Tribhuvan University

Institute of Engineering, Pulchowk Campus

A

Final Thesis Report

On

"Comparative Analysis of Backpropagation Algorithm Variants for Network Intrusion Detection"

Submitted By:
Nabin Neupane
(2071/MSCS/656)

Under the Supervision of
Prof. Dr. Subarna Shakya

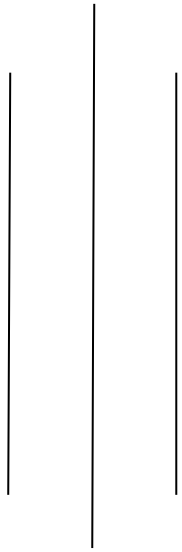
Submitted To:
Department of Electronics and Computer Engineering
October 20, 2016

A

Final Thesis Report

On

"Comparative Analysis of Backpropagation Algorithm Variants for Network Intrusion Detection"



Nabin Neupane

(2071/MSCS/656)

Department of Electronics and Computer Engineering

IOE, Pulchowk Campus

October 20, 2016

Acknowledgement

I would like to express my sincere gratitude to **Prof. Dr. Subarna Shakya**, my thesis supervisor for the constant guidance with his insightful ideas and valuable suggestions and encouragement during the period of thesis.

I would like to express my sincere thanks and indebtedness to our Head of Department **Dr. Dibakar Raj Pant**, **Prof. Dr. Sashidhar Ram Joshi**and **Dr. Aman Shakya** for their encouragement and precious guidance. I would also like to thank **Dr. Sanjeeb Prasad Panday**, Program Coordinator, Master's Degree, for his constant focus on research activity, choosing Thesis Topic and cooperation to give out the best. Last but not least I would like to thank everyone who directly or indirectly helped me to make this thesis successful.

Abstract

The network security has become a very important issue as network attacks have been increasing with the growth of hacking tools, complexity of networks and intrusions in number and severity. The intrusion detection is the process that detects possible network attacks or different security violations, abnormal activities and alerts the occurrences to network administrator. This research is focused on intrusion detection by using Multilayer Perceptron (MLP) with different algorithm of backpropagation neural network. In this research, performance of various backpropagation algorithms has been evaluated using KDDCup99 dataset. The dataset has been preprocessed to be made suitable for neural network input and the input set and target set are separated. The modified dataset has been used to evaluate the performance of BFGS Quasi-Newton, Levenberg-Marquardt, Gradient Descent with Adaptive lr backpropagation algorithm. Different performance parameters such as mean square error, attack detection rate, recall rate, precision rate, epochs has been used for the algorithm comparison.

Based on the evaluation results, the research purposes Levenberg-Marquardt backpropagation algorithm to be the best performing and efficient algorithm for the network intrusion detection for KDDCup dataset. Different classes of attacks have been also determined comparing the output values obtained with the target set.

Keywords –Intrusion detection, Multilayer Perceptron, KDDCup99, Backpropagation, Detection Rate, Recall Rate.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS----------iii

ABSTRACT----------iv

LIST OF FIGURES----------vi

LIST OF TABLES----------viii

ABBREVIATION----------ix

CHAPTER 1: INTRODUCTION

1.1 Background----------1

1.2 Problem Statement----------5

1.3 Objectives----------6

CHAPTER 2: LITERATURE REVIEW----------7

CHAPTER 3: METHODOLOGY

3.1 Input Dataset Analysis----------9

3.2 Pre Processing----------10

3.3 Determining Architecture of MLP----------12

3.4 Training and Testing of MLP----------14

3.5 Backpropagation Algorithm----------14

3.5.1 BFGS Quasi - Newton Backpropagation Algorithm----------16

3.5.2 Levenberg-Marquardt backpropagation----------17

3.5.3 Gradient descent with adaptive lr backpropagation----------19

3.6 Performance Parameters----------20

3.7 Tools----------21

CHAPTER 4: RESULTS, ANALYSIS AND COMPARISONS

4.1 Results and Analysis----------22

4.1.1 Determining Hidden Layer Neurons----------22

4.1.2 Performance Assessment of various Backpropagation Algorithms-----24

4.2 Comparison----------31

CHAPTER 5: CONCLUSION----------34

REFERENCES----------35

BIBLIOGRAPHY----------36

LIST OF FIGURES

Figure 1.1: Multilayer Perceptron----------2

Figure 1.2: Bipolar Sigmoid Function----------5

Figure 3.1: Block diagram of the proposed system----------9

Figure 3.2: Back propagation Algorithm Diagram----------15

Figure 4.1: MLP Architecture of the System----------
152

Figure 4.2: Performance of BFGSBP Algorithm----------
155

Figure 4.3: Confusion matrix of BFGS Quasi- Newton algorithm----------26

Figure 4.4: Performance of LMBP Algorithm----------27

Figure 4.5: Confusion matrix of LMBP algorithm----------28

Figure 4.6: Performance of GDABP Algorithm----------29

Figure 4.7: Confusion matrix of LMBP algorithm----------30

Figure 4.8: Comparison of MSE among BFGSBP, LMBP and GDABP Algorithms31

Figure 4.9: Comparison of Detection Rate among BFGSBP, LMBP and GDABP Algorithms----------32

Figure 4.10: Comparison of Epoch among BFGSBP, LMBP and GDABP Algorithms-
----------32

Figure 4.11: Comparison of Recall rate among BFGSBP, LMBP and GDABP Algorithms----------33

Figure 4.12: Comparison of Precision Rate among BFGSBP, LMBP and GDABP Algorithms----------33

LIST OF TABLES

Table 1: KDD feature columns name and type----------10
Table 2: Protocol Type----------11
Table 3 :Service Type----------11
Table 4 : Flag Types----------12
Table 5 : Attacks Classification----------12
Table 6 : Label Transformation----------13
Table 7: Feature Column Before Transformation----------13
Table 8 : Feature Column After Transformation----------13
Table 9: Selection of Number of Neurons in Hidden Layer----------14
Table 10: Evaluation Results for each Attack Classes (BFGSBP)----------26
Table 11: Evaluation Results for each Attack Classes (LMBP)----------28
Table 12: Evaluation Results for each Attack Classes (BFGSBP)----------30
Table 13: Simulation Result of various BP Algorithms----------31

ABBREVIATION

AFRL	Air Force Research Laboratory
ANN	Artificial Neural Network
DARPA	Defense Advance Research Project Agency
DOS	Denial of Service
FN	False Negative
FP	False Positive
IDS	Intrusion Detection System
MIT	Massachusetts Institute of Technology
NIDS	Network based IDS
R2L	Remote to Local
TN	True Negative
TP	True Positive
U2R	User to Root
BFGSBP	BFGS Quasi-Newton Backpropagation algorithm
LMBP	Levenberg-Marquardt Backpropagtion algorithm
GDABP	Gradient Descent with Adative lr Backpropagation algorithm

CHAPTER 1: INTRODUCTION

1.1 Background

The word intrusion means the act of wrongfully entering upon, seizing, or taking possession of the property of another. The number of intrusions into computer systems is growing. The reason is that new automated hacking tools are appearing every day, and these tools with variety of system vulnerability information are easily available on the web [1].

The attack detection tools are very important for providing safety in computer and network system. These tools fully depend on accuracy of attack detection. Moreover, the detection is also must for prevention of any attack. Therefore accurate detection of attack is very important. A number of attempts have been done in the field of attack detection but they suffered many limitations such as time consuming statistical analysis, regular updating, non adaptivity, lack of accuracy and flexibility. Therefore, an Artificial Neural Network (ANN) supports an ideal specification of an attack detection system and is a solution to the problems of previous systems. As a result, an ANN inspired by nervous system has become an interesting tool in the applications of attack detection systems due to its promising features. Attack detection by artificial neural networks is an ongoing area and thus interest in this field has increased among the researchers. Neural networks have the ability to classify patterns, and thus can be used in other aspects of intrusion detection systems such as attack classification and alert validation [1].

An unauthorized user who tries to enter in network or computer system is known as intruder. A system that detects and logs inappropriate activities is called as intrusion detection system (IDS). The intrusion detection systems can be classified into three categories: host based, network based and vulnerability assessment based. A host based IDS evaluates information found on a single or multiple host systems, including contents of operating systems, system files and application files. While network based IDS evaluates information captured from network communications, analyzing the stream of packets traveling across the network. Packets are captured through a set of sensors. Vulnerability assessment based IDS detects vulnerabilities on internal networks and firewall [2].

There are two general methods of detecting intrusions into computer and network systems:

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].

