Modeling Intrusion Detection System by Optimized Selection of ANN Training Algorithms

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering
in
Computer Science & Engineering

Thapar University, Patiala

By:

M. GNANA PRASAD
(80632011)

Under the supervision of:
Dr. V. P. Singh
CSED

JULY 2008

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

Certificate

I hereby certify that the work which is being presented in the thesis report entitled, **'Modeling Intrusion Detection System by Optimized Selection of ANN Training Algorithm',** submitted by me in partial fulfillment of the requirements for the award of degree of Master of Engineering in Computer Science and Engineering submitted in the Department of Computer Science and Engineering of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of Dr. V. P. Singh and refers other researcher's works which are duly listed in the reference section.

The matter presented in this thesis has not been submitted for the award of any other degree of this or any other university.

(M. Gnana Prasad)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

(Dr. V. P. Singh)
Department of Computer Science & Engineering
Thapar University
Patiala

Countersigned by

(Dr. SEEMA BAWA)
Professor & Head
Computer Science & Engineering, Department
Thapar University
Patiala

(Dr. R.K.SHARMA)
Dean(Academic Affairs)
Thapar University,
Patiala.

Acknowledgment

I express my sincere and deep gratitude to my guide Dr. V. P. Singh, Department of Computer Science & Engineering, for the invaluable guidance, support and encouragement. He provided me all resource and guidance throughout of the thesis work.

I am thankful to Dr. (Mrs.) Seema Bawa, Head of Department of Computer Science & Engineering, Thapar University, Patiala, for providing me adequate environment, and facility for carrying thesis work.

I would like to like to say thanks from deep inside my heart to my friends who had helped me at every moment of my study and is constantly guiding me and strengthening me. They are with me all the time.

M. Gnana Prasad

(80632011)

.

ABSTRACT

With the rapid expansion of computer networks during the past decade, security has become a crucial issue for computer systems. Different soft-computing based methods have been proposed in recent years for the development of intrusion detection systems. This paper presents a neural network approach to intrusion detection**.** The purpose of this work is to design, implement and evaluate an anomaly based intrusion detection system and to select the optimized training algorithm by validating the performance of different neural network training functions for IDS. Intrusion Detection System is a detection mechanism that detects unauthorized, malicious presence in the computer systems.

In this work the user behavior is taken as parameter to detect the intrusions. The neural network learns about the normal user's behavior form the network traffic that only contains information about normal users. The proposed Intrusion Detection System in this work uses a Back Propagation neural network to learn user's behavior. An intrusion detection system is divided into two phases: learning and detection. In learning phase, the system learns about the normal user's or system's behavior. In the detection phase, system detects the intruder's by matching their behavior with that of normal user's. It is obvious that the intruder's behavior is different than that of normal user's.

The 1998 DARPA Intrusion Detection Evaluation data sets collected from the literature review are used in this work. For the training of the neural network 220 sessions of traffic is used. Out of these there were 112 sessions with normal traffic and 108 sessions with attacks. When the learning is over, the system is tested with the network traffic that contains attacks and normal data. 100 sessions of traffic to test the trained network is used, Out of these 50 sessions with normal traffic and 50 sessions with attacks.

Keywords: Intrusion Detection, Artificial Neural Networks, Back propagation, Anomaly based system.

Contents

List of Figures

List of Tables

List of Abbreviations

IDS	Intrusion Detection System
ANN	Artificial Neural Networks
NIDS	Network Intrusion Detection System
HTTPS	Hypertext Transfer protocol Secure
SQL	Structured Query Language
DoS	Denial of Service
LAN	Local area Network

Chapter 1: Introduction

1.1 Introduction

Internet has almost become a “new world”, and as in the real world the “new world” has criminals and vandals. The big threat of vandalism and theft has given users a need for security components to protect themselves. The detection methods of intruders in the computer networks have drawn attention of many researchers in recent years.

An intrusion detection system is an important component of the computer and information security framework. Its main goal is to differentiate between normal activities of the system and behaviors that can be classified as suspicious or intrusive. The work aims at finding optimized ANN train algorithm to design, implement and evaluate an anomaly based intrusion detection system. In this work the user behavior is taken as parameter to detect the intrusions. In this system, neural network is used to learn about the normal user’s behavior to form the network traffic that only contains information about normal user’s. When the learning is over, the system is tested with the network traffic that contains attacks and normal data.

