INTRUSION DETECTION SYSTEM USING BACK PROPAGATION ALGORITHM AND COMPARE ITS PERFORMANCE WITH SELF ORGANIZING MAP

Subarna Shakya[1], Bisho Raj Kaphle[2]

[1]Department of Electronics and Computer Engineering

Central Campus, Pulchowk, Lalitpur

Email Address: drss@ioe.edu.np

[2] Kathmandu Engineering College

Email Address: bishorajkafle@gmail.com

Abstract

In recent years, internet and computers have been utilized by many people all over the world in several fields. On the other hand, network intrusion and information safety problems are ramifications of using internet. In this thesis it propose a new learning methodology towards developing a novel intrusion detection system (IDS) by back propagation neural networks (BPN) and self organizing map (SOM) and compare the performance between them. The main function of Intrusion Detection System is to protect the resources from threats. It analyzes and predicts the behaviors of users, and then these behaviors will be considered an attack or a normal behavior. The proposed method can significantly reduce the training time required. Additionally, the training results are good. It provides a powerful tool to help supervisors and unsupervisors analyze, model and understand the complex attack behavior of electronic crime.

Keywords: *Intrusion Detection, Neural Network, Back Propagation Neural Network, Intrusion Attacks, Self Organizing Map*

1. Introduction

The problem of protecting information has existed since information has been managed. However, as technology advances and information management systems become more and more powerful, the problem of enforcing information security also becomes more critical. The enlargement of this electronic environment comes with a corresponding growth of electronic crime where the computer is used either as a tool to commit the crime or as a target of the crime [1].

In past years, numerous computers are hacked because they do not consider the necessary of precautions to protect against network attacks. The failure to secure their systems puts many companies and organizations at a much greater risk of loss. Usually, a single attack can cost millions of dollars in potential revenue. Moreover, that's just the beginning. The damages of attacks include not only loss of intellectual property and liability for compromised customer data (the time/money spent to recover from the attack) but also customer confidence and market advantage. There is a need to enhance the security of computers and networks for protecting the critical infrastructure from threats. Accompanied by the rise of electronic crime, the design of safe-guarding information infrastructure such as the intrusion detection system (IDS) for preventing and detecting incidents becomes increasingly challenging. The intrusion detector learning task is to build a predictive model (i.e. a classifier) capable of distinguishing between

bad intrusions and normal connections. Recently, an increasing amount of research has been conducted on applying neural networks to detect intrusions. An artificial neural network consists of a collection of processing elements that are highly interconnected. Give a set of inputs and a set of desired outputs, the transformation from input to output is determined by the weights associated with the interconnections among processing elements. There are two general methods of detecting intrusions into computer and network systems, namely Anomaly detection and Signature recognition.

Anomaly detection techniques establish a profile of the subject's normal behavior (norm profile), compare the observed behavior of the subject with its norm profile, and signal intrusions when the subject's observed behavior differs significantly from its norm profile. Signature recognition techniques recognize signatures of known attacks, match the observed behavior with those known signatures, and signal intrusions when there is a match [2].

Neural network is an universal classifier and with the proper choosing of its architecture it can solve any, even very complicated, classification task [3].

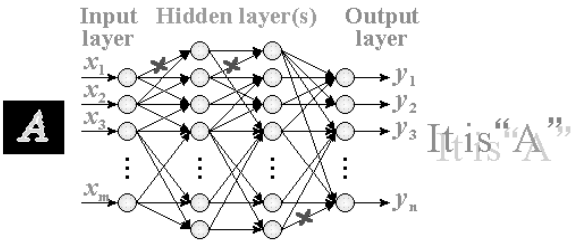

Fig 1 Multilayer Perceptron[1]

Here above figure 1.2 shows the input layer, hidden layer(s) and output layer of Multilayer Perceptron (MLP).
Attacks can be gathered in four main categories:

1) Denial of Service Attack (DoS): is an attack in which the attacker makes some computing or memory resource too busy or too full to handle legitimate requests, or denies legitimate users access to a machine.
2) User to Root Attack (U2R): is a class of exploit in which the attacker starts out with access to a normal user account on the system (perhaps gained by sniffing passwords, a dictionary attack, or social engineering) and is able to exploit some vulnerability to gain root access to the system.