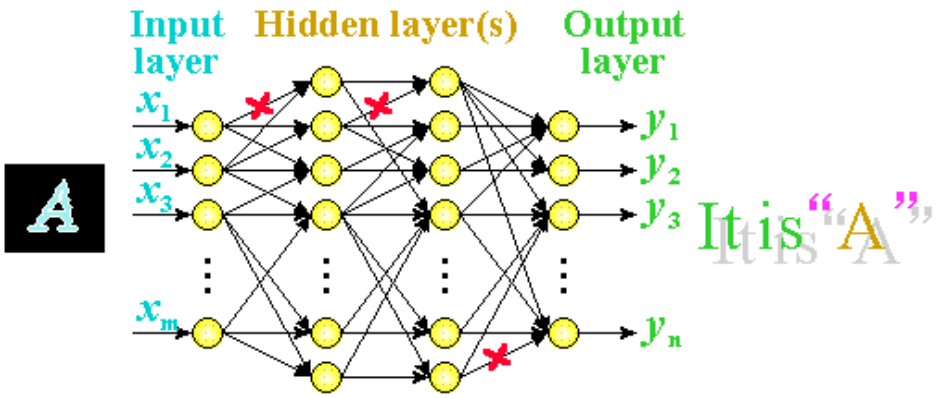


Figure 1.1: Multilayer Perceptron [3]

Figure 1 above shows the input layer, hidden layer(s) and output layer of Multilayer Perceptron (MLP).

Types of attacks

The goal of efficient IDS is to detect novel attacks by unauthorized users in network traffic. Attacks can be gathered in four main categories. They are as follows:

1. **Denial of Service Attack (DoS)**
2. **User to Root Attack (U2R)**
3. **Remote to Local Attack (R2L)**
4. **Probing Attack**

1) Denial of Service Attack (DoS)

A DoS (Denial of Service) attack is a type of attack in which the attacker or unauthorized users makes a computing or memory resources too busy or too full to provide reasonable networking requests and hence denying users access to a machine e.g. ping of death, neptune, back, smurf, apache, UDP storm, mail bomb etc. are all DoS attacks.

2) User to Root Attack (U2R)

A user to root (U2R) attack is an attack in which the attacker 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. The attacker forwards networking packets to a machine through the internet, which he/she does not have right of access in order to expose the machines vulnerabilities and exploit privileges which a local user would have on the computer e.g. guest, xlock, xnsnoop, sendmail dictionary, phf etc.

3) Remote to Local Attack (R2L)

A remote to local (R2L) attacks are regarded as the exploitations in which the unauthorized users start off on the system with a normal user account and tries to misuse vulnerabilities in the system in order to achieve super user access rights e.g. xterm, perl. It 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

A probing is an attack in which the hacker scans a machine or a networking device in order to determine weaknesses or vulnerabilities that may later be exploited so as to negotiate the system. This practice is commonly used in data mining e.g. portsweep, saint, mscan, nmap etc. Attacker tries to gain information about the target host. [7]

Training and Testing of MLP

In ANN, generally initial weights and biases are set randomly with small values. Once, these values are set, the network becomes ready to be trained. Training a network generally means feeding the network with the training sequence. The training sequences are simply the vectors of input combinations along with the required output. The network processes the input vector, changes its internal weights and biases to give the result near to the output. After certain epochs of the training, the weights and biases are kept constant and real environment data is fed to the network to test the network. There are quite a few training algorithms developed during the years of time which provides good result in terms of how fast network converges to problem, how much memory does the network uses to produce the output etc.

Backpropagation (BP)

Backpropagation is the most well-known and widely used neural network system. It uses the backpropagation rule for training. The BP 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.

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: $f(x) = -1 + \frac{2}{1+e^{-x}}$…….....……..... (1.1)

which has derivative of: $f'(x) = 0.5 * [1 + f(x)] * [1 - f(x)]$ …….....…..... (1.2)

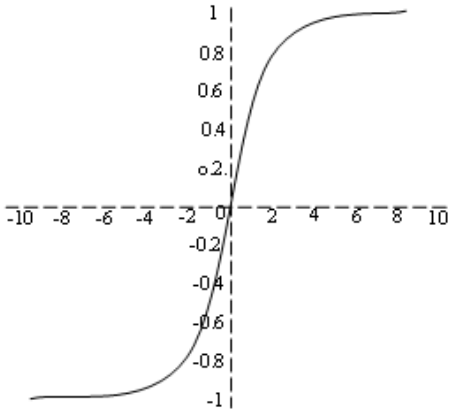


Figure 1.2: Bipolar Sigmoid Function

Mean Square Error (MSE)

The process of training a neural network involves tuning the values of the weights and biases of the network to optimize network performance, as defined by the network performance function `net.performFcn`. The default performance function for feedforward networks is mean square error - the average squared error between the network outputs '`a`' and the target outputs 't'. It is defined as follows:

$$F = mse = \frac{1}{N}\sum_{i=1}^{N}(e_i)^2 = \frac{1}{N}\sum_{i=1}^{N}(t_i - a_i)^2 \quad \text{…….. (1.3)}$$

In this research, MLP is trained with various backpropagation algorithms. Based on the evaluation results, the proposed research is able to suggest the best model for network based intrusion 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 Objectives

Objectives of the research are as follows:

- To analyze the performance of various backpropagation algorithms and suggest the efficient model for network intrusion detection based on the evaluation result.
- To detect intrusion using multilayer perceptron with backpropagation.

CHAPTER 2: LITERATURE REVIEW

One preliminary IDS concept consisted of a set of tools intended to help administrators review audit trails. User access logs, file access logs, and system event logs are examples of audit trails. Fred Cohen noted in 1984 that it is impossible to detect an intrusion in every case, and that the resources needed to detect intrusions grow with the amount of usage [4].

Several research works have already been done and many research papers have been published regarding improvement of intrusion detection system (IDS). Since, each of the papers has focused on different algorithmic techniques being implemented in IDS with their resulted output in simulation tools as well. However, the comparative analysis is very rare and proposed research is crucial in today's time in order get the de-facto standard for efficient IDS implementation. Some of the related works that are closely related to proposed work are highlighted below along with their scope of research.

The research work done by XiaoHang Yao put forward an IDS combining with genetic algorithm and backpropagation. The intrusion detection system model presented in this paper adopts anomaly detection and misuse detection means. The system is composed of eight different modules. IDS can offer protection from external users and internal attackers, where traffic doesn't go past the firewall at all. The research on IDS attempted to use neural networks for intrusion detection has been carried on and will continue. Such systems were trained on normal or attack behavior information and then detect intrusions or attacks. In this paper, five kinds of Neural Network technologies that are used in IDS. An IDS combining with GA and BP is put forward, and functions of each module are detailed. The result of experiment shows that combining genetic algorithm with backpropagation efficiently enhances the learning speed of backpropagation neural network and improves the detection accurate rate of IDS. Finally, a discussion of the future neural network technologies, which guarantee to enhance the detection efficiency of IDS is provided [2].

The research work done by Jingwen Tian, Meijuan Gao and Fan Zhang was network intrusion detection method based on radial basic function neural network. Aimed at the network intrusion behaviors are characterized with uncertainty, complexity, diversity and dynamic tendency and the advantages of radial basic function neural network (RBFNN), an intrusion detection method based on radial basic function

neural network is presented in this paper. They constructed the structure of RBFNN that used for detection network intrusion behavior, and adopt the K-Nearest Neighbor algorithm and least square method to train the network. They discussed and analyzed the impact factor of intrusion behaviors. With the ability of strong function approach and fast convergence of radial basic function neural network, the network intrusion detection method based on radial basic function neural network can detect various intrusion behaviors rapidly and effectively by learning the typical intrusion characteristic information [3].

There is another research work performed by Farah Jemili, Montaceur Zaghdoud and Mohamad Ben Ahmed, which uses Bayesian Network to build automatic intrusion detection system based on signature recognition. A Bayesian Network (BN) is known as graphical modeling tool used to model decision problems containing uncertainty. In this paper, a BN is used to build automatic intrusion detection system based on signature recognition. A major difficulty of this system is that the uncertainty on parameters can have two origins. The first source of uncertainty comes from the uncertain character of information due to a natural variability resulting fmm stochastic phenomena. The second source of uncertainty is related to the imprecise and incomplete character of information due to a lack of knowledge. The goal of this work is to propose a method to propagate both the stochastic and the epistemic uncertainties, coming respectively from the uncertain and imprecise character of information, through the Bayesian model, in an intrusion detection context [7].

