TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
CENTRAL CAMPUS, PULCHOWK

THESIS NO: 069MSCS654

Intrusion Detection System Using Back Propagation Algorithm And Compare Its Performance With Self Organizing Map

BY
Bisho Raj Kaphale

A THESIS
SUBMITTED TO THE DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE IN COMPUTER SYSTEM AND KNOWLWDGE ENGINEERING

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING
LALITPUR, NEPAL
NOVEMBER, 2014

Intrusion Detection System Using Back Propagation Algorithm And Compare Its performance With Self Organizing Map

By
Bisho Raj Kaphale
069/MSCS/654

Thesis Supervisor
Prof. Dr. Subarna Shakya
Department of Electronics and Computer Engineering
Institute of Engineering
Central Campus

A thesis submitted to the Department of Electronics and Computer Engineering in partial fulfillment of the requirements for the degree of Master of Science in Computer System and Knowledge Engineering

Department of Electronics and Computer Engineering
Institute of Engineering, Central Campus
Tribhuvan University
Lalitpur, Nepal

November, 2014

COPYRIGHT

TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

CENTRAL CAMPUS

DEPARTENT OF ELECTRONICS AND COMPUTER ENGINEERING

The undersigned certify that it has been read and recommended to the Department of Electronics and Computer Engineering for acceptance, a report for the thesis entitled "Intrusion Detection System Using Back Propagation Algorithm and Compare Its Performance With Self Organizing Map", submitted by Bisho Raj Kaphale in partial fulfillment of the requirement for the award of the degree of "Masters of Science in Computer Systems and Knowledge Engineering".

_____

Supervisor: Dr. Subarna Shakya
Professor
Department of Electronics and Computer Engineering
Institute of Engineering, Central Campus, Pulchowk

_____

External Examiner: Suresh Kumar Regmi
Managing Director
Professional Computer System Pvt. Ltd

_____

Committee Chairperson, Dr. Shashidhar Ram Joshi
Professor
Department of Electronics and Computer Engineering
Institute of Engineering, Central Campus, Pulchowk

_____

Date

DEPARTMENTAL ACCEPTANCE

The thesis entitled “Intrusion Detection System Using Back Propagation Algorithm and Compare Its Performance With Self Organizing Map” submitted by Bisho Raj Kaphale in partial fulfillment of the requirement for the award of the degree of “Master of Science in Computer System and Knowledge Engineering” has been accepted as a bonafide record of work independently carried out by him in the department.

_____

Dr. Dibakar Raj Pant

Assistant Professor, Head of the Department

Department of Electronics and Computer Engineering,

Central Campus, Pulchowk

Institute of Engineering,

Tribhuvan University, Nepal.

ACKNOWLEDGEMENT

I have taken effort in this thesis. However, it would not have possible without the kind support and help of many individuals and Institute of Engineering, Pulchowk Campus. I would like to extend my sincere thanks to all of them.

I am very much thankful to the Department of Electronics and Computer Engineering, Institute of Engineering for accepting my thesis on “Intrusion detection system using Backpropagation algorithm and compare its performance with Self Organizing Map”. I would like to express my special gratitude and thanks to my supervisor Prof. Dr. Subarna Shakya for his support over the whole thesis. His advices on technical matters are invaluable, and his guidance is very critical for the successful of my thesis.

Furthermore I would like to acknowledge with much appreciation the crucial role of our Master’s degree program coordinator, Dr. Sanjeeb Prasad Panday and Prof Dr. Sashidhar Ram Joshi . I express my heart-felt gratitude for providing me with all the essential co-operation, valuable suggestions for choosing the project topic.

Finally, I would like to express my heartfelt thanks to my Parents, brothers Pushkar Kafle and Bhumesh Kafle and my friends Sonkashi Singh, Suman Shrma, Sarad Chandra Joshi, Shankar Gangaju, Madan Neupane who have always encouraged and supported me.

ABSTRACT

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. The growing network intrusions have put companies and organizations at a much greater risk of loss. This thesis proposes a new learning methodology towards developing a novel Intrusion Detection System (IDS) by Back Propagation Neural Networks (BPN) and Self Organizing Map (SOM) and compares 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 resemble either an attack or the normal behavior. There are several existing techniques that provide more security to the network, but most of these techniques are static. This thesis tests the proposed method by a benchmark intrusion dataset to verify its feasibility and effectiveness. Results show that choosing good hidden layers network and input data will not only have impact on the performance, but also on the overall execution efficiency. The proposed method can significantly reduce the training time and epoch time. It provides a powerful tool to help supervisors analyze, model and understand the complex attack behavior of electronic crime. The proposed methodology implemented in sampled data from KddCup99 data set so that intrusion detection attacks database is standard for the evaluation of intrusion detection systems.

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

TABLE OF CONTENTS

COPYRIGHT © ii
DEPARTMENTAL ACCEPTANCE iv
ACKNOWLEDGEMENT v
ABSTRACT vi
LIST OF FIGURES ix
LIST OF TABLES x
LIST OF ABBREVIATION xi
1. INTRODUCTION 1
1.1 Background 1
1.2. Problem Statement 4
1.3. Objective 5
1.4. Organization of Report 5
2. LITERATURE REVIEW 6
3. RESEARCH METHODOLOGY 8
3.1 Back Propagation Algorithm 8
3.2 Self Organizing Map 11
3.2.1. Mapping Precision 13
3.2.2. Topology Preservation 13
3.2.3 SOM Implementation to Intrusion Detection System 13
4. DATA ANALYSIS 17
4.1 Input Dataset Analysis 17
4.2 Pre Processing 18
4.3 Determining Architecture of MLP 24
4.4 Training and Testing of MLP 24
4.5 Back Propagation Algorithm 25
4.6 Performance Parameters 26
4.7 Tool 27
5. SIMULATION RESULTS AND DISCUSSIONS 29
5.1 Determining Hidden Layer Neurons 29
5.2 Intrusion Detection 30

5.3 Determining Hidden Layer Neurons in Scale Conjugate Gradient (SCG):34
5.2 Performance Assessment of Back Propagation Algorithms 36
5.2.1 Scale Conjugate Gradient (SCG):..... 36
5.2.2 Determining Hidden Layer Neurons in Self Organizing Map 40
6. CONCLUSIONS AND RECOMMENDATIONS 43
6.1 Conclusions..... 43
6.2 Future Recommendations 43
7. References 44
8. Bibliography 46
Appendix A 47
Appendix B 51

LIST OF FIGURES

Figure 1.1:The Intrusion Detection System and External/Internal Network Intrusion Attacks 2
Figure 1.2:Multilayer Perceptron 3
Figure 1.3: Bipolar Sigmoid Function 4
Figure 3.1:A Back Propagation Training Set. 8
Figure 3.2: Applying a Training Pair to a Network 9
Figure 3.3: A Single Connection Learning in a Back Propagation network. 9
Figure 3.4: General SOM topology 12
Figure 3.5:Structure of an Automated User Behavior Anomaly Detection System .. 14
Figure 3.6: Form of the designed SOM architecture 16
Figure 4.1: Block diagram of the proposed system 17
Figure 4.2: Backpropagation Algorithm Diagram 25
Figure 5.1: MLP Architecture of the System 30
Figure 5.2: MLP Architecture of Back Propagation 34
Figure 5.3: Performance of MLP with 10 neurons in hidden layer. 35
Figure 5.4: Performance of MLP with 5 neurons in hidden layer. 35
Figure 5.5: Performance of SCG Algorithm 36
Figure 5.6: Confusion matrix of Scaled Conjugate Gradient algorithm 37
Figure 5.7: Relationship of Attack Types versus True Positive 38
Figure 5.8: Relationship of Attack Types versus False Positive 38
Figure 5.9: Relationship of Attack types Versus False Negative 39
Figure 5.10: Relationship of Attack Types Versus Recall Rate 39
Figure 5.11: Relationship of Attack Types Versus Precision Rate 40
Figure 5.12: SOM Network Layer of Size 10. 41
Figure 5.13: Performance of SOM with 5 neurons in hidden layer. 41
Figure 5.14: Performance of SOM with 10 neurons in hidden layer. 42

LIST OF TABLES

Table 4.1:KDD feature columns name and type 19
Table 4.2:Protocol Type 20
Table 4.3: Service Type 20
Table 4.4: Flag Types 22
Table 4.5: Attacks Classification 22
Table 4.6: Label Transformation 23
Table 4.7: Feature Column Before Transformation 23
Table 4.8: Feature Column After Transformation 23
Table 4.9: Selection of Number of Neurons in Hidden Layer 24

LIST OF ABBREVIATION

ANN	Artificial Neural Network
DARPA	Defense Advance Research Project Agency
DOS	Denial of Service
FN	False Negative
FP	False Positive
IDS	Intrusion Detection System
IEDS	Intrusion Detection Expert System
MIT	Massachusetts Institute of Technology
MLP	Multi-Layer Perception
NIDS	Network based IDS
R2L	Remote to Local
TN	True Negative
TP	True Positive
U2R	User to Root

1. INTRODUCTION

1.1 Background

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 was hacked because of 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. Figure 1.1 illustrates the intrusion detection system and external/internal network intrusion attacks. 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. By modifying these interconnections, the network is able to adapt to the desired outputs. The ability of high tolerance for learning-by-example makes neural networks flexible and powerful in IDS. However, the time required to induce the model from a large dataset is long. It can accurately predict probable attack behavior in IDS.

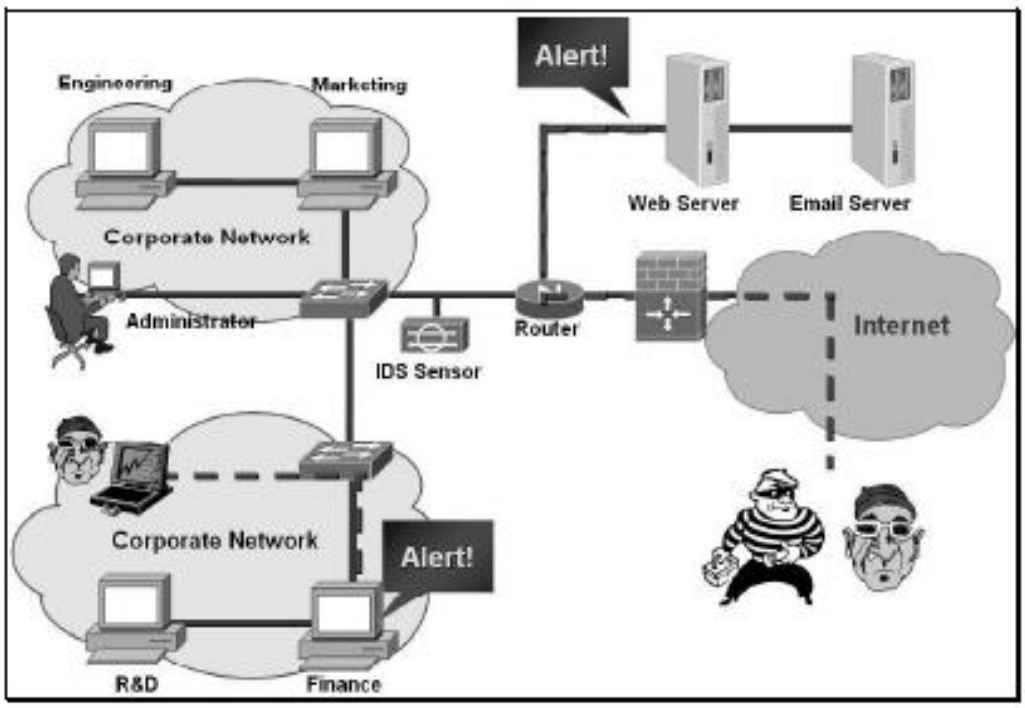


Figure 1.1:The Intrusion Detection System and External/Internal Network Intrusion Attacks[1].

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].