3) Remote to Local Attack (R2L): occurs when an attacker who has the ability to send packets to a machine over a network but who does not have an account on that machine exploits some vulnerability to gain local access as a user of that machine.
4) Probing Attack: Attacker tries to gain information about the target host [2].
Activation Function:

Multilayer perceptron networks typically use sigmoid transfer functions in the hidden layers. These functions are often called "squashing" functions, because they compress an infinite input range into a finite output range.

The bipolar sigmoid function: $f(x) = -1 + 2/ [1+e^{-x}]$
which has derivative of: $f'(x) = 0.5 * [1 + f(x)] * [1 – f(x)]$

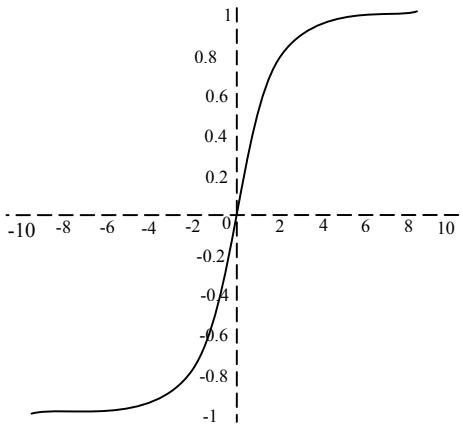

Fig 2 Bipolar Sigmoid Function

2. Literature Review

At first the concept of intrusion detection system was suggested by Anderson (1980) [1]. He applied statistic method to analyze user's behavior and to detect those attackers who accessed system in an illegal manner. In [2] proposed a prototype of IDES (intrusion detection expert system) in 1987, subsequently, the idea of intrusion detection system was known progressively, and paper was regarded as significant landmark in this area. In [4] the author proposed a data mining framework for constructing intrusion detection models. The key idea is to apply data mining programs namely, classification, meta-learning, association rules, and frequent episodes to audit data for computing misuse and anomaly detection models that accurately capture the actual behavior (i.e., patterns) of intrusions and normal activities. Although, proposed detection model can detect a high percentage of old and new PROBING and U2R attacks, it missed a large number of new DOS and R2L attacks. Reference [5] is mostly focused on data mining techniques that are being used for such purposes, and then presented a new idea on how data mining can aid IDSs by utilizing biclustering as a tool to analyze network traffic and enhance IDSs. Reference [6] proposed a new weighted support vector clustering algorithm and applied it to the anomaly detection problem. Experimental results show that mentioned method achieves high detection rate with low false alarm rate. Intrusion detection attacks are segmented into two groups,

- Host-based attacks [3-5] and
- Network-based attacks [6, 7].

Intruders attack these systems by transmitting huge amounts of network traffic, utilizing familiar faults in networking services, overloading network hosts, etc. Detection of these kinds of attacks uses network traffic data (i.e., tcpdump) to look at traffic addressed to the machines being monitored.

3. Research Methodology

Back Propagation Algorithm

The back propagation algorithm is a quite essential one of the neural network. The algorithm is the training or learning algorithm rather than the network itself. The network used is generally of the simple type shown in figure 3.1, and in the examples up until now. The network operates in exactly the same way as the others have seen. Now, let's consider what Back Propagation is and how to use it. A Back

Propagation network learns by example. You give the algorithm examples of what you want the network to do and it changes the network's weights so that, when training is finished, it will give you the required output for a particular input. Back Propagation networks are ideal for simple Pattern Recognition and Mapping Tasks4. As just mentioned, to train the network you need to give it examples of what you want the output you want (called the Target) for a particular input as shown in Figure 3.1.

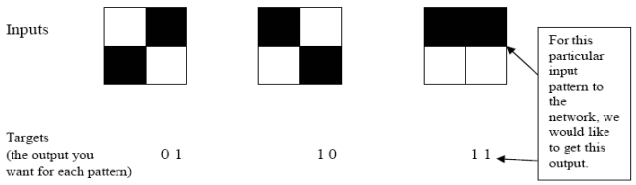

Fig 3.1 A Back Propagation Training Set.

So, if the first pattern to the network, we would like the output to be 0 1 as shown in figure 3.1 (a black pixel is represented by 1 and a white by 0 as in the previous examples). The input and its corresponding target are called a Training Pair.