CHAPTER 3: MEHODOLOGY

Methodology of the system is shown in the Figure 3.1 below:

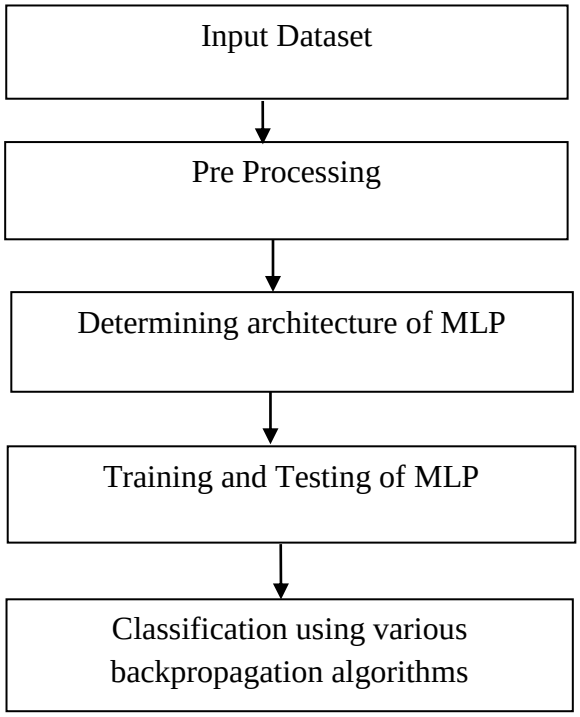


Figure 3.1: Block diagram of the system.

3.1 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 [5]. 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 KDDCup'99 features from each record, the instances are labeled based on the characteristics of traffic as Normal, Dos, Probe, R2L and U2R.

3.2 Pre Processing:

The data set is preprocessed so that it may be able to give it as an input to java programming and matlab readable format. This data set consists of numeric and symbolic features and is converted into numeric form so that it can be given as inputs to our MLP network. Now this modified data set is used as training and testing data set of the multi-layer perceptron.

Table 1 below shows the feature columns name and type of 10% KDDCup 99 dataset.

Table 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 2 demonstrate the transformation table for protocol_type.

Table 2: Protocol Type

Protocol_type	No.
TCP	1
UDP	2
ICMP	3

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

Table 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
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

Service	No.	Service	No.
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.

Table 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 5 and then the normal and attack values are transformed into numeric as shown in Table 6 below.

Table 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 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 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 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.

3.3 Determining Architecture of MLP

There is no certain mathematical approach for obtaining the optimum number of hidden layers and their neurons. In this research, 3 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 9 below shows the performance of multilayer perceptron with different numbers of hidden layer neurons. The best performance is observed with 20 neurons in the hidden layer.

Table 9: Selection of Number of Neurons in Hidden Layer

Hidden Layer	No. of Neurons	Performance
H1	10	0.0018104
H1	15	0.0024706
H1	20	0.00053962
H1	25	0.00073856

3.4 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, 2013).

In this thesis, 70% data from the input dataset are used for training, 15% for validation and 15% for testing of the MLP to analyze the performance of various backpropagation algorithms. The results are shown in chapter 4.

3.5 Backpropagation Algorithm:

The BFGS Quasi-NewtonBackpropagation Algorithm, Levenberg-Marquardt backpropagationand Gradient descent with adaptive lr backpropagationis used for training of MLP and performances of these algorithms are compared.

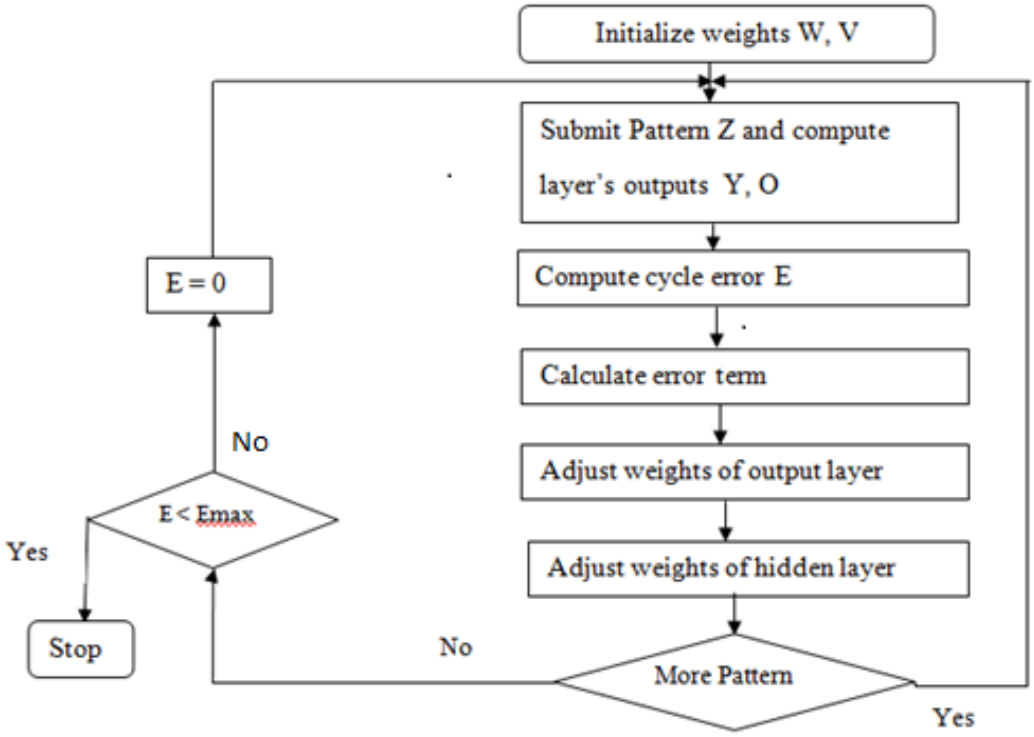


Figure 3.2: Backpropagation Algorithm Diagram

Figure 3.2 above shows the flowchart of backpropagation algorithm.

Backpropagation Algorithm Steps:

Step 0: Initialize the weights to small random values.

Step 1: Feed the training sample through the network and determine the final output.

Step 2: Compute the error for each output unit, for unit k it is:

$$\delta_k = (t_k - y_k) f'(y_{ink}) \quad \text{..... (3.1)}$$

Step 3: Calculate the weight correction term for each output unit, for unit k if it is:

$$\Delta w_{jk} = \alpha \delta_k Z_j \quad \text{..... (3.2)}$$

Step 4: Propagate the delta terms (error) back through the weights of the hidden units where the delta input for the j^{th} hidden unit is:

$$\delta_{inj} = \sum_{k=1}^{m} \delta_k W_{jk} \quad \text{..... (3.3)}$$

The delta term for the j^{th} hidden unit is:

$$\delta_j = \delta_{inj} f'(z_{inj}) \quad \text{..... (3.4)}$$

Step 5: Calculate the weight correction term for the hidden units:

$$\Delta V_{ij} = \alpha \delta_j Xi \quad \text{..... (3.5)}$$

Step 6: Update the weights:

$$W_{JK}(new) = W_{jk}(old) + \Delta W_{jk}(for\_output\_layer) \quad \text{..... (3.6)}$$

$$V_{ij}(new) = V_{ij}(old) + \Delta V_{ij}(for\_hidden\_layer) \quad \text{.....… (3.7)}$$

Step 7: Test for stopping (maximum cycles, small changes, etc).

The BFGS Quasi-Newton, Levenberg-Marquardt and Gradient Descent with Adaptive lr backpropagation algorithms are used for training of MLP and performance of these algorithms is compared.

3.5.1 BFGS Quasi - Newton Backpropagation Algorithm:

Quasi-Newton BP (BFGS) method is an alternative to the conjugate gradient methods for fast optimization. Newton's method often converges faster than conjugate gradient methods. The weight update for the Newton's method is:

$$\boldsymbol{w_{k+1} = w_k - H^{-1}_k g_k} \quad \text{.....(3.8)}$$

$\mathbf{H}_k$ is the Hessian matrix of the performance index at the current values of the weights and biases. When $\mathbf{H}_k$ is large, it is complex and time consuming to compute w_{k+1}. Fortunately, there is a class of algorithms based on the works of Broyden, Fletcher, Goldfarb, and Shanno (BFGS) that are based on Newton's method but which don't require intensive calculation. This new class of method is called quasi-Newton method. The new weight w_{k+1} is computed as a function of the gradient and the current weight w_k.

Training occurs according to training parameters, with default values.Any or all of these can be overridden with parameter name/value argumentpairs appended to the input argument list, or by appending a structureargument with fields having one or more of these names:

epochs 100 Maximum number of epochs to train

show	25	Epochs between displays
showCommandLine	0	generate command line output
showWindow	1	show training GUI
goal	0	Performance goal
time	inf	Maximum time to train in seconds
min_grad	1e-6	Minimum performance gradient
max_fail	5	Maximum validation failures

Parameters related to line search methods (not all used for all methods):

scal_tol	20	Divide into delta to determine tolerance for linear search.
alpha	0.001	Scale factor which determines sufficient reduction in perf.
beta	0.1	Scale factor which determines sufficiently large step size.
delta	0.01	Initial step size in interval location step.
gama	0.1	Parameter to avoid small reductions in performance. Usually set to 0.1
low_lim	0.1	Lower limit on change in step size.
up_lim	0.5	Upper limit on change in step size.
maxstep	100	Maximum step length.
minstep	1.0e-6	Minimum step length.
bmax	26	Maximum step size.