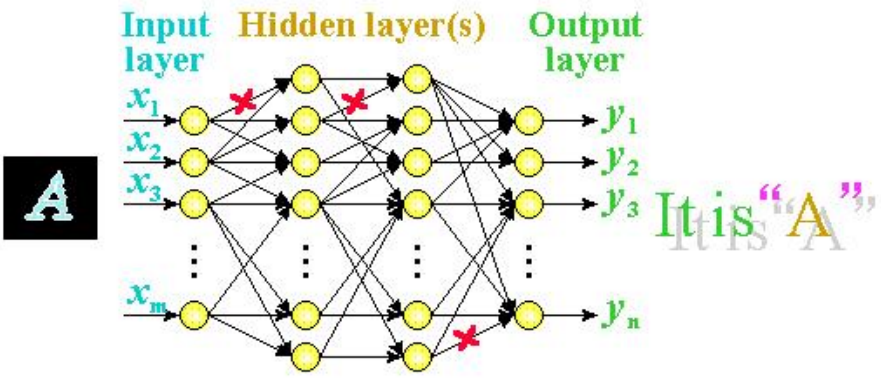


Figure 1.2:Multilayer Perceptron[2]

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: $f'(x) = 0.5 * [1 + f(x)] * [1 – f(x)]$

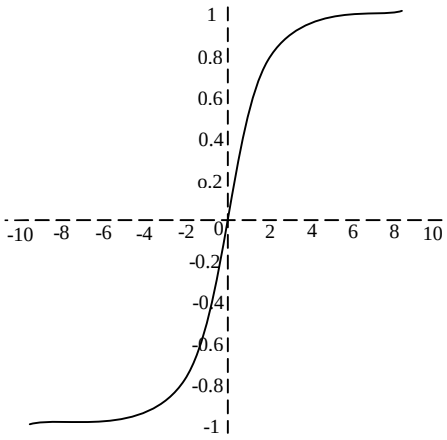


Figure 1.3: Bipolar Sigmoid Function

In this research, multilayer perception is trained with various Backpropagation algorithms. Based on the evaluation result, the proposed research is able to suggest the best model for network based attack detection.

1.2. Problem Statement

IDS is Rule Based Monitoring and Controlling System, therefore, selection of algorithm used to define standard rule base is a major challenge. The selection of improper algorithm and model can maximize the occurrence of false alarm rate, high resource consumption, and low intrusion detection rate and may result inefficiency to entire system and may even lead to security vulnerabilities. The proper selection of classifier algorithm leads to increase in efficiency of IDS being implemented.

1.3. Objective

Objectives of the proposed research are as follows:

- To detect intrusion using multilayer perception with Back Propagation and Self Organizing Map.
- To analyze the performance of Back Propagation algorithms and Self Organizing Map and suggest the efficient model for network intrusion detection based on the evaluation result.

1.4. Organization of Report

The report contains six chapters. Chapter 1 provides in detail about the background, statements of problems, objectives of research. Chapter 2 explains about literature review of intrusion detection system. Chapter 3 explains about different research methodology on intrusion detection using Backpropagation system and Self Organizing Map. Chapter 4 explains in detail about the input data analysis, hidden layer network architecture and performance parameters. Chapter 5 explains about the analysis of simulation results and discussions. Chapter 6 provides brief conclusions of simulation results and suggests the most efficient model for intrusion detection along with the future enhancements.

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].

In case of host-based attacks, the intruders aim at a particular machine and attempt to get access to privileged services or resources on that particular machine. Detection of these kinds of attacks typically uses routines that acquire system call data from an audit-process which monitors all system calls made with the support of each user. In case of network-based attacks, it is extremely complicated for legitimate users to use various network services by purposely occupying or disrupting network resources and services. 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

3.1 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.

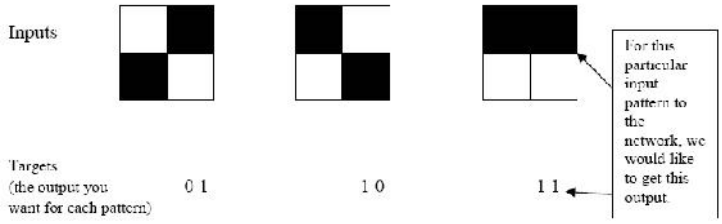


Figure 3.1:A Back Propagation Training Set[1].

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.

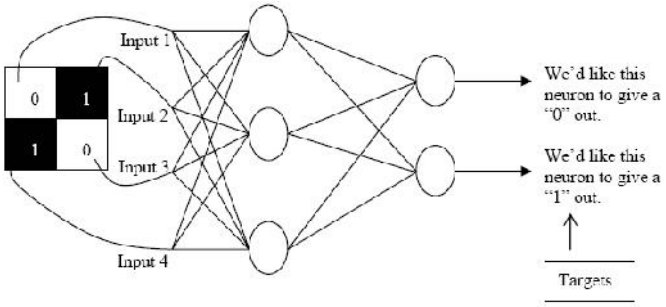


Figure 3.2: Applying a Training Pair to a Network[1]

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.

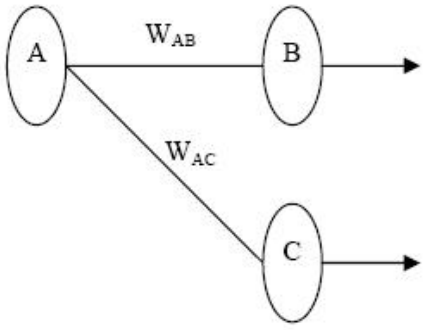


Figure 3.3: A Single Connection Learning in a Back Propagation network[1].

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. This may well have left some doubt in your mind about the operation, so let's clear that up by explicitly showing all the calculations for a full sized network with 2 inputs, 3 hidden layer neurons and 2 output neurons as shown in figure 3.3. W+ represents the new, recalculated weight, where as W (without the

superscript) represents the old weight. Reverse process is done in the same way. Hence the attackers are calculated.

3.2 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 Fig 3.4 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.

$M_i [M_1 \ldots\ldots\ldots M_n]^T$ ……(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:

$||x-m_c||=\min \{||x-m_i||\}$……(3.2)

2. After the BMU(Best Matching Unit) 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:

$m_i(t+1) = \{ \ m_i(t) + \alpha(t) [x(t) – m_i(t)] , I € N_c(t)$

$m_i(t+1)=\{m_i(t),I € N_c(t)$ …….(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.

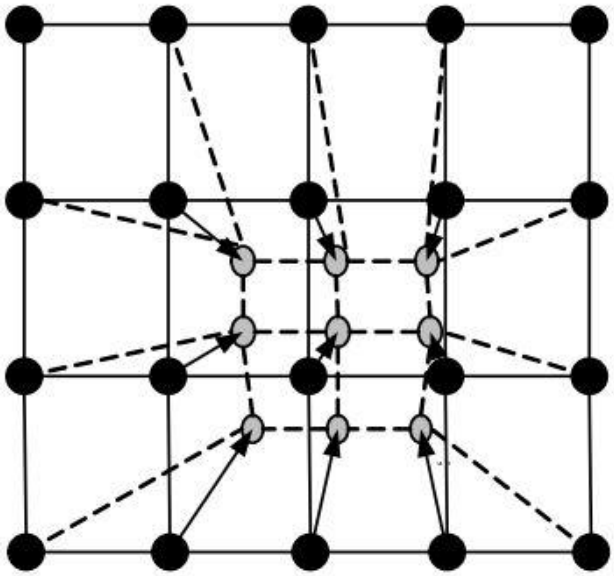


Figure 3.4: General SOM topology[2]

After the training is over, the map should be topologically ordered. This means that n topologically close input data vectors map to n adjacent map neurons or even to the same single neuron.

3.2.1. 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:

$E_q = 1/N \sum_{i=1}^{N} || x_i + m_c ||$……..……..(3.4)

3.2.2. 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:

$E_q = 1/N \sum_{i=1}^{N} u(x_x)$……..(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.

3.2.3 SOM Implementation to Intrusion Detection System

The goal of the proposed architecture is to investigate effectiveness of application a neural network SOM figure.3.6 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.

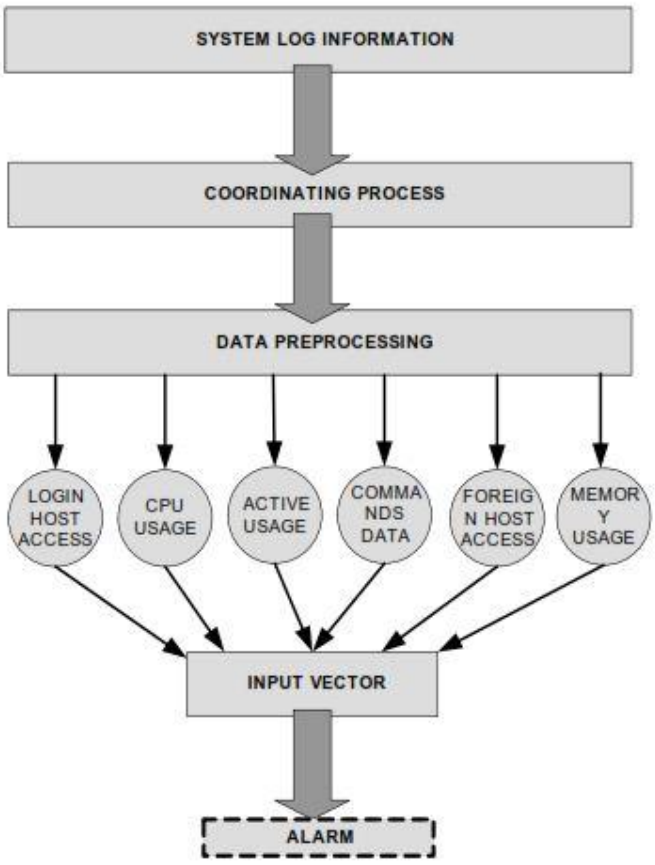


Figure 3.5:Structure of an Automated User Behavior Anomaly Detection System[3]

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 Fig. 3.5, separating selected data from audit data. The input to the neural network Fig. 3.6 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 Inforatics in Košice . This normalization is more suitable for implementation in proposed SOM network. The architecture uses normalization given by:

$$nv[i] = \frac{v[i]}{\sqrt{\sum_k v[k]^2}} \quad \text{.....(3.6)}$$

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.

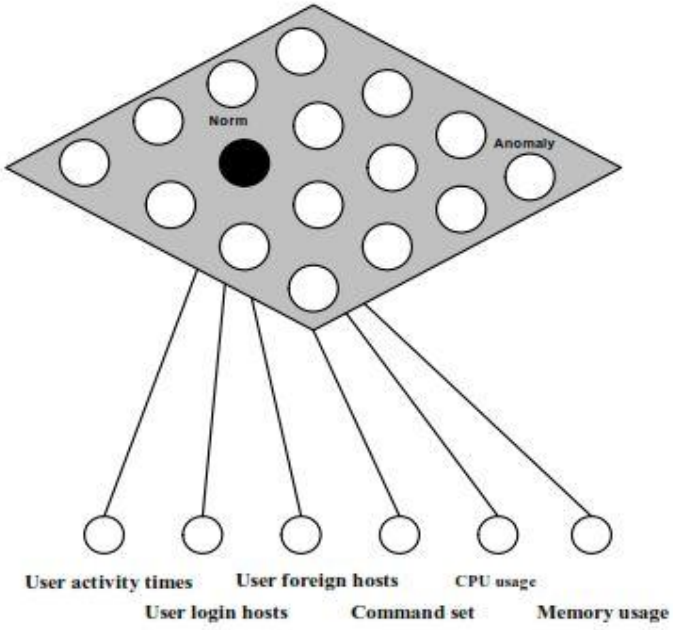


Figure 3.6: Form of the designed SOM architecture[2]

4. DATA ANALYSIS

Methodology of the proposed system is shown in the Figure 4.1 below.

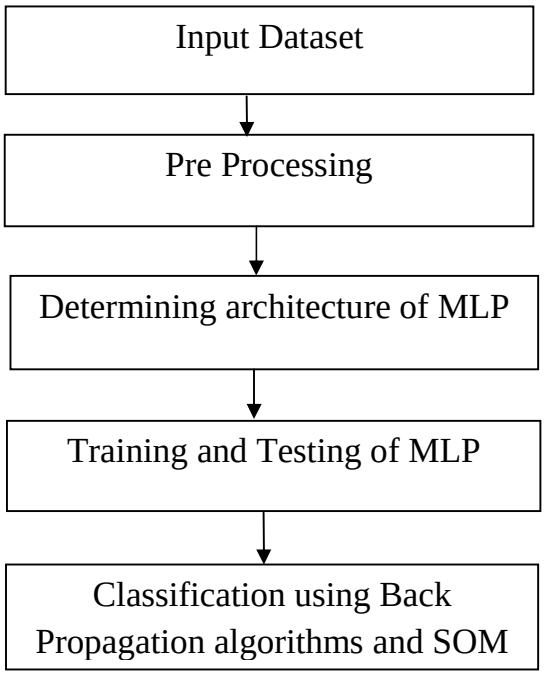


Figure 4.1: Block diagram of the proposed system

4.1 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 [10]. 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.