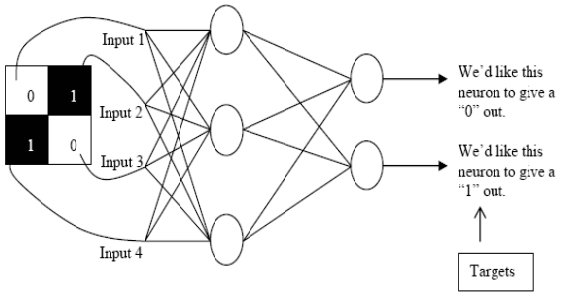

Fig 3.2 Applying a Training Pair to a Network

Once the network is trained, it will provide the desired output for any of the input patterns. Let's now look at how the training works. The network is first initialized by setting up all its weights to be small random numbers – say between –1 and +1. Next, the input pattern is applied and the output calculated (this is called the forward pass). The calculation gives an output which is completely different to what you want (the Target), since all the weights are random. Then calculate the Error of each neuron, which is essentially: Target – Actual Output (i.e. what you want – What you actually get). This error is then used mathematically to change the weights in such a way that the error will get smaller. In other words, the Output of each neuron will get closer to its Target (this part is called the reverse pass). The process is repeated again and again until the error is minimal. Let's do an example with an actual network to see how the process works. Just look at one connection initially, between a neuron in the output layer and one in the hidden layer, figure 3.2.

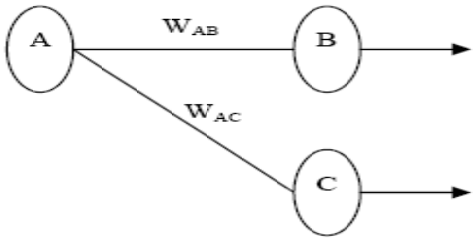

Fig 3.3 A Single Connection Learning in a Back Propagation network.

The connection interested in is between neuron A (a hidden layer neuron) and neuron B (an output neuron) and has the weight WAB. The diagram also shows another connection, between neuron A and C, but we'll return to that later. The algorithm works like this:

1. First apply the inputs to the network and work out the output – remember this initial output could be anything, as the initial weights were random numbers.
2. Next work out the error for neuron B. The error is what you want – What you actually get, in other words: ErrorB = OutputB (1-OutputB) (TargetB – OutputB) The "Output (1-Output)" term is necessary in the equation because of the Sigmoid Function – if we were only using a threshold neuron it would just be (Target –Output).
3. Change the weight. Let W+AB be the new (trained) weight and WAB be the initial weight. W+AB = WAB + (ErrorB x OutputA) Notice that it is the output of the connecting neuron (neuron A) we use (not B). We update all the weights in the output layer in this way.
4. Calculate the Errors for the hidden layer neurons. Unlike the output layer we can't calculate these directly (because we don't have a Target), so we Back Propagate them from the output layer (hence the name of the algorithm). This is done by taking the Errors from the output neurons and running them back through the weights to get the hidden layer errors. For example if neuron A is connected as shown to B and C then we take the errors from B and C to generate an error for A. ErrorA = Output A (1 - Output A) (ErrorB WAB + ErrorC WAC) Again, the factor "Output (1 - Output)" is present because of the sigmoid squashing function.
5. Having obtained the Error for the hidden layer neurons now proceed as in stage 3 to change the hidden layer weights. By repeating this method we can train a network of any number of layers. W+ represents the new, recalculated weight, whereas W (without the superscript) represents the old weight. Reverse process is done in the same way. Hence the attackers are calculated.

Self-Organizing Map

The Self-Organizing Map [9] is a neural network model for analyzing and visualizing high dimensional data. It belongs to the category of competitive learning network. The SOM defines a mapping from high dimensional input data space onto a regular two-dimensional array of neurons. In designed architecture is input vector with six input values and output is realized to 2 dimension space. Every neuron i of the map is associated with an n-dimensional reference vector.

$$M_i[M_1 \dots \dots \dots \dots \dots M_n]^T \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots (3.1)$$

Where, n denotes the dimension of the input vectors. The reference vectors together form a codebook. The neurons of the map are connected to adjacent neurons by a neighborhood relation, which dictates the

topology, or the structure, of the map. Adjacent neurons belong to the neighborhood Ni of the neuron i. In the SOM algorithm, the topology and the number of neurons remain fixed from the beginning. The number of neurons determines the granularity of the mapping, which has an effect on the accuracy and generalization of the SOM. During the training phase, the SOM forms elastic net that is formed by input data. The algorithm controls the net so that it strives to approximate the density of the data. The reference vectors in the codebook drift to the areas where the density of the input data is high. Eventually, only few codebook vectors lie in areas where the input data is sparse. The learning process of the SOM goes as follows:

