the genetic algorithm makes the feature selection process highly efficient in the IDS. Since there is no gradient information present in the feature values, it is otherwise difficult to formulate a directed search for feature selection. The automatically evolving neuro-genetic algorithm caters to the dynamically changing needs of effective IDS. NGIDS imposes a relatively small burden on the system operator; operators need to mark the false alarms after they identify them.

These results are encouraging. The overall detection rate of NGIDS is 90.2%, somewhat higher than conventional IDS. We have further noticed that if system behavior changes drastically or if the tuning is delayed too long, the benefit of model tuning might be diminished or even negative. In the former case, new patterns could be trained and added to the detection model. If it takes too much time to identify a false prediction, tuning on this particular false prediction is easily prevented as long as the prediction result is not fed back to the model tuner.

References

[1] N. Ye, S. M. Emran, Q. Chen, S. Vilbert. Multivariate Statistical Analysis of Audit Trails for Host-based Intrusion Detection. *IEEE Transactions on Computers*, vol. 51, no. 7, pp. 810–820, 2002.

[2] D. Naccache. Secure and Practical Identity-based Encryption. *IET Information Security*, vol. 1, no. 2, pp. 59–64, 2007.

[3] C. W. Chang, H. Pan, H. Y. Jia. A Secure Short Message Communication Protocol. *International Journal of Automation and Computing*, vol. 5, no. 2, pp. 202–207, 2008.

[4] R. Curry, P. Lichodzijewski, M. I. Heywood. Scaling Genetic Programming to Large Datasets Using Hierarchical Dynamic Subset Selection. *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, vol. 37, no. 4, pp. 1065–1073, 2007.

[5] S. X. Yang, R. Tin. A Hybrid Immigrants Scheme for Genetic Algorithms in Dynamic Environments. *International Journal of Automation and Computing*, vol. 4, no. 3, pp. 243–254, 2007.

[6] Z. W. Yu, J. J. P. Tsai, T. Weigert. An Automatically Tuning Intrusion Detection System. *IEEE Transactions on Sys-*

tems, Man, and Cybernetics – Part B: Cybernetics, vol. 37, no. 2, pp. 373–384, 2007.

[7] F. Berzal, J. C. Cubero. An Effective Algorithm for Mining Subspace Clusters in Categorical Datasets. *Data and Knowledge Engineering*, vol. 60, no. 1, pp. 51–70, 2007.

[8] L. Z. Xiao, Z. Q. Shao, G. Liu. K-means Algorithm Based on Particle Swarm Optimization Algorithm for Anomaly Intrusion Detection. In *Proceedings of the 6th World Congress on Intelligent Control and Automation*, IEEE Press, vol. 2, pp. 5854–5858, 2006.

[9] F. H. F. Leung, H. K. Lam, S. H. Ling, P. K. S. Tam. Tuning of the Structure and Parameters of a Neural Network Using an Improved Genetic Algorithm. *IEEE Transactions on Neural Networks*, vol. 14, no. 1, pp. 79–88, 2003.

[10] S. M. Bridges, R. B. Vaughn. Fuzzy Data Mining and Genetic Algorithms Applied to Intrusion Detection. In *Proceedings of the 12th Annual Canadian Information Technology Security Symposium*, Baltimore, USA, pp. 109–122, 2000.

[11] W. Li. *A Genetic Algorithm Approach to Network Intrusion Detection*, SANS Institute, USA, 2004.

[12] K. Triantafyllopoulos, J. Pikoulas. Multivariate Bayesian Regression Applied to the Problem of Network Security. *Journal of Forecasting*, vol. 21, no. 8, pp. 579–594, 2002.

[13] F. Min. A Novel Intrusion Detection Method Based on Combining Ensemble Learning with Induction-enhanced Particle Swarm Algorithm. In *Proceedings of the 3rd International Conference on Natural Computation*, IEEE Press, vol. 3, pp. 520–524, 2007.

[14] D. Parrott, X. D. Li. Locating and Tracking Multiple Dynamic Optima by a Particle Swarm Model Using Speciation. *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 4, pp. 440–458, 2006.

[15] S. C. Chiam, K. C. Tan, A. S. Al Mamum. Evolutionary Multi-objective Portfolio Optimization in Practical Context. *International Journal of Automation and Computing*, vol. 5, no. 1, pp. 67–80, 2008.

[16] Java Neural Network Simulator, Department of Computer Architecture, Wilhelm-Schickard-Institute for Computer Science, University of Tbingen, [online], Available: http://www-ra.informatik.uni-tuebingen.de/downloads/JavaNNS.

[17] KDD-cup 1999 data, [Online], Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, October 28, 1999.

Siva S. Sivatha Sindhu received the B. Eng. and M. Eng. degrees in computer science and engineering from Manonmaniam Sundaranar University and Anna University, India in 2002 and 2004, respectively. She has more than four years of teaching experience in the field of information technology. She is currently doing her research in the area of network security in Anna University, Chennai, India.

Her research interests include information security, intrusion detection systems, and soft computing approaches.

S. Geetha received the B. Eng. and M. Eng. degrees in computer science and engineering from Madurai Kamaraj University and Anna University, India in 2000 and 2004, respectively. She is currently with the Department of Information Technology, Thiagarajar College of Engineering, Madurai, Tamilnadu, India. She has more than six years of teaching experience in the field of computer science and engineering.

Her research interests include data hiding algorithms, information security, and soft computing approaches.

M. Marikkannan received the B. Eng. degree in computer science and engineering from the Government College of Engineering, Tirunelveli, India in 1994, and M. Eng. degree in computer science and engineering from College of Engineering, Anna University, Chennai, India in 1999. Currently, he is a lecturer at the Department of Computer Science and Engineering, Institute of Road and Transport Technology (IRTT), Erode, India.

His research interests include temporal database management systems and object oriented systems.

A. Kannan received the M. Sc. degree in mathematics from Annamalai University, India in 1986, M. Eng. degree in computer science and engineering from the College of Engineering, Anna University, Chennai, India in 1991, and Ph. D. degree in computer science and engineering from College of Engineering, Anna University, Chennai in 2000. He had been with Bhabha Atomic Research Centre (BARC), Bombay, India as a computer programmer from August 1981 to March 1989. He was a lecturer at Anna University from September 1991 to November 2000, and an assistant professor from December 2000 to July 2006. He has been a professor at the Department of Computer Science and Engineering, Anna University, since July 2006.

His research interests include database management systems, knowledge engineering, and software engineering.