3.5.2 Levenberg-Marquardt backpropagation:

This method updates weight and bias values according to Levenberg-Marquardt optimization. It is often the fastest backpropagation algorithm available, and is highly recommended as a first-choice supervised algorithm, although it does require more memory than other algorithms.

Like the quasi-Newton methods, the Levenberg-Marquardt algorithm was designed to approach second-order training speed without having to compute the Hessian matrix.

When the performance function has the form of a sum of squares (as is typical in training feedforward networks), then the Hessian matrix can be approximated as

$\mathbf{H} = \mathbf{J}^T\mathbf{J}$……(3.9)

and the gradient can be computed as

$\mathbf{g_k} = \mathbf{J}^T\mathbf{e}$……....(3.10)

where **J** is the Jacobian matrix that contains first derivatives of the network errors with respect to the weights and biases, and **e** is a vector of network errors. The Jacobian matrix can be computed through a standard backpropagation technique that is much less complex than computing the Hessian matrix.

The Levenberg-Marquardt algorithm uses this approximation to the Hessian matrix in the following Newton-like update:

$$w_{k+1}=w_k-[J^TJ+\mu I]^{-1}J^Te \quad \text{…… (3.11)}$$

When the scalar μ (mu) is zero, this is just Newton's method, using the approximate Hessian matrix. When μ is large, this becomes gradient descent with a small step size. Newton's method is faster and more accurate near an error minimum, so the aim is to shift toward Newton's method as quickly as possible. Thus, μ is decreased after each successful step (reduction in performance function) and is increased only when a tentative step would increase the performance function. In this way, the performance function is always reduced at each iteration of the algorithm.

Training occurs according to training parameters, with default values. Any or all of these can be overridden with parameter name/value argument pairs appended to the input argument list, or by appending a structure argument with fields having one or more of these names:

show	25	Epochs between displays
showCommandLine	0	generate command line output
showWindow	1	show training GUI
epochs	100	Maximum number of epochs to train
goal	0	Performance goal

max_fail 5 Maximum validation failures

min_grad 1e-10 Minimum performance gradient

mu 0.001 Initial Mu

mu_dec 0.1 Mu decrease factor

mu_inc 10 Mu increase factor

mu_max 1e10 Maximum Mu

time inf Maximum time to train in seconds

3.5.3 Gradient descent with adaptive lr backpropagation:

This updates weight and bias values according to gradient descent with adaptive learning rate. An adaptive learning rate attempts to keep the learning step size as large as possible while keeping learning stable. The learning rate is made responsive to the complexity of the local error surface. An adaptive learning rate requires some changes in the training procedure used by *Gradient Descent Backpropagation*. First, the initial network output and error are calculated. At each epoch new weights and biases are calculated using the current learning rate. New outputs and errors are then calculated. This method can train any network as long as its weight, net input, and transfer functions have derivative functions.

Backpropagation is used to calculate derivatives of performance dperf with respect to the weight and bias variables X. Each variable is adjusted according to gradient descent:

$$dX = lr * dperf/dX \quad \text{(3.12)}$$

At each epoch, if performance decreases toward the goal, then the learning rate is increased by the factor lr_inc. If performance increases by more than the factor max_perf_inc, the learning rate is adjusted by the factor lr_dec and the change that increased the performance is not made.

show 25 Epochs between displays

showCommandLine 0 generate command line output

showWindow 1 show training GUI

epochs 10 Maximum number of epochs to train

goal 0 Performance goal

lr 0.01 Learning rate

lr_inc 1.05 Ratio to increase learning rate

lr_dec 0.7 Ratio to decrease learning rate

max_fail 5 Maximum validation failures

max_perf_inc 1.04 Maximum performance increase

min_grad 1e-10 Minimum performance gradient

time inf Maximum time to train in seconds

Training stops when any of these conditions occurs:

- The maximum number of epochs (repetitions) is reached.
- The maximum amount of time is exceeded.
- Performance is minimized to the goal.
- The performance gradient falls below min_grad.
- Validation performance has increased more than max_fail times since the last time it decreased (when using validation).

3.6 Performance Parameters:

Mean Square Error, Total CPU Time of Converge and Accuracy will be the performance parameters to compare various backpropagation 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)

3.7 Tools:

Matlab 2013:

Simulation is performed using neural network in MATLAB. Coding is also done to perform training and testing of MLP in Java. 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.

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.

CHAPTER 4: RESULTS,ANALYSIS AND COMPARISON

4.1 Results and Analysis

4.1.1 Determining Hidden Layer Neurons

The Multilayer Perceptron 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

Above Table 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 4.1.

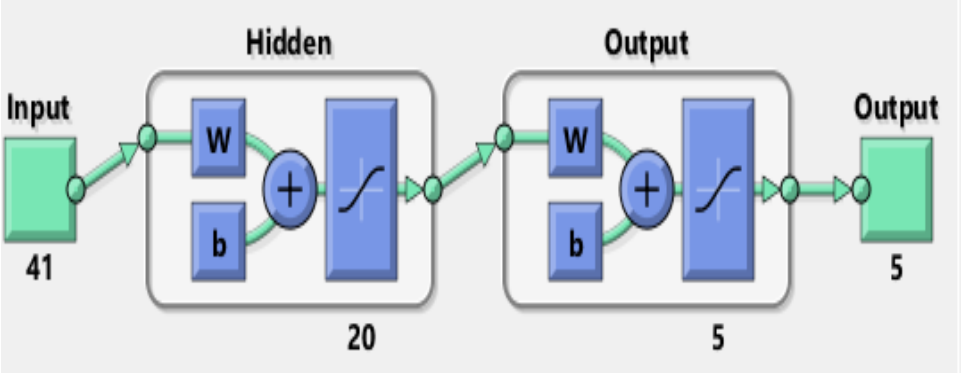


Figure 4.1: MLP Architecture of the System

The preprocessing step is carried out and obtained dataset is fed to the multilayer perceptron. Since there is no accurate formula for the selection of hidden layer neurons, a comparison is made for many cases to find optimum number of neurons. The simulation is performed in Matlab to find the proper number of neurons in the hidden layer. Simulation is done to analyze the performance of BFGS Quasi Newton, Levenberg-Marquardt and Gradient Descent with adaptive lr back propagation algorithms. The performance, mean square error, detection rate, time, epoch of those algorithms were compared.

The snapshots of target and actual output for different values of learning rate are shown below. The best output is observed at learning rate (α) = 0.01.

The target and actual output for learning rate (α) = 0.1.

```
 target and the actual output:

target
1.0     0.0     0.0     0.0     0.0
actual
0.97767 0.02281 0.03112 0.01994 0.02364
target
1.0     0.0     0.0     0.0     0.0
actual
0.97905 0.02881 0.01367 0.02834 0.03069
target
1.0     0.0     0.0     0.0     0.0
actual
0.97741 0.01766 0.02028 0.01962 0.01568
target
1.0     0.0     0.0     0.0     0.0
actual
0.97741 0.01766 0.02028 0.01962 0.01568
```

The target and actual output for learning rate (α)=0.05

```
 target and the actual output:

target
1.0     0.0     0.0     0.0     0.0
actual
0.98673 0.01326 0.01325 0.01326 0.01325
target
1.0     0.0     0.0     0.0     0.0
actual
0.98673 0.01326 0.01325 0.01326 0.01325
target
1.0     0.0     0.0     0.0     0.0
actual
0.98673 0.01326 0.01325 0.01326 0.01325
target
1.0     0.0     0.0     0.0     0.0
actual
0.98673 0.01326 0.01325 0.01326 0.01325
```

The target and actual output putting learning rate (α)=0.01

```
 target and the actual output:

target
1.0     0.0     0.0     0.0     0.0
actual
0.99453 0.00638 0.00696 0.00655 0.00695
target
1.0     0.0     0.0     0.0     0.0
actual
0.98939 0.00871 0.00699 0.00825 0.00704
target
1.0     0.0     0.0     0.0     0.0
actual
0.99453 0.00638 0.00696 0.00655 0.00695
target
1.0     0.0     0.0     0.0     0.0
actual
0.99453 0.00638 0.00696 0.00655 0.00695
```

By comparing the values of actual output and Table 6 (label transformation), type of attack can be identified easily. For example, if first value of actual output is nearly equal to 1 and other values are nearly equals to 0 that is normal type activity. If second value of actual output is nearly equal to 1 and other values are nearly equal to 0 that is DoS type attack and so on.