1. One sample vector x is randomly drawn from the input data set and its similarity (distance) to the codebook vectors is computed by using Euclidean distance measure:

$$\|x - m_c\| = min\{\|x - m_i\|\} \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (3.2)$$

2. After the BMU has been found, the codebook vectors are updated. The BMU itself as well as its topological neighbors are moved closer to the input vector in the input space i.e. the input vector attracts them. The magnitude of the attraction is governed by the learning rate. As the learning proceeds and new input vectors are given to the map, the learning rate gradually decreases to zero according to the specified learning rate function type. Along with the Intrusion Detection System, Using Self Organizing Map learning rate, the neighborhood radius decreases as well. The update rule for the reference vector of unit i is the following:

$$m_i(t+1) = m_i(t) + \alpha(t)\big[\big(x(t) - m_i(t)\big)\big], I \in N_c(t)$$

$$m_i(t+1) = m_i(t), I \in N_c(t) \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (3.3)$$

3. The steps 1 and 2 together constitute a single training step and they are repeated until the training ends. The number of training steps must be fixed prior to training the SOM because the rate of convergence in the neighborhood function and the learning rate are calculated accordingly.

Mapping Precision

The mapping precision measure describes how accurately the neurons respond to the given data set. If the reference vector of the BMU calculated for a given testing vector xi is exactly the same xi, the error in precision is then 0. Normally, the number of data vectors exceeds the number of neurons and the precision error is thus always different from 0. A common measure that calculates the precision of the mapping is the average quantization error over the entire data set:

$$E_q = \frac{1}{N}\sum_{i=1}^{N}\|x_i + m_c\| \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (3.4)$$

Topology Preservation

The topology preservation measure describes how well the SOM preserves the topology of the studied data set. Unlike the mapping precision measure, it considers the structure of the map. For a strangely twisted map, the topographic error is big even if the mapping precision error is small. A simple method for calculating the topographic error:

$$E_q = \frac{1}{N}\sum_{i=1}^{N} u(x_x) \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (3.5)$$

Where $u(x_x)$ is 1 if the first and second BMUs of $x_k$are not next to each other. Otherwise $u(x_k)$ is 0.

SOM Implementation to Intrusion Detection System

The goal of the proposed architecture is to investigate effectiveness of application a neural network SOM figure.3.5 at modeling user behavioral patterns so they can distinguish between normal and abnormal behavior. In order to model user behavior identified and isolated the system logs that were required as sources of information for the networks. These logs being common log data provided the required user activity information from where system derived the following behavioral characteristics which typifies users on the system:

- User activity times - The time at which a user is normally active.
- User login hosts - The set of hosts from which a user normally logs in from.
- User foreign hosts - The set of hosts which a user normally accesses via commands on the system (FTP hosts).
- Command set - The set of commands which a user normally uses.
- CPU usage - The typical CPU usage patterns of a user.
- Memory usage – The typical usage of memory for a user.

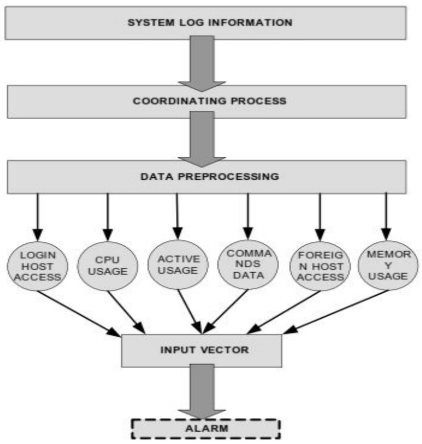

Fig 3.4 Structure of an Automated User Behavior Anomaly Detection System

Figure 3.5 illustrates how a complete system for the detection of user behavioral anomalies is structured. The coordination process is responsible for channeling system information to the neural networks. Each of the behavioral characteristics are both modeled by a SOM network, as well as checked by a limited static rule filter for easy breaches of security. Data acquired from the system logs is required to filter through input data preprocessor. The input to the neural network Fig. 3.5 represents data vector consisting from data controlled on the monitored system. Before input vector processing it is needed to normalize input data. The input to neural network is data vector, which consists from six properties representing User activity times, User login hosts, User foreign hosts, Command set CPU usage and Memory Usage. According to large numbers of variations of this data it is necessary to normalize every input vector to be

value in range of values [-1, 1]. This range comes out from the previous applications of neural network to system IDS realized within research activity on the Department of Computers and Informatics in Kosice. This normalization is more suitable for implementation in proposed SOM network. The architecture uses normalization given by:

$$nv[i] = \frac{v[i]}{\sqrt{\sum_k^n v[k]^2}} \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (3.6)$$