4.2 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 4.1:KDD feature columns name and type [9]

	Feature Name	Feature Type		Feature Name	Feature Type
1	duration	continuous.	22	is_guest_login	discrete.
2	protocol_type	symbolic.	23	count	continuous.
3	service	symbolic.	24	srv_count	continuous.
4	flag	symbolic.	25	serror_rate	continuous.
5	src_bytes	continuous.	26	srv_serror_rate	continuous.
6	dst_bytes	continuous.	27	rerror_rate	continuous.
7	land	discrete.	28	srv_rerror_rate	continuous.
8	wrong_fragment	continuous.	29	same_srv_rate	continuous.
9	urgent	continuous.	30	diff_srv_rate	continuous.
10	hot	continuous.	31	srv_diff_host_rate	continuous.
11	num_failed_logins	continuous.	32	dst_host_count	continuous.
12	logged_in	discrete.	33	dst_host_srv_count	continuous.
13	num_compromised	continuous.	34	dst host same srv rate	continuous.
14	root_shell	continuous.	35	dst_host_diff_srv_rate	continuous.
15	su_attempted	continuous.	36	dst_host_same_src_port_rate	continuous.
16	num_root	continuous.	37	dst_host_srv_diff_host_rate	continuous.
17	num_file_creations	continuous.	38	dst_host_serror_rate	continuous.
18	num_shells	continuous.	39	dst_host_srv_serror rate	continuous.
19	num_access_files	continuous.	40	dst_host_rerror_rate	continuous.
20	num_outbound_cmds	continuous.	41	dst_host_srv_rerror_rate	continuous.
21	is_host_login	discrete.	42	Label	symbolic.

Symbolic columns which are protocol_type, service, flag and label are transformed to numeric values using transformation tables given below. The protocol_type column has 3 protocol values: TCP, UDP and ICMP. Table 4.2 demonstrates the transformation table for protocol_type.

Table 4.2:Protocol Type

Protocol_type	No.
TCP	1
UDP	2
ICMP	3

The service column values are transformed to numeric values as shown in Table4.3.

Table 4.3: Service Type

Service	No.	Service	No.
Auth	1	netbios_ssn	34
Bgp	2	Netstat	35
Courier	3	Nnsp	36
csnet_ns	4	nntp	37
Ctf	5	ntp_u	38
Daytime	6	Other	39
Discard	7	pm_dump	40
Domain	8	pop_2	41
domain_u	9	pop_3	42
Echo	10	Printer	43
eco_i	11	Private	44
ecr_i	12	red_i	45
Efs	13	remote_job	46
Exec	14	Rje	47

Service	No.	Service	No.
Finger	15	Shell	48
ftp	16	Smtp	49
ftp_data	17	sql_net	50
Gopher	18	Ssh	51
Hostnames	19	Sunrpc	52
http	20	Supdup	53
http_443	21	Systat	54
imap4	22	telnet	55
Irc	23	tftp_u	56
iso_tsap	24	tim_i	57
Klogin	25	Time	58
Kshell	26	urh_i	59
Ldap	27	urp_i	60
Link	28	Uucp	61
Login	29	uucp_path	62
Mtp	30	Vmnet	63
Name	31	Whois	64
netbios_dgm	32	x11	65
netbios_ns	33	z39_50	66

The flag column values are transformed to numeric values as shown in Table 4.4.

Table 4.4: Flag Types

Flag	No.	Flag	No.
Oth	1	S1	7
REJ	2	S2	8
RSTO	3	S3	9
RSTOS0	4	SF	10
RSTR	5	SH	11
S0	6		

The Label column has normal and different kinds of sub attack values. Sub attack values are classified as shown in Table 4.5 and then the normal and attack values are transformed into numeric as shown in Table 6 below.

Table 4.5: Attacks Classification

Main Attack	DOS	U2R	R2L	Prob
Sub Attack	apache2 back land mailbomb neptune pod processtable smurf teardrop upstorm	buffer_overflow load module perl ps rootkit xterm	ftp_write guess_passwd imap mscam warezclient warezmaster xclock xsnoop	Ipsweep nmap portsweep satan

Table 4.6: Label Transformation

Label	Column1	Column2	Column3	Column4	Column5
Normal	1	0	0	0	0
DoS	0	1	0	0	0
U2R	0	0	1	0	0
R2L	0	0	0	1	0
Prob	0	0	0	0	1

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

Table 4.7: Feature 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 4.8: 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.

4.3 Determining Architecture of MLP

There is no certain mathematical approach for obtaining the optimum number of hidden layers and their neurons. In this research, 10, 16, 23 and 30 layered MLP with 41 neurons in the input layer and 5 neurons in the output layer is used. The numbers of nodes in hidden layer are chosen by hit and trial method.

Table 4.9 below shows the performance of multilayer perceptron with different numbers of hidden layer neurons. The best performance is observed with 30 neurons in the hidden layer.

Table 4.9: Selection of Number of Neurons in Hidden Layer

Hidden Layer	No. of Neurons	Performance
H1	10	0.00175
H1	16	0.000346
H1	23	0.00101
H1	30	0.000286

4.4 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, 2012).

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 backpropagation algorithms. The results are shown in chapter 4.

4.5 Back Propagation Algorithm

The gradient descent, gradient descent with momentum and resilient Back Propagation algorithms will be used for training of MLP and performance of these algorithms will be compared.

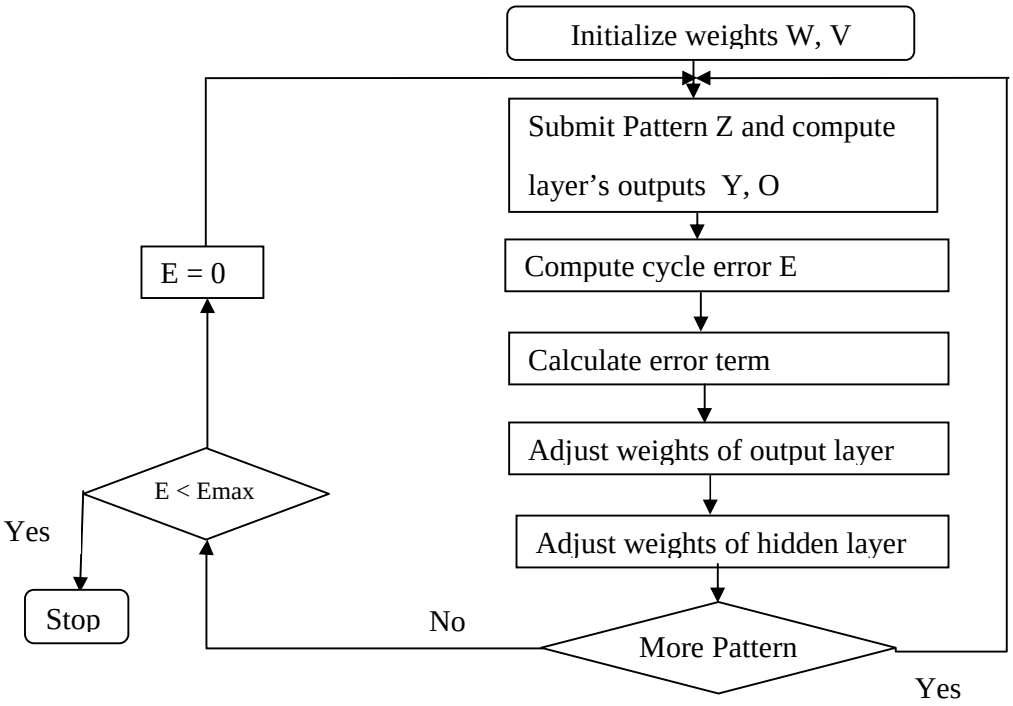


Figure 4.2: Backpropagation Algorithm Diagram

Figure 4.2 above shows the flowchart of Back Propagation algorithm. The Back Propagation learning algorithm can be divided into two phases: propagation and weight update.

Phase 1: Propagation

Each propagation involves the following steps:

1. Forward propagation of a training pattern's input through the neural network in order to generate the propagation's output activations.
2. Backward propagation of the propagation's output activations through the neural network using the training pattern target in order to generate the deltas of all output and hidden neurons.

Phase 2: Weight update

For each weight-synapse follow the following steps:

1. Multiply its output delta and input activation to get the gradient of the weight.
2. Subtract a ratio (percentage) of the gradient from the weight.

This ratio (percentage) influences the speed and quality of learning; it is called the learning rate. The greater the ratio, the faster the neuron trains; the lower the ratio, the more accurate the training is. The sign of the gradient of a weight indicates where the error is increasing; this is why the weight must be updated in the opposite direction.

Repeat phase 1 and 2 until the performance of the network is satisfactory.

4.6 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)

4.7 Tool

MATLAB 2012:

MATLAB (Matrix Laboratory) is a programming environment for algorithm development, data analysis, visualization, and numerical computation. MATLAB can solve technical computing problems faster than with traditional programming languages, such as C, C++, and FORTRAN. MATLAB can be used in a wide range of applications, including signal and image processing, communications, control design, test and measurement, financial modeling and analysis, and neural networks. For a million engineers and scientists in industry and academia, MATLAB is the language of technical computing (MathWorks Matlab Help, 2012).

Neural Network Toolbox supports supervised learning with feed forward, radial basis, and dynamic networks. It also supports unsupervised learning with self-organizing maps and competitive layers. With the toolbox we can design, train, visualize, and simulate neural networks. Simulation is done using Neural Network toolbox in Matlab.

Notepad++:

Notepad++ is a text editor and source code editor for Windows. It differs from the built-in Windows text editor Notepad, is that Notepad++ supports tabbed editing, which allows working with multiple open files in a single window. Notepad++ opens large files significantly faster than Windows Notepad. Data preprocessing is done using Notepad++ tool.

5. SIMULATION RESULTS AND DISCUSSIONS

To access the effectiveness of the proposed intrusion detection approach, the following simulation is performed. Intel (R) Core™ i5 CPU M 430 @ 2.27GHz, having 4 GB of RAM is used. The operating system Microsoft Windows 7 Home Premium is used. Simulation is performed using Matlab2012 version R2012a Sun Feb 17 18:06:36 2013. KDD dataset containing 494021 sample data is used as input to the IDS. The input layer has 41 input neurons to describe 41 attributes in the KDD dataset. For the hidden layer, hit and trial method is used that is explained in section 5.1 below. Since the network traffic is grouped into five different classes of attacks, the output layer has 5 neurons.

5.1 Determining Hidden Layer Neurons

The Multilayer Perceptron is trained with scale congugate BP algorithm to find the proper number of hidden layer neurons using the following default parameters:

Increment to weight change = 1.2, Decrement to weight change = 0.5, Initial weight change = 0.07 and Maximum weight change = 50.

The simulation results are shown in an Appendix. Table 5.1 below shows the performance of MLP with different number of hidden layer neurons and Figure 5.1 below shows the required MLP architecture of the system.