4.1.2 Performance Assessment of various Backpropagation Algorithms

Simulation is done to analyze the performance of BFGS Quasi Newton, Levenberg-Marquardt and Gradient Descent with adaptive lr back propagation algorithms.

BFGS Quasi Newton Backpropagation (BFGSBP):

The Multilayer Perceptron was trained with BFGSBP algorithm by using following parameters:

Scale factor that determines sufficient reduction in perf(α)= 0.001

Scale factor that determines sufficiently large stepsize (β)=0.1

Initial step size in interval location step (Δ)=0.01

Simulation result of BFGSBP algorithm (Figure 4.2) shows the MSE and number of epochs. The best performance is observed at epoch 79.

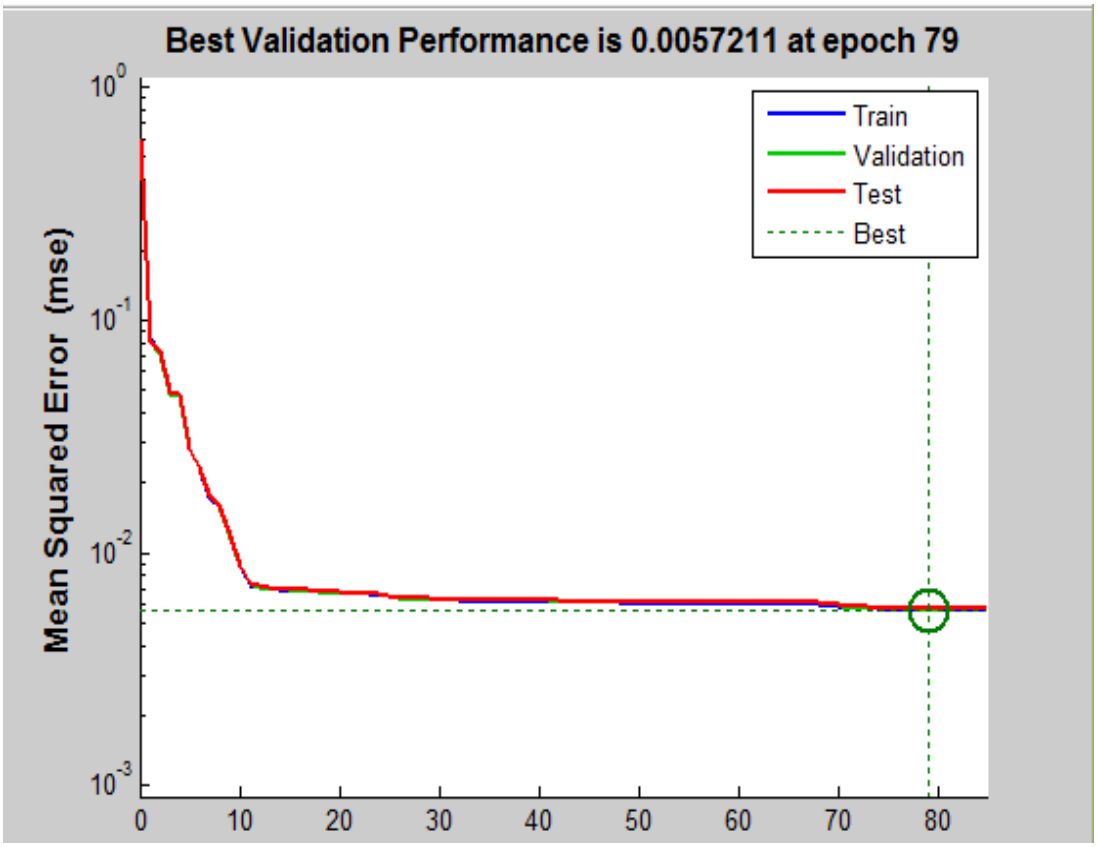


Figure 4.2: Performance of BFGSBP Algorithm

All confusion matrix gives the value of True positive (TP), False Negative (FN) and True Negative (TN). The diagonal values starting from column 2, row 2 are the values of TP. The vertical values starting from column 1, row 2 are the values of FP and the horizontal values starting from column 2, row 1 are the values of FN. Recall and Precision rate are calculated according to the formula mentioned above.

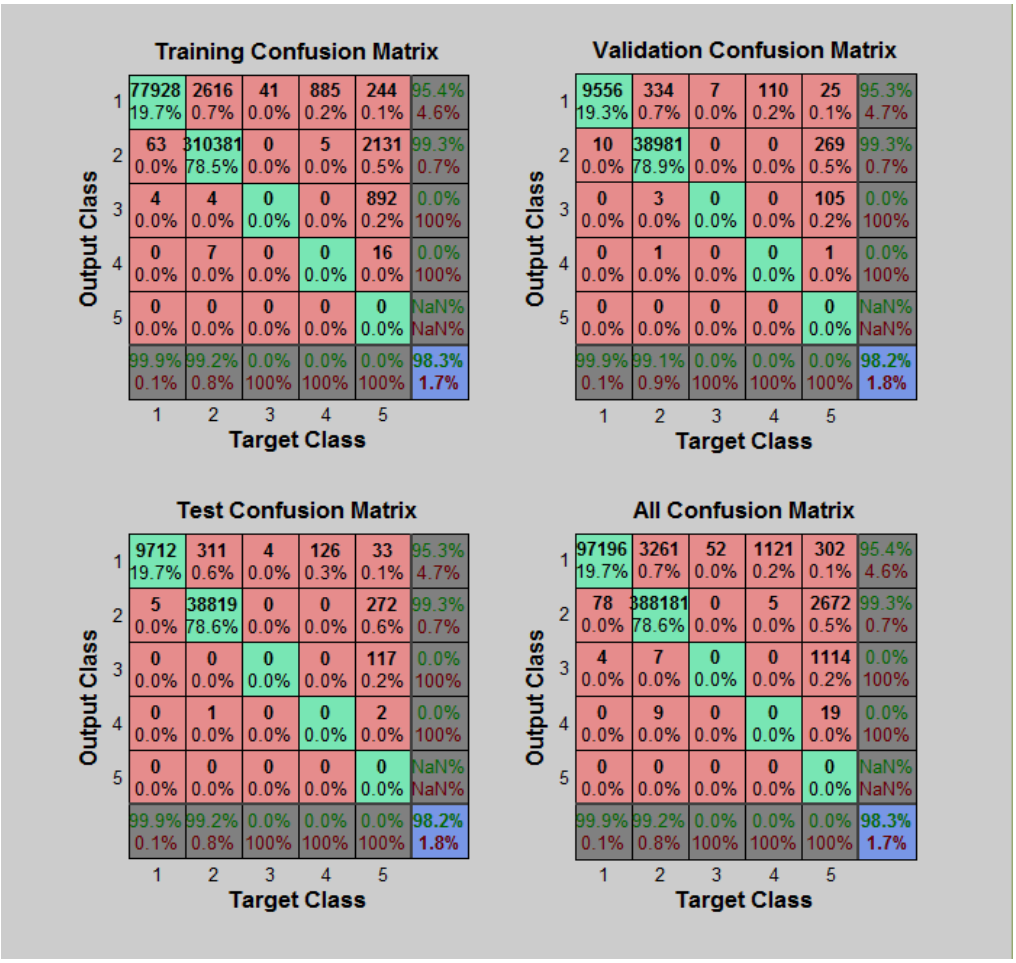


Figure 4.3: Confusion matrix of BFGS Quasi- Newton algorithm.

Table 10 below shows the evaluation results for each attack classes.

Table 10: Evaluation Results for each Attack Classes (BFGSBP)

Attack	**TP**	**FP**	**FN**	**Recall**	**Precision**
DoS	388181	78	3261	99.16%	99.97%
U2R	0	4	52	0%	0%
R2L	0	0	1121	0%	0%
Probe	0	0	302	0%	0%
Total	388181	82	4736	98.79%	99.97%

Levenberg – Marquardt Backpropagation (LMBP):

The Multilayer Perceptron was trained with LMBP algorithm by using default parameters. Simulation result of LMBP algorithm (Figure 4-4) shows the MSE and number of epochs. The best performance is observed at epoch 240.