Where *nv[i]* is the normalized value of feature *(i)*, *v[i]* is the feature value of *i*, and *K* is the number of features in a vector. The processing realized by the SOM network consequently produces results for every user characteristic gives as input to the SOM network. Expected network reply is the value close to-for user, which behavior does not divert from normal behavior. If the value for given user exceeds specified threshold value obtained through the SOM network representing its intrusion behavior denotes raising alarm. If the output value of network is above specified threshold value, alarm is raised. It is necessary to remark that basic request for this detection mechanism is to setup threshold value to specific system whereby make it possible to adapt sensitivity directly to computer system.

4. Data Analysis

Input Dataset Analysis

Under the sponsorship of Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory (AFRL), the MIT Lincoln laboratory has established a network and captured the packets of different attack types and distributed the data sets for the evaluation of researches in computer network intrusion detection systems. The KDDCup99 data set is a subset of the DARPA benchmark data set [9]. Each KDDCup99 training connection record contains 41 features and is labeled as either normal or an attack, with exactly one specific attack type. This dataset will be taken as training data for performing the proposed research work. The result thus obtained will be compared with the rest of test data set. One of the reasons for choosing this data set is that the data set is standard. Another reason is that it is difficult to get another data set which contains so rich a variety of attacks.

Feature Extraction: For each network connection in the data set, the following three key groups of features for detecting intrusions will be extracted.

- **Basic features:** This group summarizes all the features that can be extracted from a TCP/IP connection. Some of the basic features in the KDDCup99 data sets are protocol type, service, src_bytes and dst_bytes.
- **Content features:** These features are purely based on the contents in the data portion of the data packet.
- **Traffic features:** This group comprises features that are computed with respect to a 2 Sec. time window and it is divided into two groups: same host features and same service features. Some of the traffic features are counted, rerror_rate, rerror_rate and srv_serror_rate.

Instance Labeling: After extracting KDDCup99 features from each record, the instances are labeled based on the characteristics of traffic as Normal, Dos, Probe, R2L and U2R.

Pre Processing

The data set will be preprocessed so that it may be able to give it as an input to our proposed system. This data set consists of numeric and symbolic features and will be converted in numeric form so that it can be

given as inputs to our MLP network. Now this modified data set will be used as training and testing of the multi layer perceptron. Table 4.1 below shows the feature columns name and type of 10% KDDCup 99 dataset.Table 0.1:KDD feature columns name and type [9]

The following tables represent the data feature columns before and after transformation.

Table 1Feature Column Before Transformation

0,tcp,http,SF,181,5450,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,8,8,0.00,0.00,0.00,0.00,1.00,0.00,0.00,9,9,1.00,0.00, 0.11,0.00,0.00,0.00,0.00,0.00,normal.

Table 2 Feature Column After Transformation

0,1,20,10,181,5450,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,8,8, ,0.00,0.00,0.00,0.00,1.00,0.00,0.00,9,9,1.00,0.00,0.11, 0.00,0.00,0.00,0.00,0.00,1,0,0,0,0.

Training and Testing of MLP

The input dataset is divided into 3 subsets. The first subset is the training set, which is used for computing the gradient and updating the network weights and biases. The second subset is the validation set. The error on the validation set is monitored during the training process. The validation error normally decreases during the initial phase of training, as does the training set error. However, when the network begins to over-fit the data, the error on the validation set typically begins to rise. When the validation error increases for a specified number of iterations (net.trainParam.max_fail), the training is stopped, and the weights and biases at the minimum of the validation error are returned. The test set error is not used during training, but it is used to compare different models (MathWorks Matlab Help, 2013).

In this thesis, 80% data from the input dataset are used for training, 10% for validation and 10% for testing of the MLP to analyze the performance of various back propagation algorithms.