Table 5.1: Performance of MLP with different Number of Hidden Layer Neurons

Hidden Layer	No. of Neurons	Performance
H1	10	0.00175
H1	16	0.000346
H1	23	0.00101
H1	30	0.000286

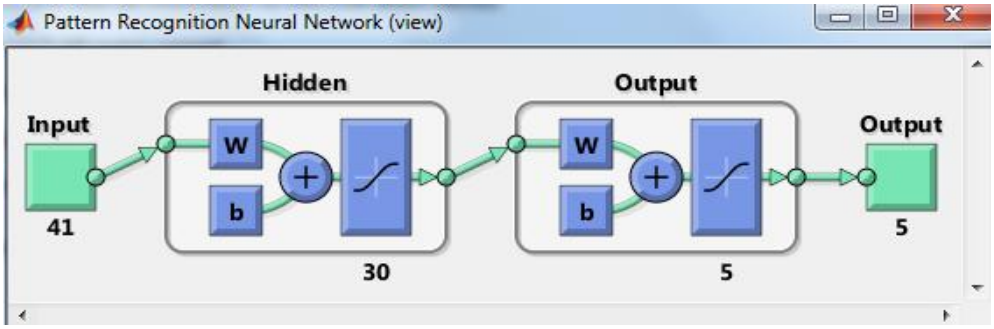
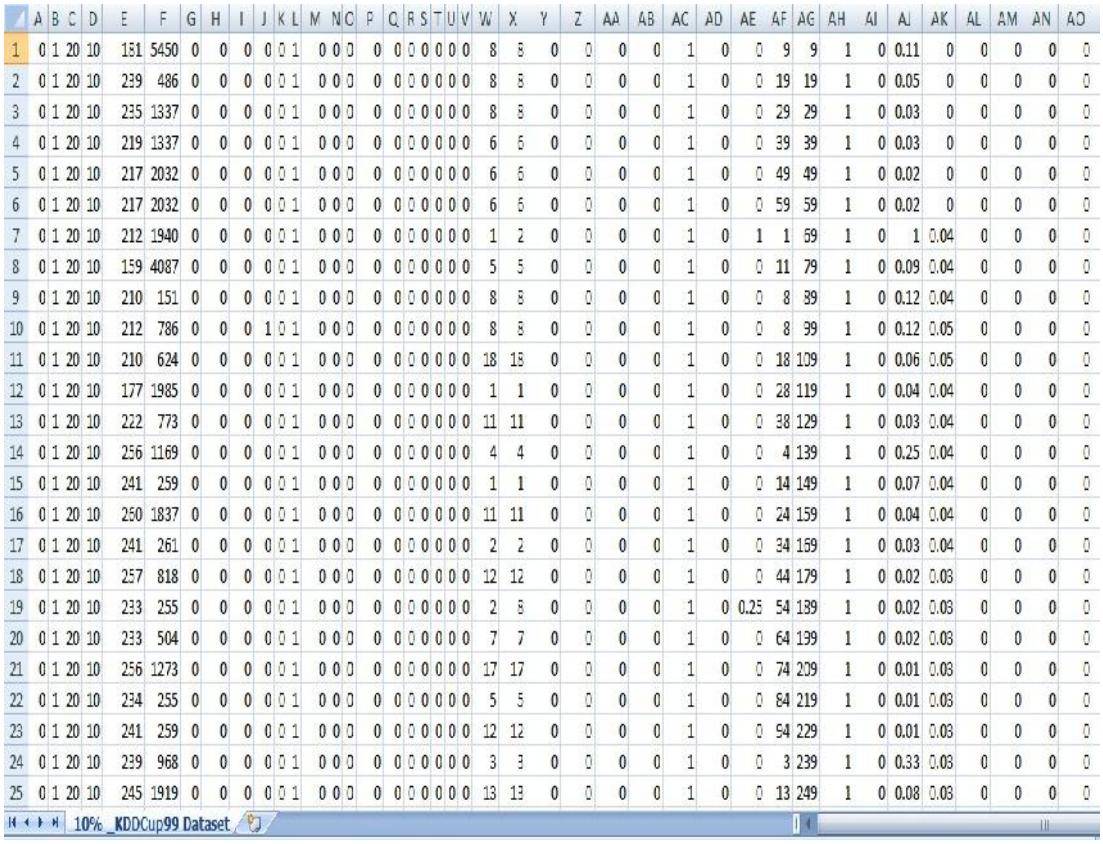


Figure 5.1: MLP Architecture of the System

5.2 Intrusion Detection

The input and target datasets are fed to the MLP and the output is observed. The portion (25 records) of input dataset and target dataset is shown below in the Table 5.2 and 5.3 respectively. The output of the most efficient model (Scale Conjugate BP algorithm) is shown below in the Table 5.4.

Table 5.2: Portion of Input Dataset



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL	AM	AN	AO
1	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	0	0	0	1	0	0	9	9	1	0	0.11	0	0	0	0	0
2	0	1	20	10	239	486	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	8	8	0	0	0	0	1	0	0	19	19	1	0	0.05	0	0	0	0	0
3	0	1	20	10	235	1337	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	8	8	0	0	0	0	1	0	0	29	29	1	0	0.03	0	0	0	0	0
4	0	1	20	10	219	1337	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	6	6	0	0	0	0	1	0	0	39	39	1	0	0.03	0	0	0	0	0
5	0	1	20	10	217	2032	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	6	6	0	0	0	0	1	0	0	49	49	1	0	0.02	0	0	0	0	0
6	0	1	20	10	217	2032	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	6	6	0	0	0	0	1	0	0	59	59	1	0	0.02	0	0	0	0	0
7	0	1	20	10	212	1940	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	2	0	0	0	0	1	0	1	1	69	1	0	1	0.04	0	0	0	0
8	0	1	20	10	159	4087	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	5	5	0	0	0	0	1	0	0	11	79	1	0	0.09	0.04	0	0	0	0
9	0	1	20	10	210	151	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	8	8	0	0	0	0	1	0	0	8	89	1	0	0.12	0.04	0	0	0	0
10	0	1	20	10	212	786	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	8	8	0	0	0	0	1	0	0	8	99	1	0	0.12	0.05	0	0	0	0
11	0	1	20	10	210	624	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	18	18	0	0	0	0	1	0	0	18	109	1	0	0.06	0.05	0	0	0	0
12	0	1	20	10	177	1985	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	0	28	119	1	0	0.04	0.04	0	0	0	0
13	0	1	20	10	222	773	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	11	11	0	0	0	0	1	0	0	38	129	1	0	0.03	0.04	0	0	0	0
14	0	1	20	10	256	1169	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	4	4	0	0	0	0	1	0	0	4	139	1	0	0.25	0.04	0	0	0	0
15	0	1	20	10	241	259	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	0	14	149	1	0	0.07	0.04	0	0	0	0
16	0	1	20	10	250	1837	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	11	11	0	0	0	0	1	0	0	24	159	1	0	0.04	0.04	0	0	0	0
17	0	1	20	10	241	261	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	2	2	0	0	0	0	1	0	0	34	159	1	0	0.03	0.04	0	0	0	0
18	0	1	20	10	257	818	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	12	12	0	0	0	0	1	0	0	44	179	1	0	0.02	0.03	0	0	0	0
19	0	1	20	10	233	255	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	2	8	0	0	0	0	1	0	0.25	54	139	1	0	0.02	0.03	0	0	0	0
20	0	1	20	10	233	504	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	7	7	0	0	0	0	1	0	0	64	139	1	0	0.02	0.03	0	0	0	0
21	0	1	20	10	256	1273	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	17	17	0	0	0	0	1	0	0	74	209	1	0	0.01	0.03	0	0	0	0
22	0	1	20	10	234	255	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	5	5	0	0	0	0	1	0	0	84	219	1	0	0.01	0.03	0	0	0	0
23	0	1	20	10	241	259	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	12	12	0	0	0	0	1	0	0	54	229	1	0	0.01	0.03	0	0	0	0
24	0	1	20	10	239	968	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	3	3	0	0	0	0	1	0	0	3	239	1	0	0.33	0.03	0	0	0	0
25	0	1	20	10	245	1919	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	13	13	0	0	0	0	1	0	0	13	249	1	0	0.08	0.03	0	0	0	0

10%_KDDCup99 Dataset

Table 5.3: Portion of Target Dataset

	A	B	C	D	E
1	1	0	0	0	0
2	1	0	0	0	0
3	1	0	0	0	0
4	1	0	0	0	0
5	1	0	0	0	0
6	1	0	0	0	0
7	1	0	0	0	0
8	1	0	0	0	0
9	1	0	0	0	0
10	1	0	0	0	0
11	1	0	0	0	0
12	1	0	0	0	0
13	1	0	0	0	0
14	1	0	0	0	0
15	1	0	0	0	0
16	1	0	0	0	0
17	1	0	0	0	0
18	1	0	0	0	0
19	1	0	0	0	0
20	1	0	0	0	0
21	1	0	0	0	0
22	1	0	0	0	0
23	1	0	0	0	0
24	1	0	0	0	0
25	1	0	0	0	0

10%_KDDCup99 Target Data

Table 5.4: Output Data

	A	B	C	D	E
1	0.87	2.75E-06	4.73E-05	0.00058	0
2	0.88	3.12E-06	4.58E-05	0.00058	0
3	0.89	3.32E-06	4.51E-05	0.00057	0
4	0.89	3.34E-06	4.50E-05	0.00057	0
5	0.89	3.49E-06	4.45E-05	0.00057	0
6	0.89	3.53E-06	4.44E-05	0.00057	0
7	0.85	2.33E-06	4.96E-05	0.00059	0
8	0.97	1.39E-05	3.11E-05	0.00052	0
9	0.98	1.88E-05	2.87E-05	0.00051	0
10	0.98	2.15E-05	2.77E-05	0.00051	0
11	0.99	7.32E-05	2.02E-05	0.00047	0
12	0.99	8.55E-05	1.94E-05	0.00046	0
13	1.00	1.09E-04	1.82E-05	0.00046	0
14	0.99	6.63E-05	2.07E-05	0.00047	0
15	1.00	3.85E-04	1.31E-05	0.00042	0
16	1.00	4.88E-04	1.23E-05	0.00041	0
17	1.00	5.06E-04	1.22E-05	0.00041	0
18	1.00	5.61E-04	1.19E-05	0.00041	0
19	1.00	7.66E-04	1.10E-05	0.00040	0
20	1.00	5.69E-04	1.19E-05	0.00041	0
21	1.00	6.17E-04	1.16E-05	0.00041	0
22	1.00	6.05E-04	1.17E-05	0.00041	0
23	1.00	6.25E-04	1.16E-05	0.00041	0
24	1.00	7.92E-04	1.09E-05	0.00040	0
25	1.00	9.80E-04	1.03E-05	0.00040	0

Output Data

By comparing the rows of Table 4.6 (Label Transformation) and Table 5.4 (Output Data), type of attack can be identified easily. For example in Table 5.4 above, if value of first column (A) is nearly equal to 1 and value of other columns is nearly equals to 0 that is normal type activity. If value of second column (B) is nearly equal to 1 and value of other columns is nearly equal to 0 that is DoS type attack and so on.

5.3 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

Above Table 4.9 shows the performance of MLP with different number of hidden layer neurons. The best performance is observed with 20 neurons in the hidden layer. The required MLP architecture is shown below in Figure 5.2.

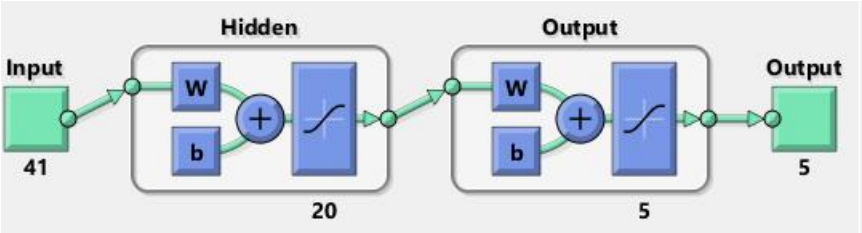


Figure 5.2: MLP Architecture of Back Propagation

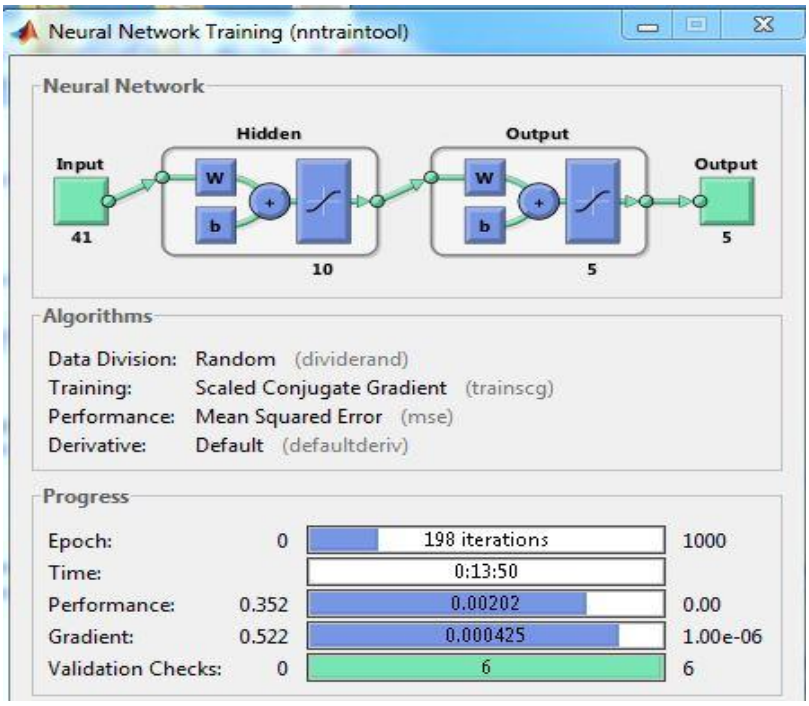


Figure 5.3: Performance of MLP with 10 neurons in hidden layer.