Artificial neural network is a kind of information processing paradigm that is inspired by the biological nervous systems, such as the brain, to process information. It tries to represent the physical brain and thinking process by means of an electronic circuit or software. Artificial neural network is the network of individual neurons. Each neuron in a neural network acts as an independent processing element. Like human or other brains, neural networks also learn by examples or training, and they cannot be programmed to perform a specific task. It can be configured for any specific application with learning process. Neural networks perform very successfully for recognizing and matching complicated, vague, or incomplete patterns. The most successful applications of neural network are classification and pattern recognition.

Intrusion detection tools seek to detect attacks against computer systems by monitoring

the behavior of users, networks, or computer systems. The goal for this research is to develop an Intrusion Detection System that is able to detect both known and unknown attacks by using the neural networks learning ability.

1.2 Motivations and Aim

Neural networks are specified with many small processors working simultaneously on the same ask. It has the ability to 'learn' from training data and use its 'knowledge' to compare patterns in a data set. The aim of this research is to design, implement, and evaluate an Anomaly based Network Intrusion Detection Systems and to compare performance of different neural network training functions which one works better for IDS. This project describes an artificial neural network based system for network anomaly detection. The system will take network traffic data to analyze and classify the behavior of the authorized users and to recognize the likely attacks. In this work only offline detection of attacks has to be done. The steps that has been performed:

- Giving input to the neural network (DARPA datasets are used). .
- Classifying the user's behavior from the input data using backpropagation neural network with different ANN training algorithms. i.e. Training the neural network.
- Detecting/testing the neural network for the likely attacks with different ANN train algorithms.

This proposed work detects effectively on the Misuse (known) attacks and capabilities of monitoring the Anomaly (unknown) attacks.

1.2 Disposition

Chapter 2 introduces the state art of intrusion detection system

Chapter 3 introduces the state art of artificial neural network and the literature based on intrusion detection system is also surveyed here.

Chapter 4 describes approach to solve the present problem and importance of neural network to solve the present problem.

Chapter 5 provides the details of the training algorithm that has been used, architecture of the neural network, converting the data sessions into binary format i.e. making input files to the neural network.

Chapter 6 discusses about the results of training and testing of the neural network.

Chapter 7 draws conclusions and further enhancement of work.

CHAPTER 2: Intrusion Detection System

INTRODUCTION

This chapter describes intrusion detection in detail, provides the various kinds of attacks that system user's face in daily life and the need for security computer systems and also discusses the role of intrusion detection in their security.

2.1 Intrusion detection system

Generally speaking, an intrusion detection system is a tool for detecting abnormal behavior in a system. An abnormal pattern covers many definitions but in general it is likely described as unwanted, malicious and/or misuse activity occurring within a system. The two main techniques of intrusion detection are called misuse detection and anomaly detection, which we are going to discuss detail in this chapter

2.1.1 Background

In today's life with the development of computer and computer networks, network-based computer systems play increasingly vital roles in modern society. In addition to intrusion prevention techniques, such as user authentication and authorization, encryption, and defensive programming, Intrusion detection is clearly necessary with the growing number of computer systems being connected to networks.

Because of the increasing dependence on the network based computer systems, the importance of protecting these systems from attack is critical. A single intrusion of a computer network can result in the loss or unauthorized utilization or modification of large amounts of data and cause users to question the reliability of all of the information on the network. There are numerous methods of responding to a network intrusion, but they all require the accurate and timely identification of the attack [1].

Generally speaking, an intrusion detection system is a tool for detecting abnormal behaviors in a system. An abnormal pattern covers many definitions but in general it is likely described as unwanted, malicious and/or misuse activity occurring within a system.

The problem of intrusion detection has been studied for several years with early papers on the subject appearing in the late 1970s and early 1980s. While the definition of an intrusion varies slightly from paper to paper, definitions such as the following are widely accepted:

“any set of actions that attempt to compromise the integrity, confidentiality or availability of a resource ”.