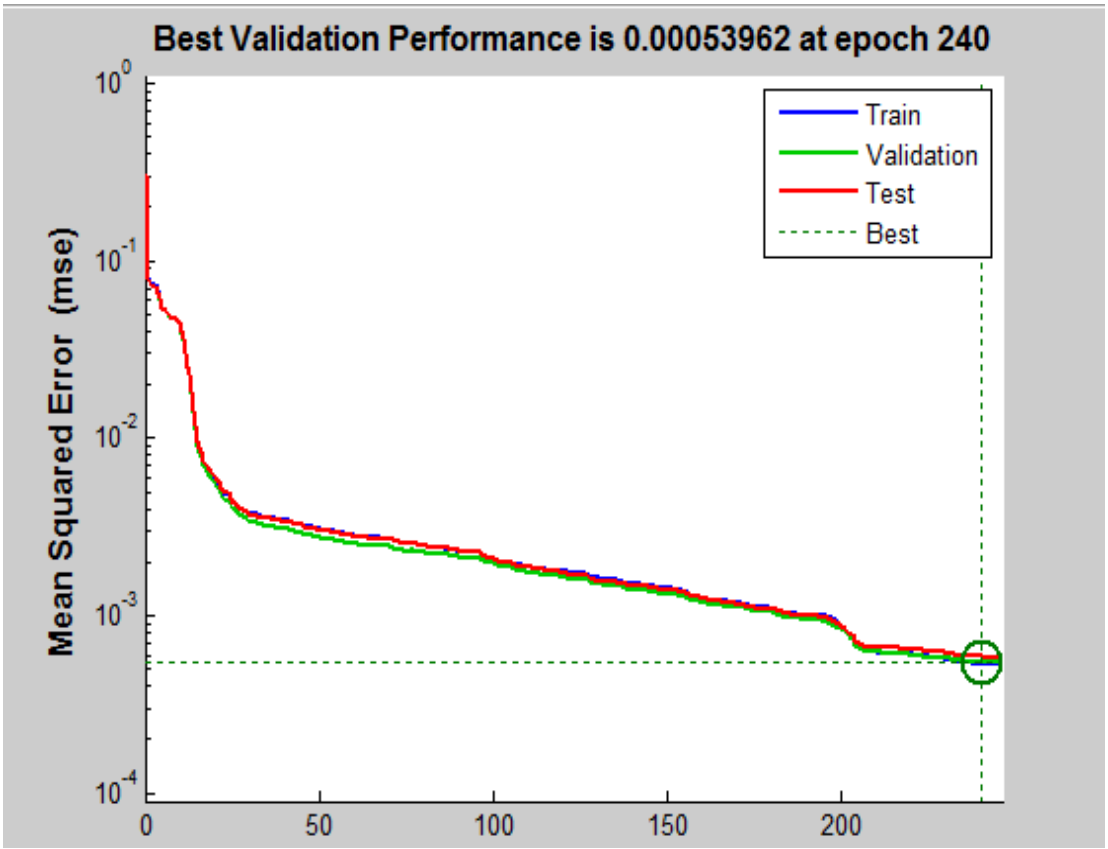


Figure 4.4: Performance of LMBP Algorithm

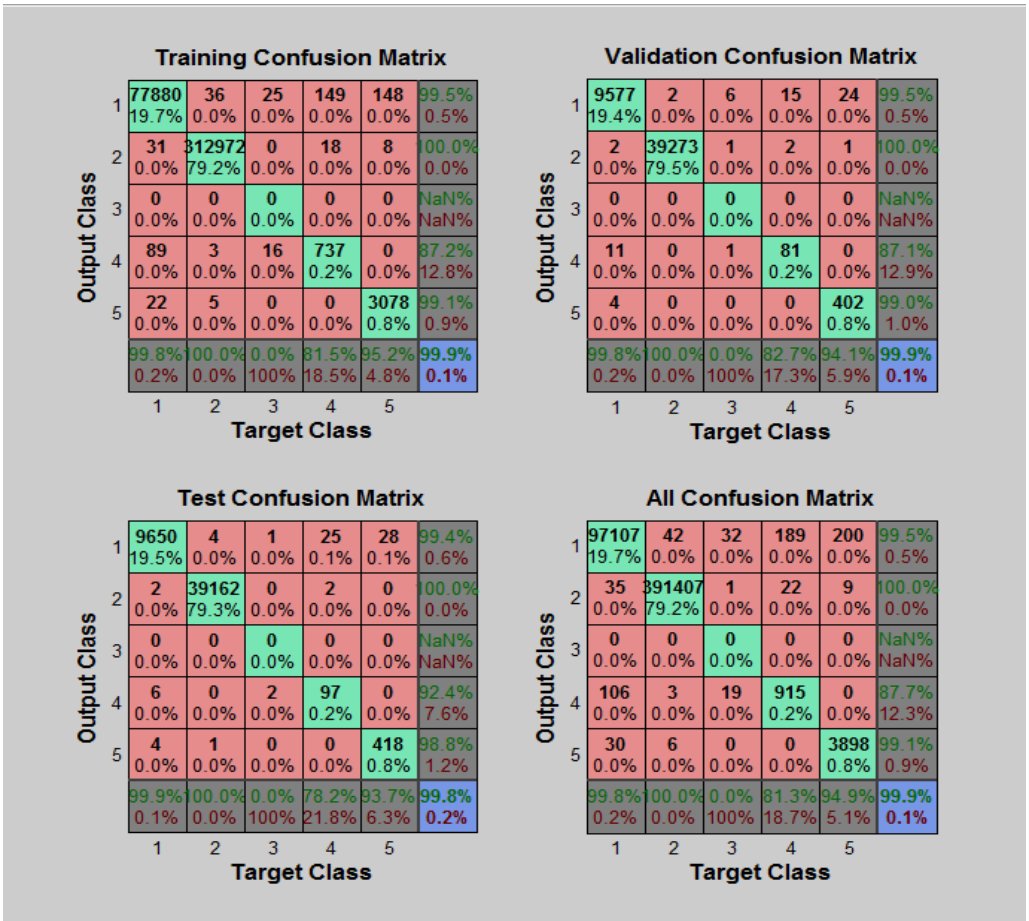


Figure 4.5: Confusion matrix of Levenberg-Marquardt algorithm.

Table 11: Evaluation Results for each Attack Classes (LMBP)

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%

Gradient Descent with Adaptive lr Backpropagation (GDABP):

The Multilayer Perceptron was trained with GDABP algorithm by using default parameters. Simulation result of GDABP algorithm (Figure 4.6) shows the MSE and number of epochs. The best performance is observed at epoch 1000.