5. Performance Parameters

Mean Square Error, Total CPU Time of Converge and Accuracy will be the performance parameters to compare various back propagation algorithms.

Following parameters will be calculated while training and testing of MLP.

- **True Positive (TP):** Situation in which a signature is fired properly when an attack is detected and an alarm is generated.
- **False Positive (FP):** Situation in which normal traffic causes the signature to raise an alarm.
- **True Negative (TN):** Situation in which normal traffic does not cause the signature to raise an alarm.
- **False Negative (FN):** Situation in which a signature is not fired when an attack is detected.
- **Attack Detection Rate (ADR)**: The detection rate is defined as the number of intrusion instances detected by the system (True Positive) divided by the total number of intrusion instances present in the test set.

Attack Detection Rate (ADR) = **(**Total detected attacks / Total attacks) * 100 %

- **False Alarm Rate (FAR):** It is the ratio between the total number of misclassified instances and the total number of normal connections present in the data set.
  False Alarm Rate (FAR) = **(**Total misclassified instances / Total normal instances) * 100 %
- **Recall Rate:** Recall rate measures the proportion of actual positives which are correctly identified.

  Recall Rate = TP/ (TP + FN)
- **Precision Rate:** Precision rate is the ratio of true positives to combined true and false positives.

  Precision Rate = TP/ (TP + FP)

6. Simulation Result

6.1 Determining Hidden Layer Neurons in Scale Conjugate Gradient (SCG):

The Multilayer Perception is trained to find the number of hidden layer neurons using the following parameters:

Number of input data = 494021

Number of input layer neurons = 41

Number of output layer neurons = 5

Change in weight for second derivative approximation (**σ**) = 5.0e-5

Parameter for regulating the indefiniteness of the Hessian(**λ**)=5.0e-7

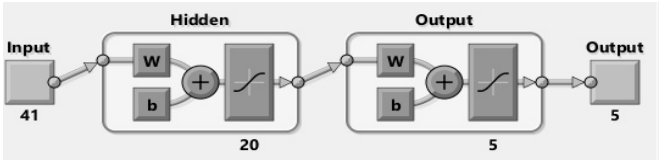

Fig 6.1 MLP Architecture of Back Propagation of 20 hidden neuron layer.

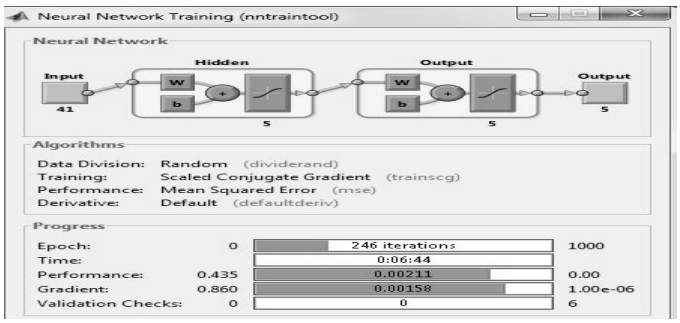

Fig 6.2 Performance of MLP with 5 neurons in hidden layer.

6.2 Performance Assessment of Scale Conjugate Gradient (SCG) Back Propagation Algorithms

Simulation is done to analyze the performance of Scaled Conjugate Gradient.The Multilayer Perceptron was trained with SCG algorithm by using following parameters.

Change in weight for second derivative approximation (**σ**) = 5.0e-5

Parameter for regulating the indefiniteness of the Hessian (**λ**) = 5.0e-7

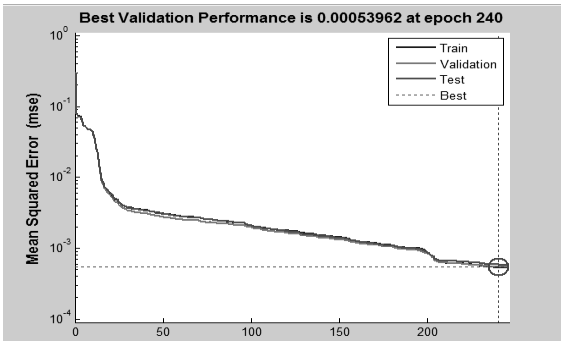

Fig. 6.3 Performance of SCG Algorithm

Table 6.1 Evaluation Results for each Attack Classes (SCG)