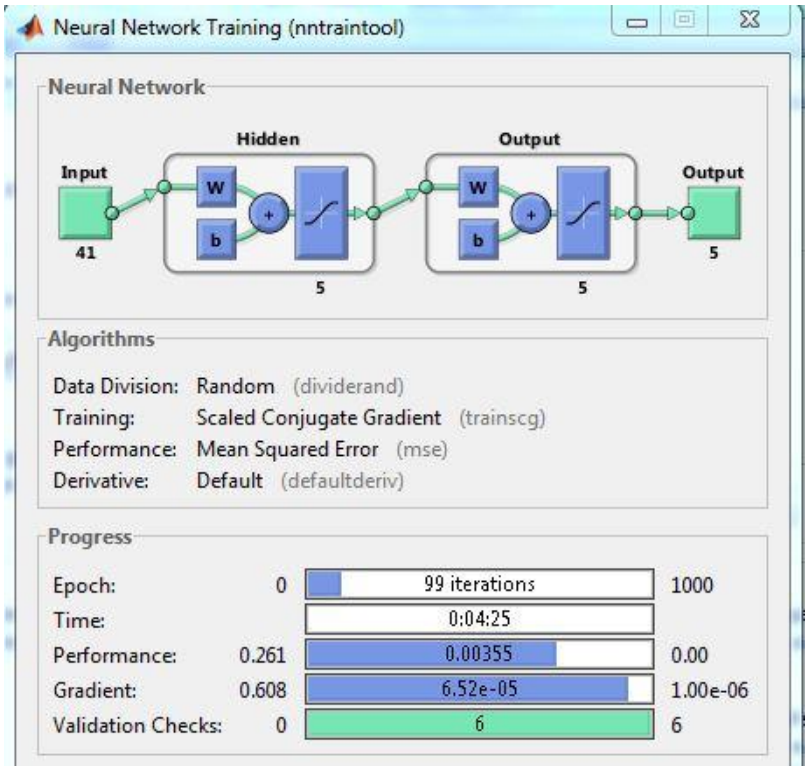


Figure 5.4: Performance of MLP with 5 neurons in hidden layer.

5.2 Performance Assessment of Back Propagation Algorithms

Simulation is done to analyze the performance of Scaled Conjugate Gradient.