An intrusion detection system then is a system, which attempts to detect and in some cases react to intrusions, whether on one system, group of systems, or computer network. Early research in this topic developed two classification systems for different intrusion detection systems, anomaly detection and signature or misuse detection.

2.1.2 Misuse detection systems

These use patterns of known attacks or weak spots of the system to match and identify intrusions. For instance, if someone tries to guess a password, a signature rule for this kind of behavior could be that too many failed login attempts within some time’ and this event would result in an alert. Misuse detection is not effective against unknown attacks that have no matched rules or patterns yet [2]. The main disadvantage of misuse detection approaches is that they will detect only the attacks for which they are trained to detect. Novel attacks or unknown attacks or even variants of common attacks often go undetected.

At a time when new security vulnerabilities in software are discovered and exploited every day, the reactive approach embodied by misuse detection methods is not feasible for defeating malicious attacks [3].

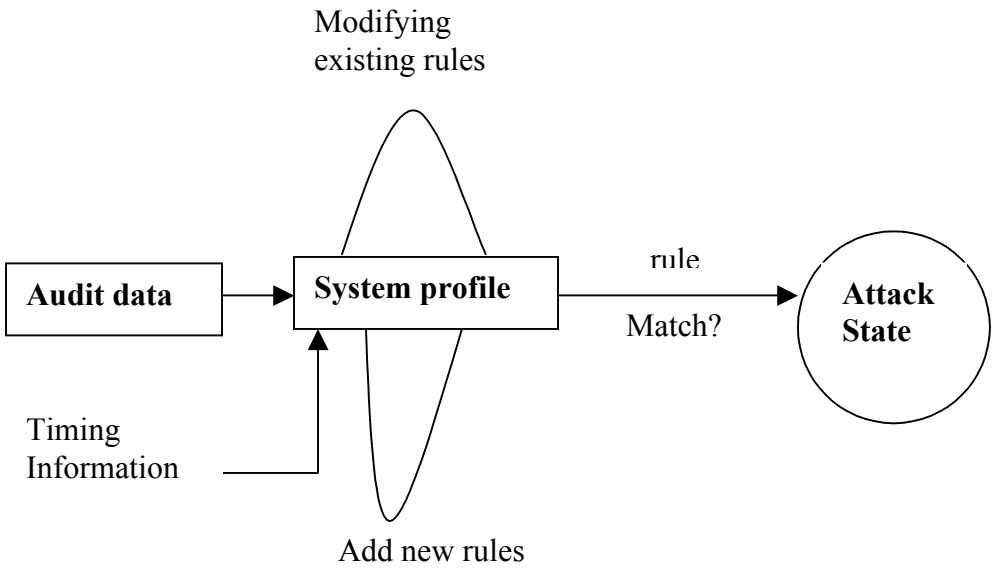

Figure 2.1: Typical Misuse detection system

2.1.3 Anomaly detection systems

These systems flags observed activities that deviate significantly from the established normal usage profiles as anomalies. For instance a profile of a user may contain the averaged frequencies of some system commands in his or her logging sessions. For a logging session that is being monitored if it has significantly lower or higher frequencies an anomaly alert will be raised. Anomaly detection is an effective technique for detecting novel or unknown attacks since it does no require knowledge about intrusion attacks. But at the same time it tends to raise more alerts than misuse detection because whatever event happens in a session, normal or abnormal behavior, if its frequencies are significantly different from the averaged frequencies of the user it will raise an alert. The main advantage of anomaly detection approaches is the ability to detect novel attacks or unknown attacks against software systems, variants of known attacks, and deviations of normal usage of programs regardless of whether the source is a privileged internal user or an unauthorized external user [3].

The disadvantage of the anomaly detection approach is that well-known attacks may not be detected, particularly if they fit the established profile of the user. Once detected, it is often difficult to characterize the nature of the attack for forensic purposes. Another drawback of many anomaly detection approaches is that a malicious user who knows that he or she is being profiled can change the profile slowly over time to essentially train the anomaly detection system to learn the attacker's malicious behavior as normal.
Finally a high false positive rate may result for a narrowly trained detection algorithm, or conversely, a high false negative rate may result for a broadly trained anomaly detection approach.