Attack	**TP**	**FP**	**FN**	**Recall**	**Precision**
DoS	391407	35	42	99.99%	99.99%
U2R	0	0	32	0%	0%
R2L	915	106	189	82.88%	89.61%
Probe	3898	30	200	95.12%	99.23%
Total	396220	171	463	98.88%	99.95%

6.3 Determining Hidden Layer Neurons in Self Organizing Map

The Multilayer Perception is trained to find the number of hidden layer neurons using the following parameters:

Number of input data = 14020

Number of input layer neurons = 41

Number of output layer neurons = 5 and 10

Simulation is done to analyze the performance of Self Organizing Map in terms of different number of hidden layer, epoch and iteration time required.

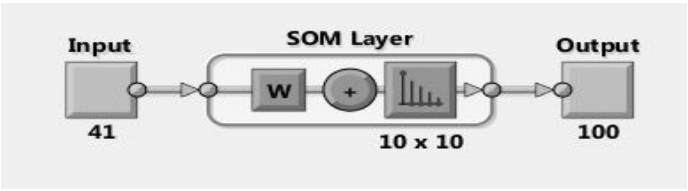

Fig 6.4 SOM Network of 10 hidden neuron layer

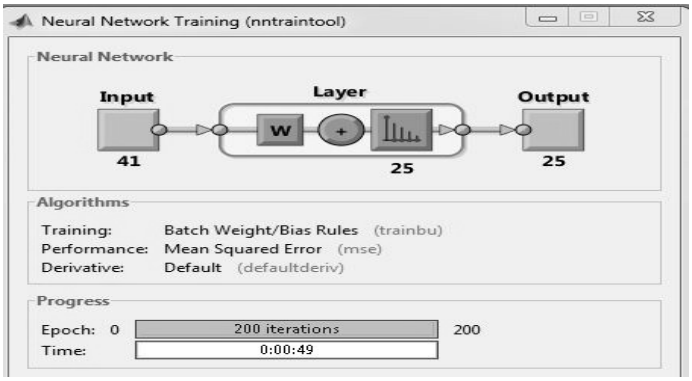

Fig 6.5 Performance of SOM with 5 neurons in hidden layer.

7. Conclusion

At present, security inside the network communication is of a major concern. Intrusion detection system tries to identify security attacks of intruders by investigating several data records observed in processes on the network. From simulation result using Matlab tool and java programming code we suggest that the performance like iteration completion time of SOM (Self Organizing Map is far better than BP (Back Propagation) algorithm.

References

1. S. Mukkamala, G. Janoski and A. Sung, *"Intrusion detection using neural networks and support vector machines"*, Proceedings of International Joint Conference on Neural Networks (IJCNN '02), Vol. 2, Pp. 1702–1707, 2002.
2. Snehal A. Mulay, P.R. Devale and G.V. Garje, *"Intrusion Detection System using Support Vector Machine and Decision Tree"*, International Journal of Computer Applications, Vol. 3, No. 3, Pp. 40-43, 2010.
3. D. Anderson, T. Frivold and A. Valdes, *"Next generation intrusion detection expert system (NIDES): a summary"*, Technical Report SRI-CSL-95-07. Computer Science Laboratory, SRI International, Menlo Park, CA, 1995.
4. S. Axelsson, *"Research in intrusion detection systems: a survey"*, Technical Report TR 98-17 (revised in 1999). Chalmers University of Technology, Goteborg, Sweden, 1999.
5. S. Freeman, A. Bivens, J. Branch and B. Szymanski, *"Host-based intrusion detection using user signatures"*, Proceedings of the Research Conference. RPI, Troy, NY, 2002.
6. K. Ilgun, R.A. Kemmerer and P.A. Porras, *"State transition analysis: A rule-based intrusion detection approach"*, IEEE Trans. Software Eng, Vol. 21, No. 3, Pp. 181–199, 1995.
7. D. Marchette, *"A statistical method for profiling network traffic"*, Proceedings of the First USENIX Workshop on Intrusion Detection and Network Monitoring, Santa Clara, CA, Pp. 119–128, 1999.
8. Chan, L. W. and Fall side, F., "*An adaptive training algorithm for back propagation networks"*, Computer Speech and Language, Vol. 2, page 205-218,1987
9. KDD Cup 1999 dataset. Available on: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html