5.2.1 Scale Conjugate Gradient (SCG):

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

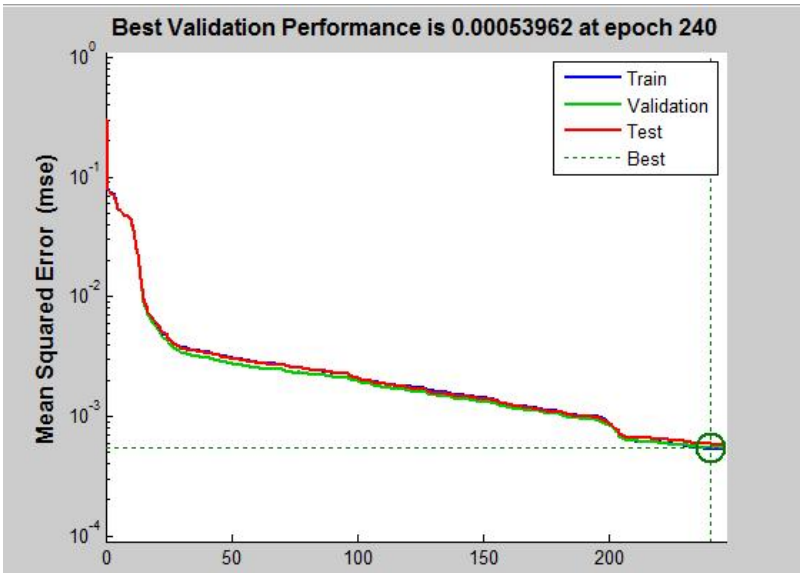


Figure 5.5: Performance of SCG Algorithm

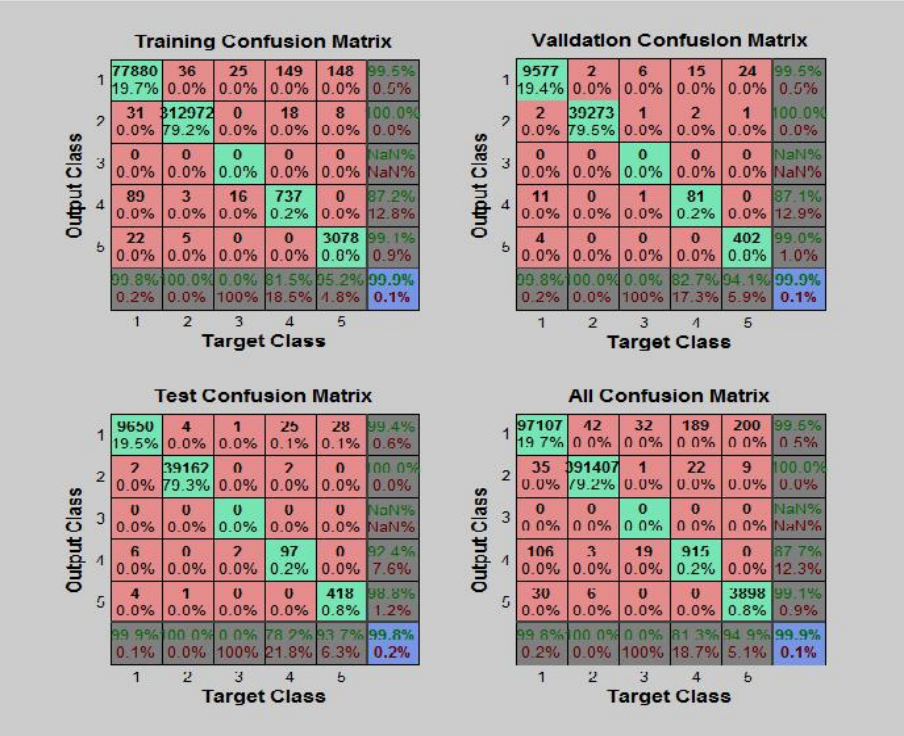


Figure 5.6: Confusion matrix of Scaled Conjugate Gradient algorithm

Table 5.5: 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%

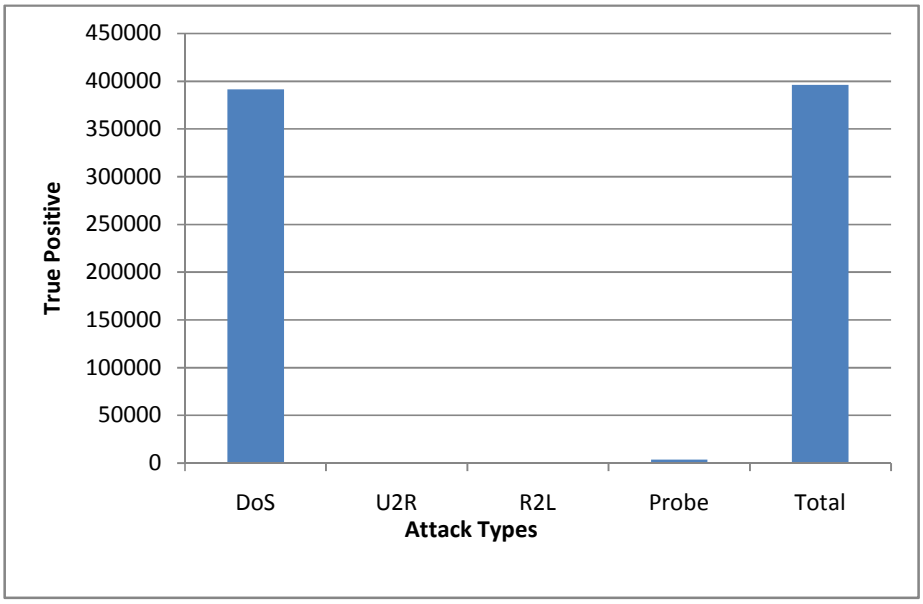


Figure 5.7: Relationship of Attack Types versus True Positive

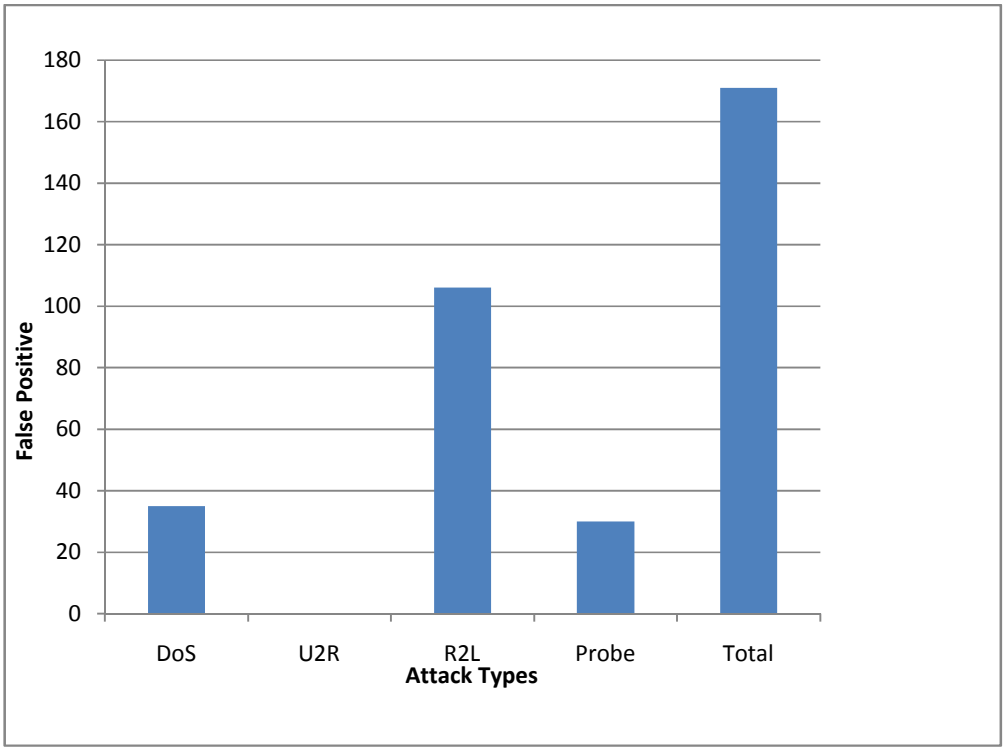


Figure 5.8: Relationship of Attack Types versus False Positive

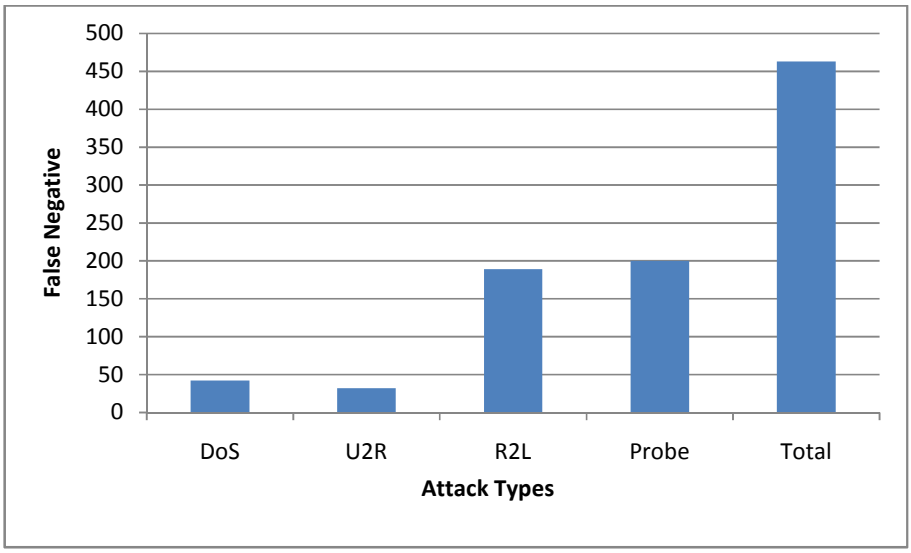


Figure 5.9: Relationship of Attack types Versus False Negative

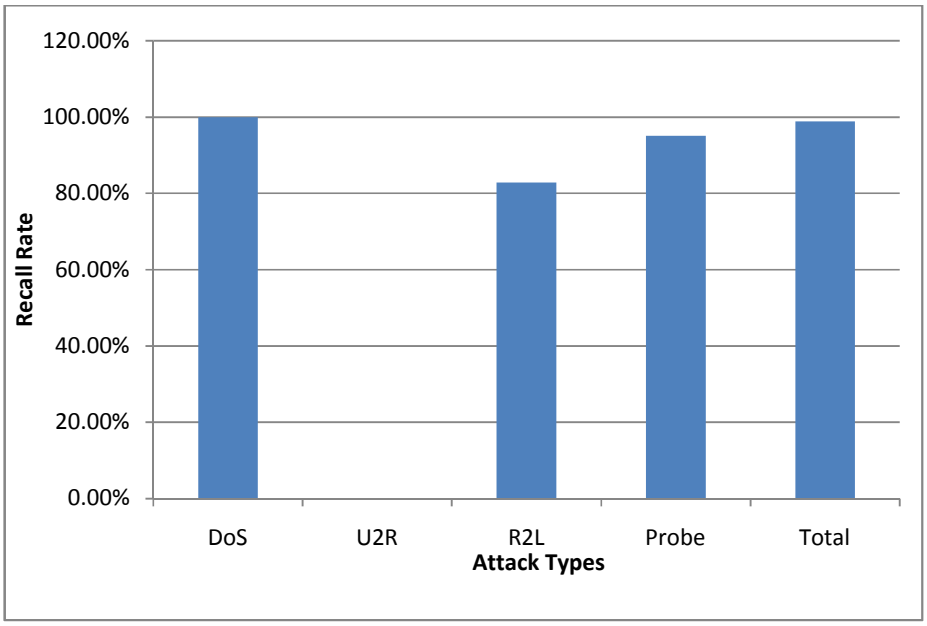


Figure 5.10: Relationship of Attack Types Versus Recall Rate

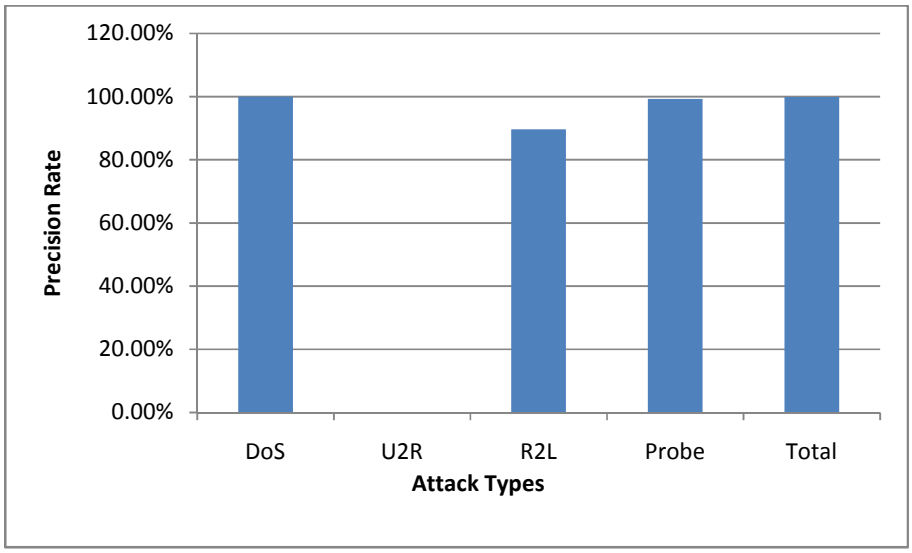


Figure 5.11: Relationship of Attack Types Versus Precision Rate

From above all comparison between different types of attack and performance we conclude that precision rate, recall rate and true positive is higher in DoS type attack. False Positive is higher in R2L type attack and false negative is higher in Probe type attack. Similarly performance is higher in 30 hidden layer network and epoch time less in 10 hidden in SCG Backpropagation algorithm and SOM but less epoch time in SOM than Backpropagation algorithm.

5.2.2 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.

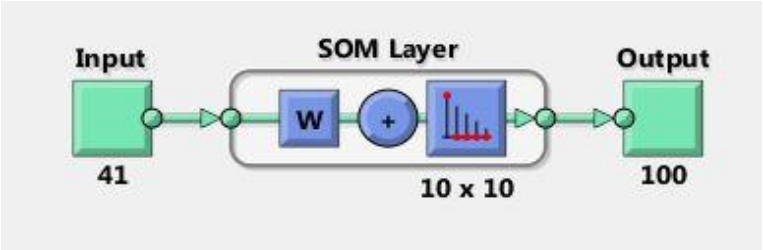


Figure 5.12: SOM Network Layer of Size 10.

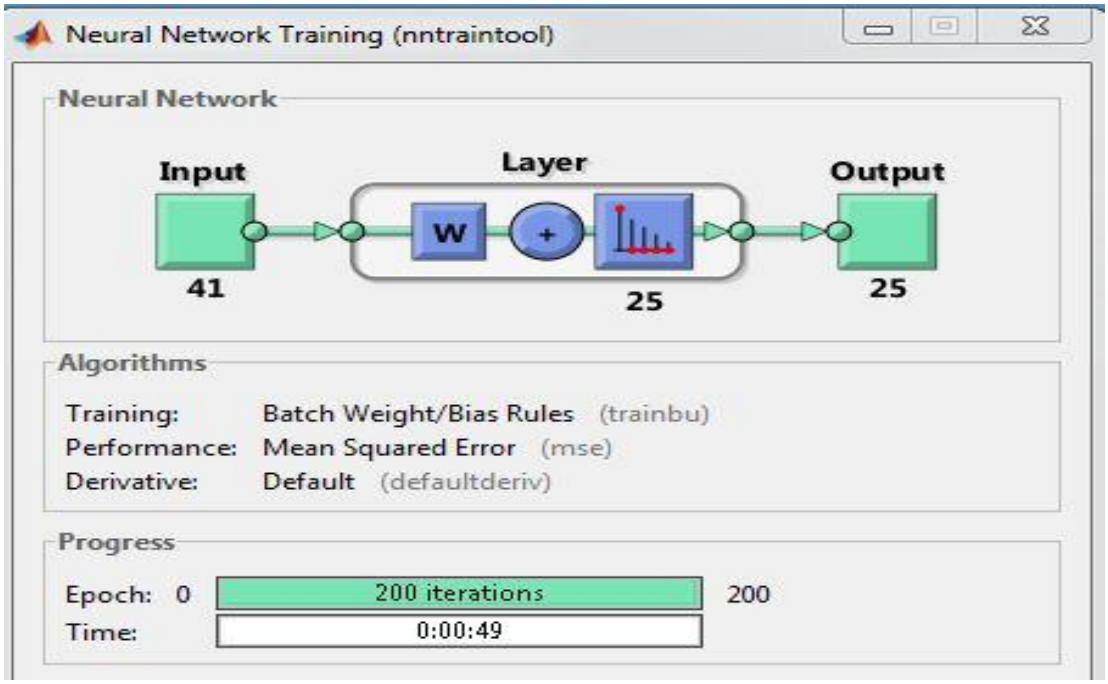


Figure 5.13: Performance of SOM with 5 neurons in hidden layer.

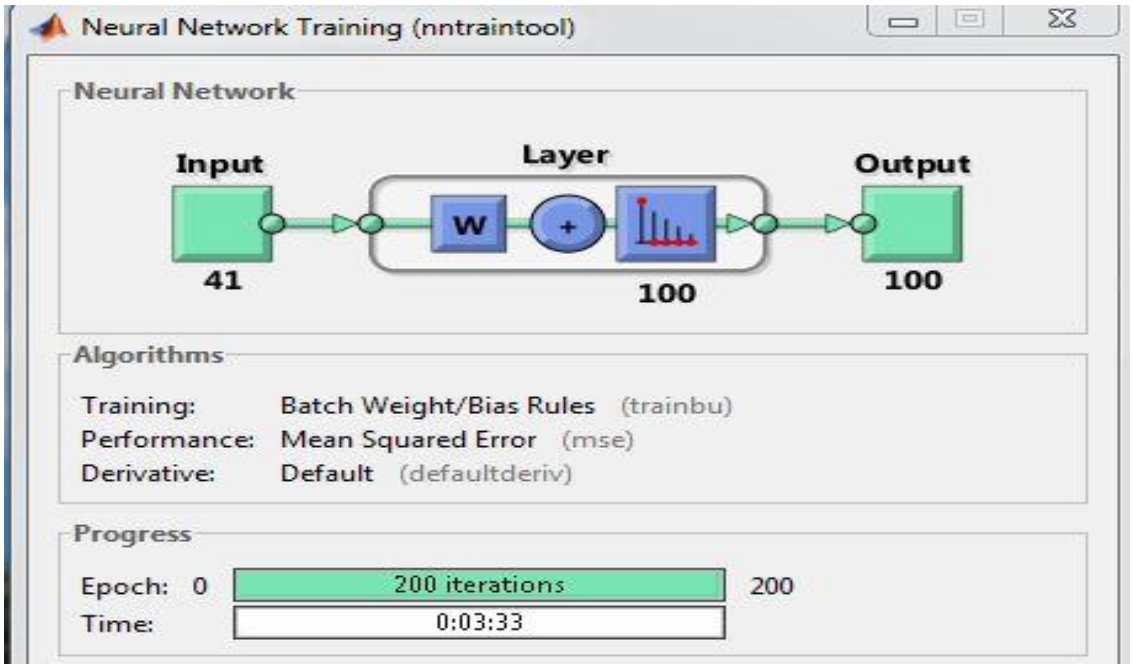


Figure 5.14: Performance of SOM with 10 neurons in hidden layer.

6. CONCLUSIONS AND RECOMMENDATIONS

6.1 Conclusions

An Intrusion Detection System is designed using supervised neural network Back Propagation Algorithm and unsupervised neural network Self Organizing Map. From above simulation result, data analysis and calculate the performance parameter we conclude the detailed in following points

- In Back Propagation algorithm calculate the performance parameters like true positive, false positive, false negative, recall rate ,precision rate in different types of attack and from result we conclude that precision rate in DoS is highest. In self organizing map calculate the performance parameters SOM neighbor connections, SOM input planes , SOM topology, SOM weight Position in different hidden layers and these parameters are good performance in 30 hidden layer network.
- In both algorithm of hidden layers are training and testing using the large number of KDD cups data set and Self Organizing Map algorithm is suggested as the most efficient model because of fast epoch time and required less memory .
- Both of Supervised and Unsupervised neural network algorithms are verified using JAVA code also.

6.2 Future Recommendations

Network Intrusion Detection can be done using other types of Neural Networks like Radial Basis Function Neural Network and using unsupervised networks like Self Organizing Map (SOM) in real data.

7. 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, “Nextgeneration 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] Kohonen, T. 1995. Self-Organizing Maps, volume 30 of Springer Series in Information Sciences. Berlin, Heidelberg: Springer. (Second Extended Edition 1997).
[10] KDD Cup 1999 dataset. Available on:
http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

8. Bibliography

[1] S. Russel and P. Norvig, " Artificial Intelligence, A Modern Approach" Pearson Education, Second Edition, First Indian Reprint, 2003.

[2] E. Rich and K. Knight, "Artificial Intelligence", Tata McGraw-Hill Edition, 26th reprint 2002.

[3] Simon Haykin, "Neural Networks", Pearson Education, 9th Indian Reprint, 2005.