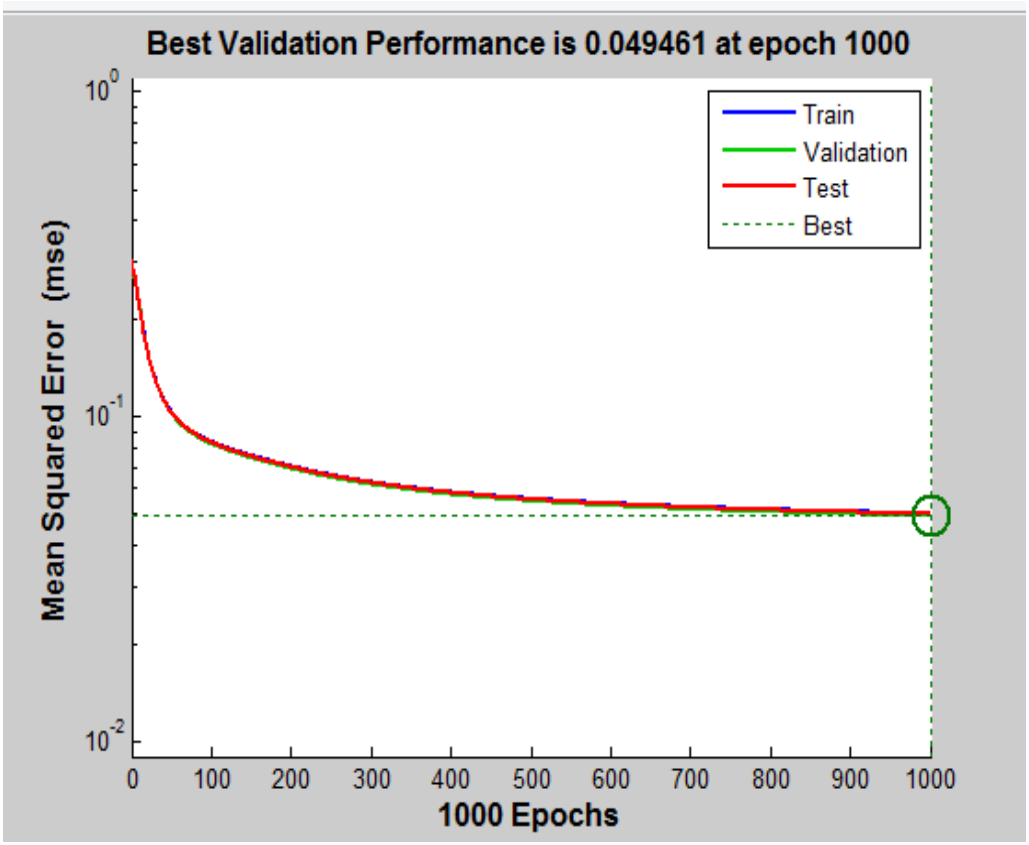


Figure 4.6: Performance of GDABP Algorithm

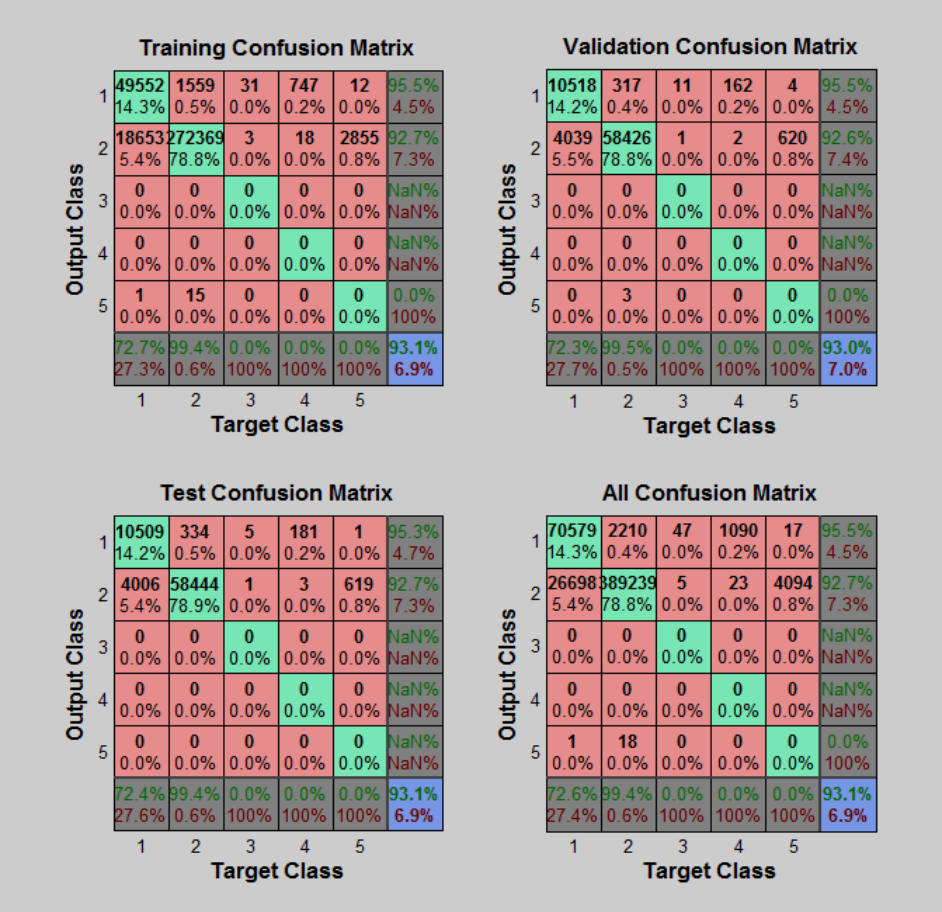


Figure 4.7: Confusion matrix of Gradient Descent with Adaptive lr algorithm.

Table 12: Evaluation Results for each Attack Classes (GDABP)

Attack	**TP**	**FP**	**FN**	**Recall Rate**	**Precision Rate**
DoS	389239	26698	2210	99.4%	93.6%
U2R	0	0	47	0%	0%
R2L	0	0	1090	0%	0%
Probe	0	1	17	0%	0%
Total	389239	26699	3364	99.1%	93.6%

On the basis of above confusion matrix, mathematical relation and other performance parameters the following table is drawn. Table below shows the MSE, Detection Rate, Epoch, Recall, Precision and Time of various BP algorithms.

Table 13: Simulation Result of various BP Algorithms

SN	Algorithm	MSE	Detection Rate	Epoch	Recall	Precision	Time
1	BFGSBP	0.00572	98.3%	79	98.7%	99.97%	14:56
2	LMBP	0.00054	99.9%	240	98.8%	99.95%	15:40
3	GDABP	0.049	97.3%	1000	99.1%	93.6%	28:59

Levenberg-Marquardt Backpropagation has the least mean square error and highest detection rate. BFGS Quasi-Newton was the faster algorithm with higher precision but has lower attack detection rate than Levenberg-Marquardt algorithm. Gradient Descent with adaptive lr back propagation could not improve the performance of the system.

4.2 Comparison

The graphical representation of obtained result is shown below (Figure 4.8 to 4.12).