Appendix A

1. Feature Name and Type of 10% KDDCup 99 dataset:

Table A.1: KDD Feature Columns Name and Type [5]

	Feature Name	Feature Type		Feature Name	Feature Type
1	duration	continuous.	22	is_guest_login	discrete.
2	protocol_type	symbolic.	23	count	continuous.
3	service	symbolic.	24	srv_count	continuous.
4	flag	symbolic.	25	serror_rate	continuous.
5	src_bytes	continuous.	26	srv_serror_rate	continuous.
6	dst_bytes	continuous.	27	rerror_rate	continuous.
7	land	discrete.	28	srv_rerror_rate	continuous.
8	wrong_fragment	continuous.	29	same_srv_rate	continuous.
9	urgent	continuous.	30	diff_srv_rate	continuous.
10	hot	continuous.	31	srv_diff_host_rate	continuous.
11	num_failed_logins	continuous.	32	dst_host_count	continuous.
12	logged_in	discrete.	33	dst_host_srv_count	continuous.
13	num_compromised	continuous.	34	dst host same srv rate	continuous.
14	root_shell	continuous.	35	dst_host_diff_srv_rate	continuous.
15	su_attempted	continuous.	36	dst_host_same_src_port_rate	continuous.
16	num_root	continuous.	37	dst_host_srv_diff_host rate	continuous.
17	num_file_creations	continuous.	38	dst_host_serror_rate	continuous.
18	num_shells	continuous.	39	dst_host_srv_serror rate	continuous.
19	num_access_files	continuous.	40	dst_host_rerror_rate	continuous.
20	num_outbound_cmds	continuous.	41	dst_host_srv_rerror_rate	continuous.
21	is_host_login	discrete.	42	Label	symbolic.

2. Types of Attacks in 10% KDDCup 99 dataset:

Table A.2: Attacks Classification

Main Attack	**DOS**	**U2R**	**R2L**	**Probe**
Sub Attack	apache2 back land mailbomb neptune pod processtable smurf teardrop upstorm	buffer_overflow load module perl ps rootkit xterm	ftp_write guess_passwd imap mscam warezclient warezmaster xclock xsnoop	Ipsweep nmap portsweep satan

3. Performance of MLP with different values of Hidden Layer Neurons:

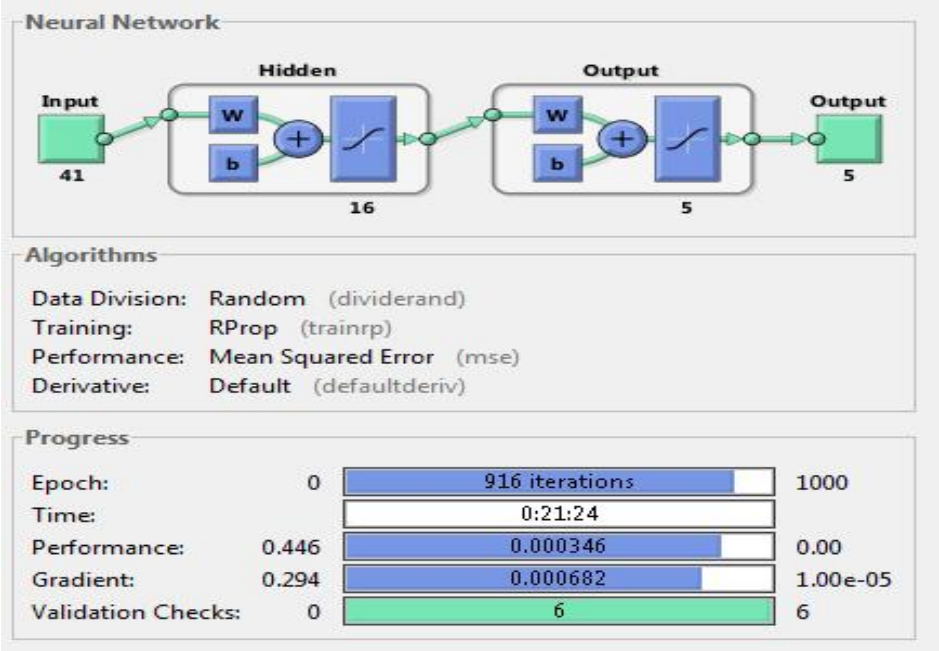


Figure A.1: Performance of MLP with 10 Neurons in Hidden Layer

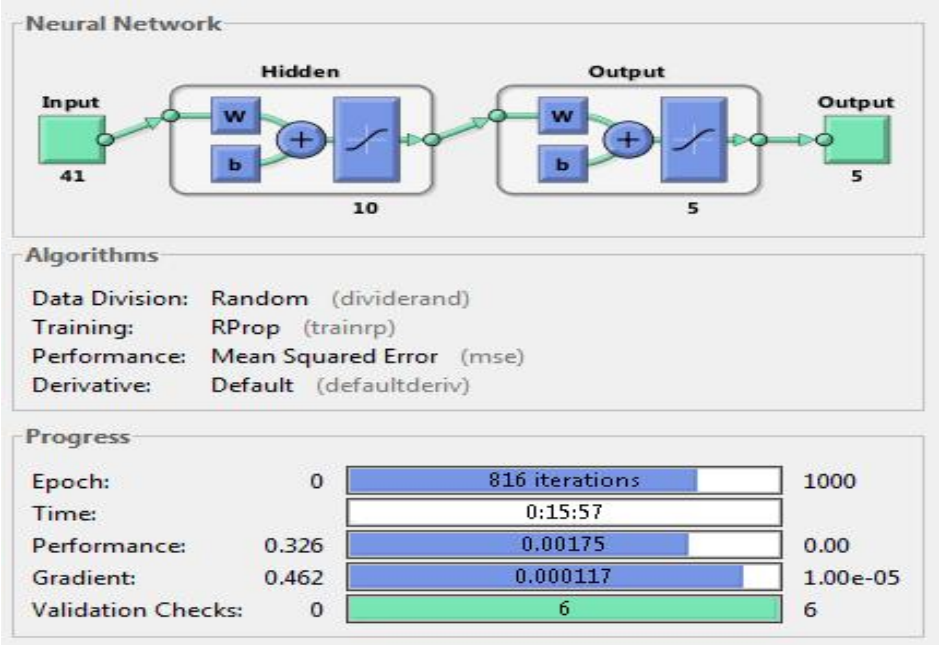


Figure A.2: Performance of MLP with 16 Neurons in Hidden Layer

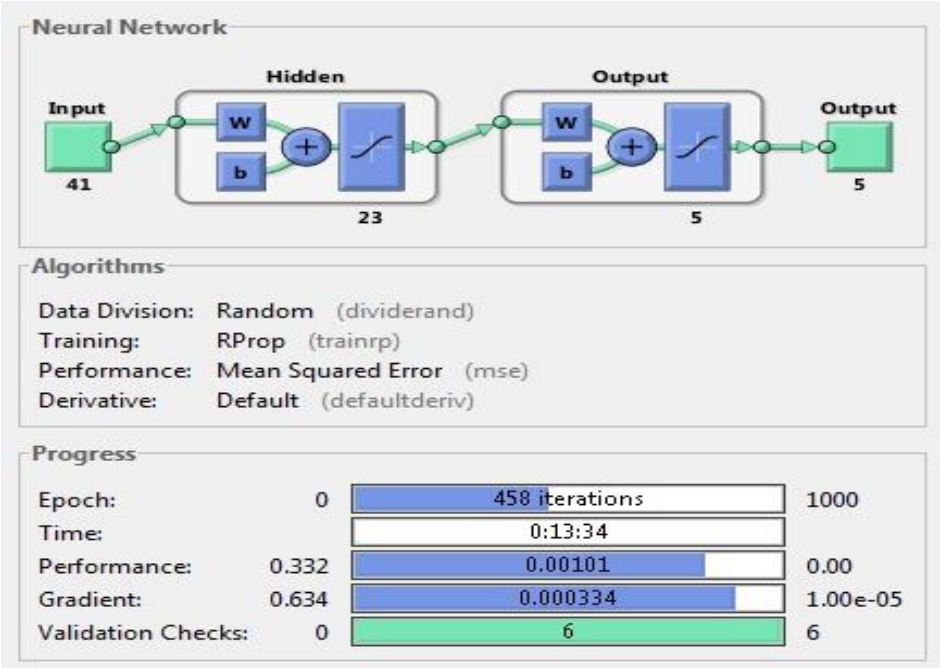


Figure A.3: Performance of MLP with 23 Neurons in Hidden Layer

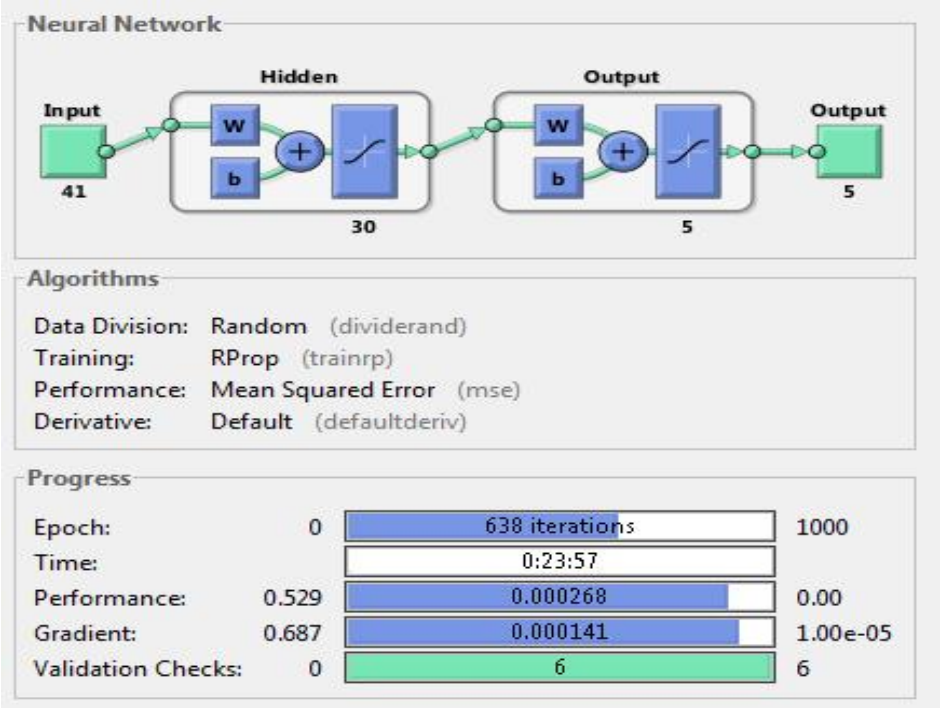


Figure A.4: Performance of MLP with 30 Neurons in Hidden Layer

Appendix B

1. Simulation Results of SCBP Algorithm:

The graphical representation of the visual impression of the distribution of Errors (Targets- Outputs) is shown as the Error Histogram plot for the given data in Figure B.1. It consists of tabular instances shown as adjacent rectangles erected over discrete bins. It shows that maximum error is ~ - 0.05. The receiver operating characteristics for training, validation and testing phases of the dataset are shown in Figure B.2. The ideal value should be close to one. As per the simulation we have got data which are mostly True Positive.