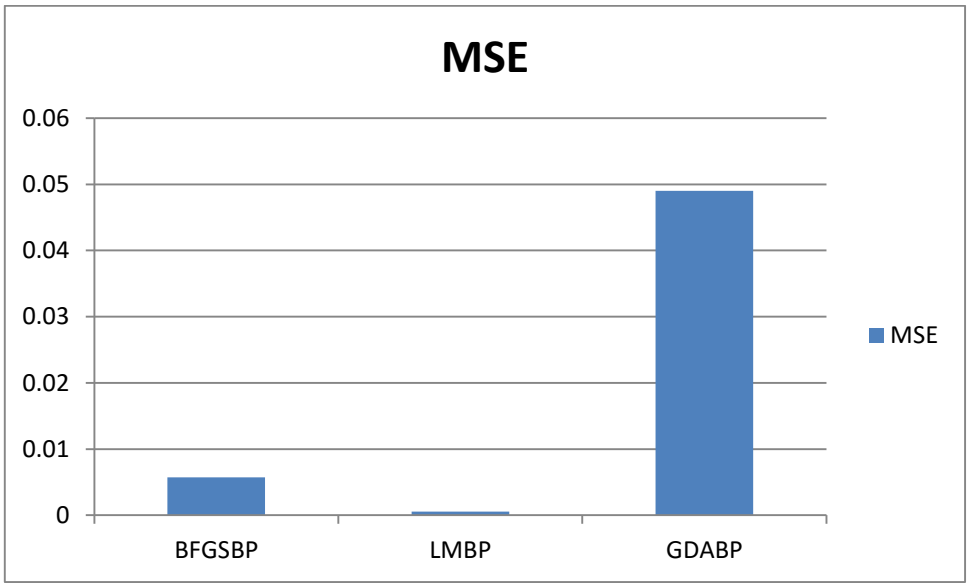


Figure 4.8: Comparison of MSE among BFGSBP, LMBP and GDABP Algorithms

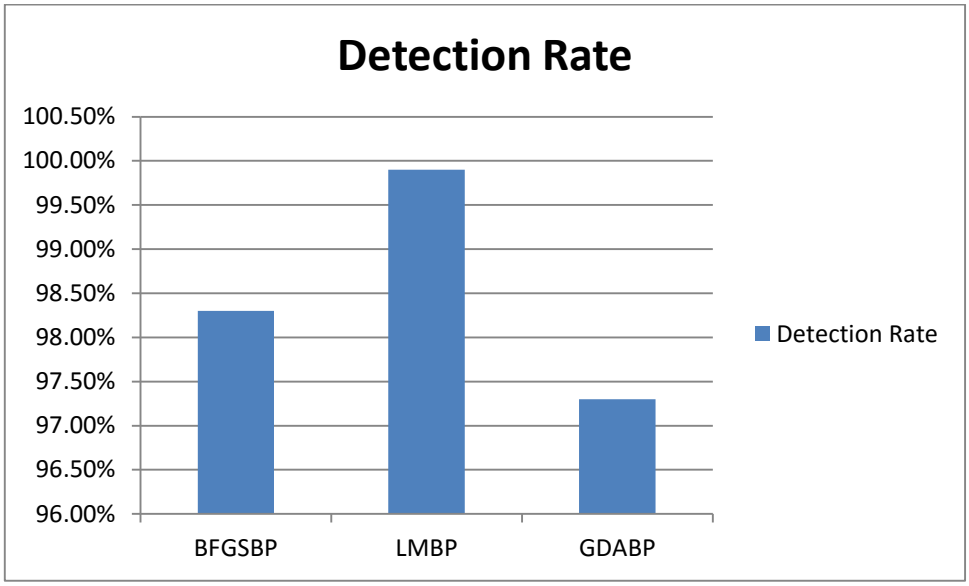


Figure 4.9: Comparison of Detection Rate among BFGSBP, LMBP and GDABP Algorithms

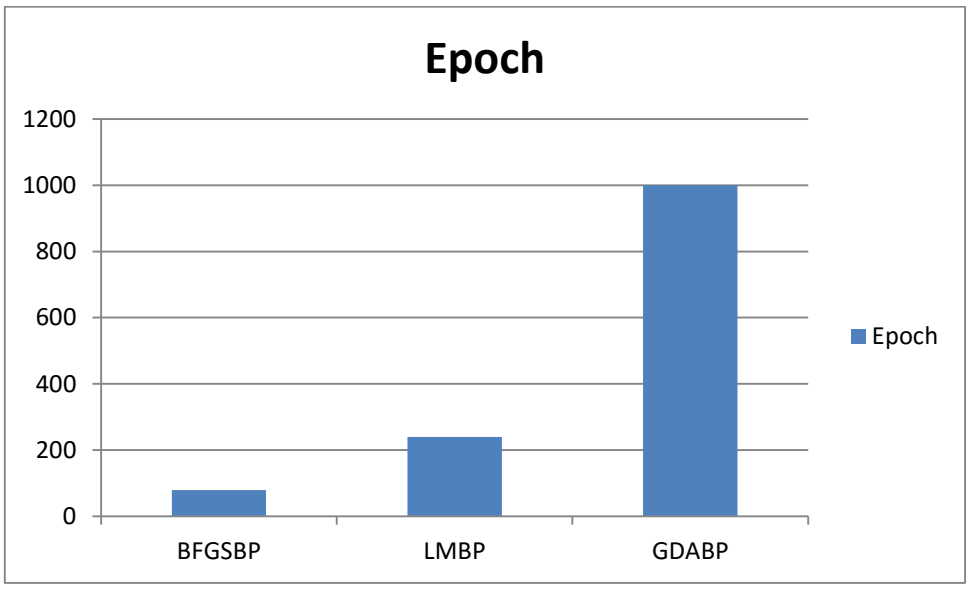


Figure 4.10: Comparison of Epoch among BFGSBP, LMBP and GDABP Algorithms

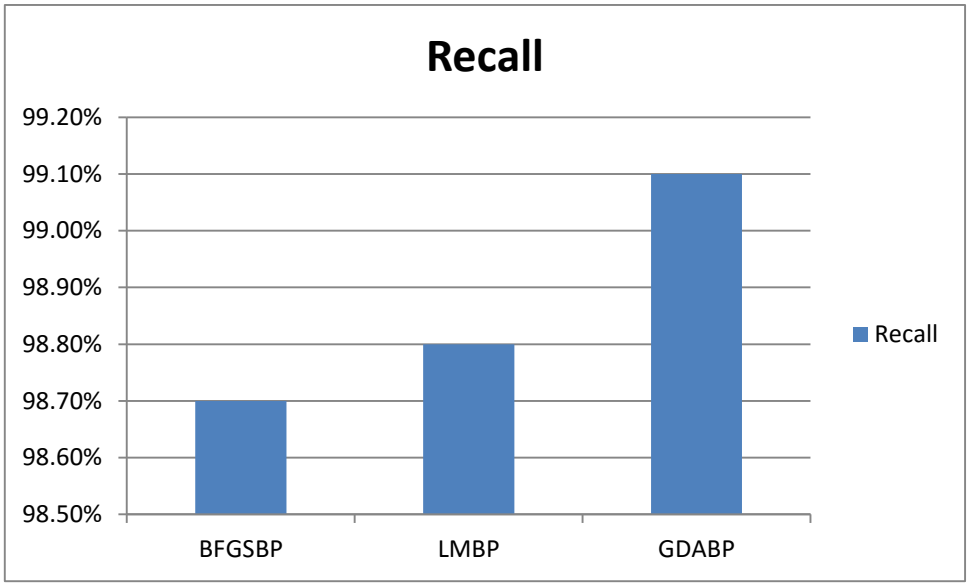


Figure 4.11: Comparison of Recall Rate among BFGSBP, LMBP and GDABP Algorithms

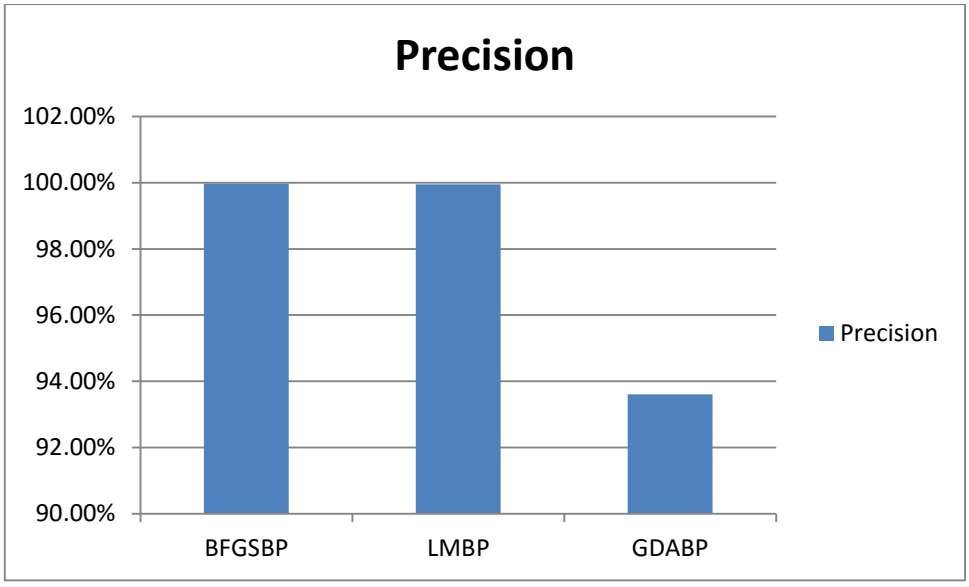


Figure 4.12: Comparison of Precision Rate among BFGSBP, LMBP and GDABP Algorithms

CHAPTER 5: CONCLUSION

Neural networks have the ability to classify patterns, and thus can be used in intrusion detection systems for attack classification. Backpropagation algorithm is used prominently for the training of neural network for its promising features. Different variants of backpropagation algorithms are available, among which I used three algorithms viz. BFGS Quasi-Newton, Levenberg-Marquardt and Gradient Descent with adaptive lr backpropagation. An Intrusion Detection System is designed and the Levenberg-Marquardt Backpropagation algorithm is suggested as the most efficient model for network intrusion detection. Also the type of attack has been identified depending upon the output values.

As a future work, 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). Performance of these algorithms can be analyzed by using real time input dataset. Other various available Backpropagation algorithms can be used to evaluate the performance of the network.

REFERENCES

[1] A. Jaya Lakshmi G. Kalyani, "Performance Assessment of Different Classification Techniques," *IOSR Journal of Computer Engineering (IOSRJCE)*, p. 5, 2012.

[2] XiaoHang Yao, “A Network Intrusion Detection Approach combined with Genetic Algorithm and Back Propagation Neural Network” , IEEE-International Conference on E-Health Networking, Digital Ecosystems and Technologies, 2010.

[3] Jingwen Tian, Meijuan Gao and Fan Zhang, "Network Intrusion Detection Method Based on Radial Basic Function Neural Network", IEEE, 20009

[4] Wang Guojun, Yue Zhiqiang, “Application Research of Support Vector Machine in the Intrusion Detection”, GUANGXI JOURNAL OF LIGHT INDUSTRY, no. 7, pp. 51-52, 2008.

[5] Luan Qinglin, Lu Huibin, “Research of intrusion detection based on neural network optimized by adaptive genetic algorithm”, Computer Engineering and Design, vol. 29,no. 12, pp. 3022-3025, 2008.

[6] Jiao Licheng. Neural network system theory. Xian: Xi an electronic science and technology university press, 1995.

[7] Farah Jemili, Montaceur Zaghdoud and Mohamad Ben Ahmed, “Intrusion Detection based on Hybrid Propagation in Bayesian Networks”, IEEE ISE 2009, Richardson, TX, USA,June 8-11, 2009.

[8] Dr. Richard Spillman, Artificial Intelligence.

http://www.cs.plu.edu/courses/csce330/notes.htm

[9] KDD Cup 1999 dataset. Available on:

http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

BIBLIOGRAPHY

[1] Anderson, James P., "Computer Security Threat Monitoring and Surveillance," Washing, PA, James P. Anderson Co., 1980.

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

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

[4] Michael Negnevitsky, “Artificial Intelligence, A Guide to Intelligent Systems”, Pearson Education, Second Edition, 2005.

[5] Simon Haykin, “Neural Networks”, Pearson Education, 9th Indian Reprint, 2005.

[6] Wolfgang Ertel, ”Introduction to Artificial Intelligence”, Springer, Second Edition, 2009.

[7] Chuck Easttom, “Computer Security Fundamentals”, Pearson Education, Second Edition, 2012.