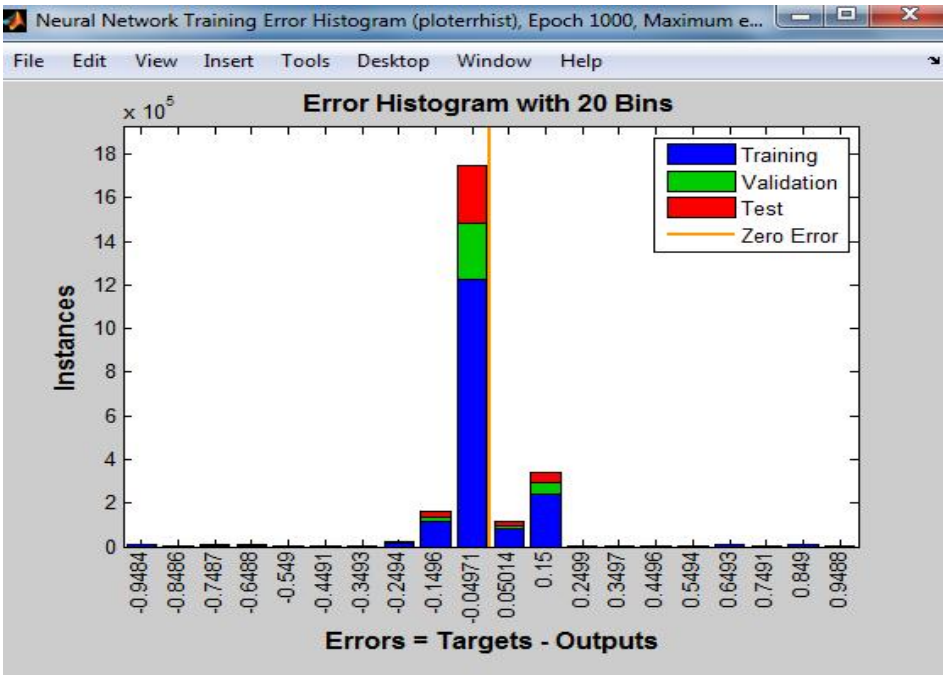


Figure B.1: Error Histogram of SCBP

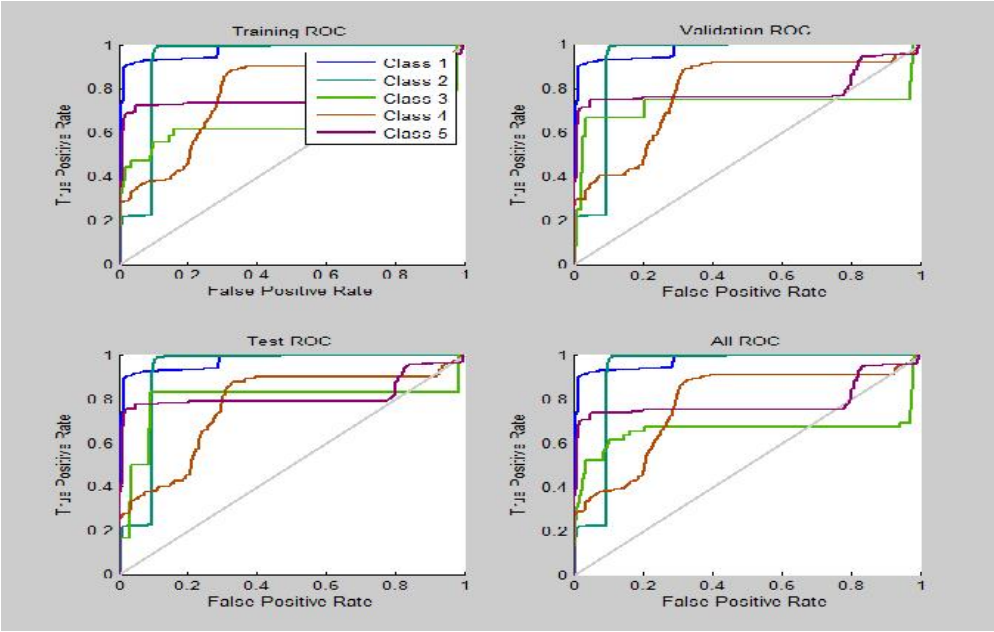


Figure B.2: ROC plot of SCBP

2 Simulation Result Of Self Organizing Map

In Self Organizing Map the simulation result of different parameters like SOM weight position, SOM input plane, SOM neighbor connection and SOM topology showing below figure.

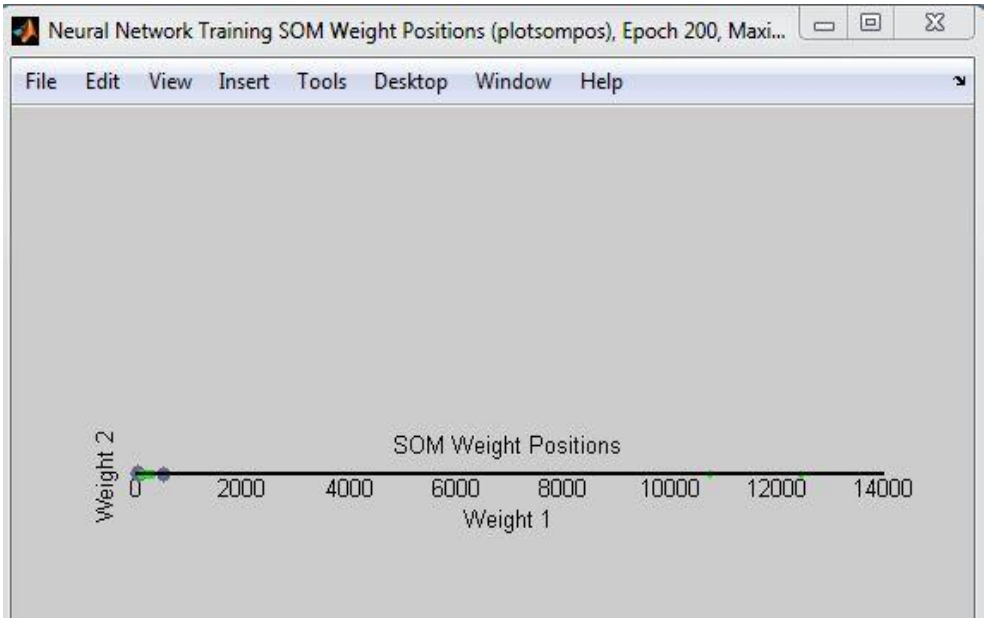


Figure: B.3 SOM weight Position

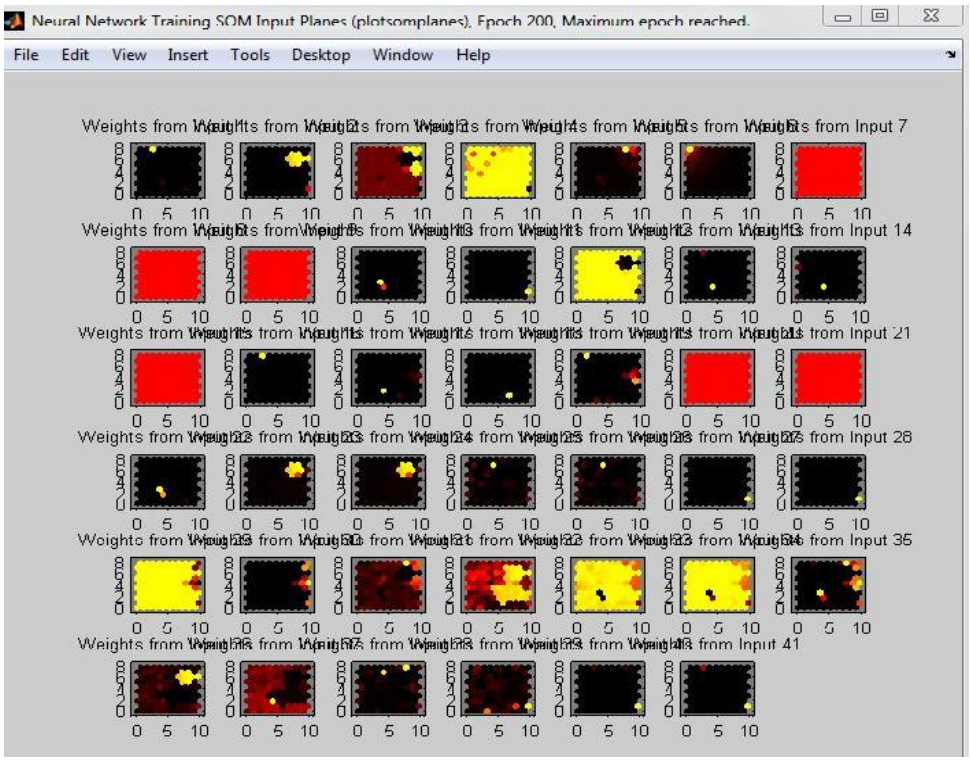


Figure: B.4 SOM input planes

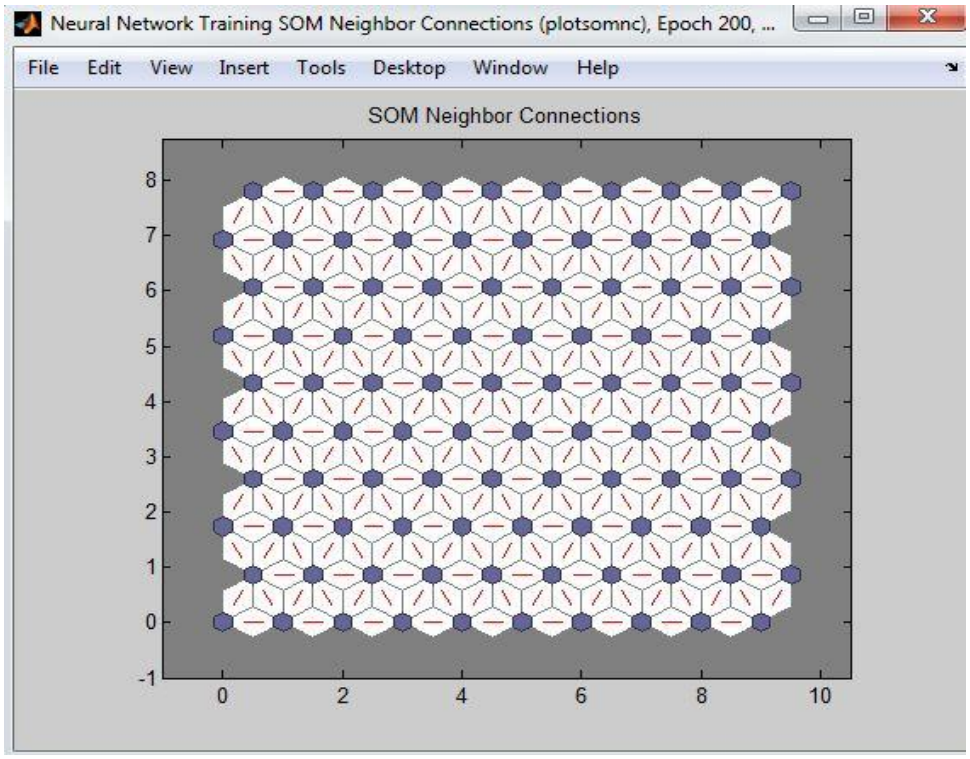


Figure : B.5 SOM neighbor connections

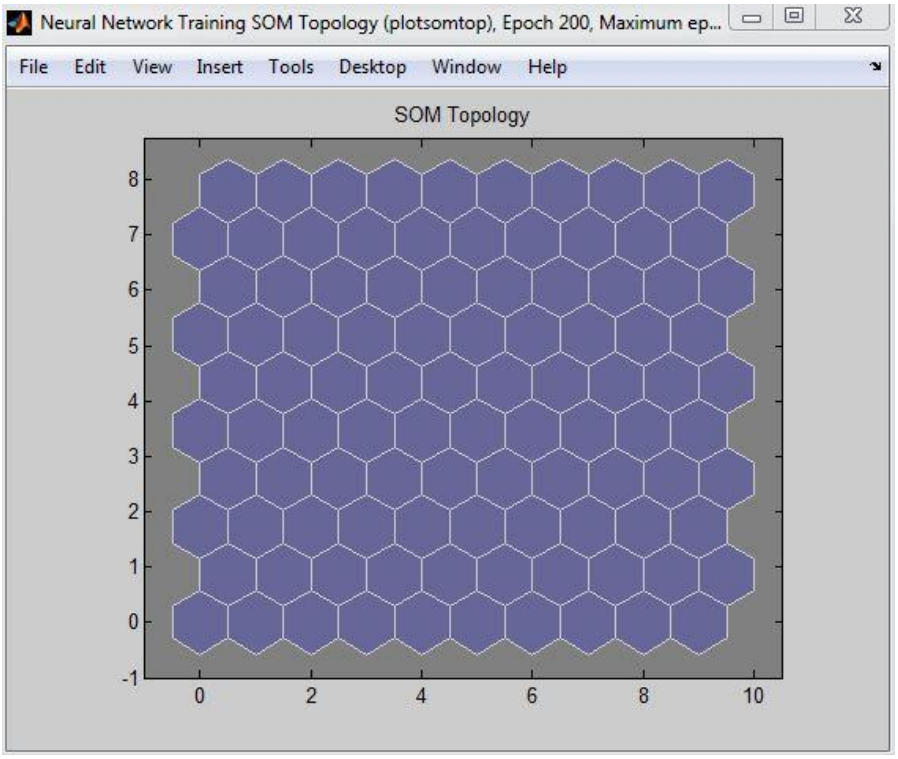


Figure: B.7 SOM topology