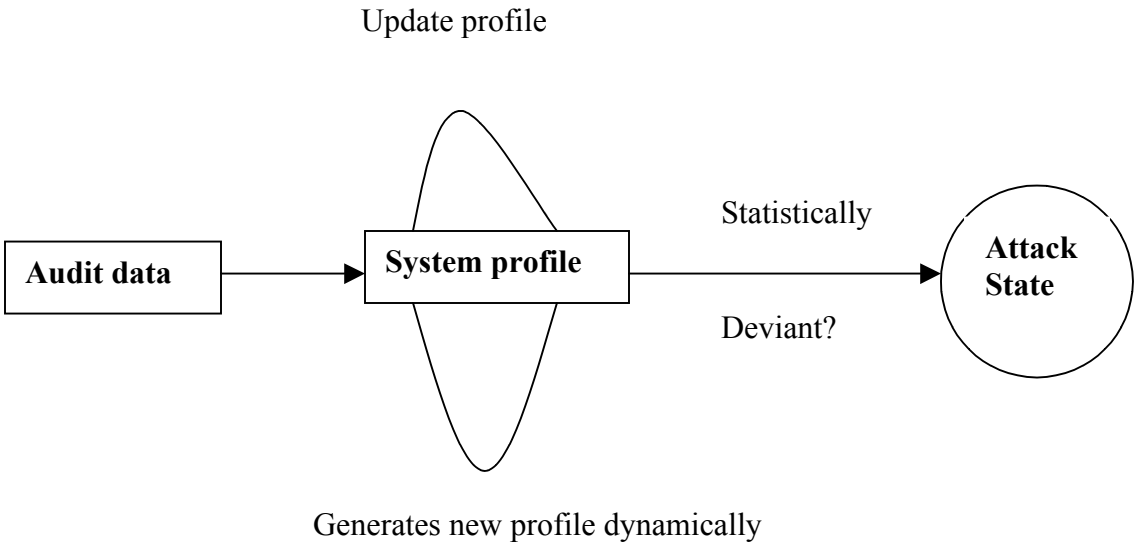

Figure 2.2: Typical Anomaly detection system

As mentioned intrusion detections can be deployed on different areas, like within a computer to spot users attempting to gain access to which they have no access right, or monitoring network traffic to detect other kind of intrusions like worms, Trojan horses or to take control of a host by yielding an illegal rootshell, etc.

2.1.4 Types of Intrusion-Detection Systems

In a network-based intrusion-detection system (NIDS), the sensors are located at choke points in the network to be monitored, at the network borders. The sensor captures all network traffic and analyzes the content of individual packets for malicious traffic. In

systems, a protocol-based intrusion detection system (PIDS) and application protocol-based intrusion detection system (APIDS) are used to monitor the transport and protocols illegal or inappropriate traffic or constructs of language (say SQL). In a host-based system, the sensor usually consists of a software agent, which monitors all activity of the host on which it is installed [4].

A network intrusion detection system is an independent platform, which identifies intrusions by examining network traffic and monitors multiple hosts. Network Intrusion Detection Systems gain access to network traffic by connecting to a hub, network switch configured for port mirroring, or network tap. An example of a NIDS is Snort.

A protocol-based intrusion detection system consists of a system or agent that would typically sit at the front end of a server, monitoring and analyzing the communication protocol between a connected device (a user/PC or system). For a web server this would typically monitor the HTTPS protocol stream and understand the HTTP protocol relative to the web server/system it is trying to protect. Where HTTPS is in use then this system would need to reside in the "shim" or interface between where HTTPS is un-encrypted and immediately prior to it entering the Web presentation layer.

- An application protocol-based intrusion detection system consists of a system or agent that would typically sit within a group of servers, monitoring and analyzing the communication on application specific protocols. For example in a web server with database this would monitor the SQL protocol specific to the middleware/business-login as it transacts with the database.
- A host-based intrusion detection system consists of an agent on a host, which identifies intrusions by analyzing system calls, application logs, user's behavior and other host activities and state.
- A hybrid intrusion detection system combines two or more approaches. Host agent data is combined with network information to form a comprehensive view of the network. An example of a Hybrid IDS is Prelude.

Intrusion Detection Systems also differ in whether they are on-line or off-line. Off-line Intrusion Detection Systems are run periodically and they detect intrusions after-the-fact based on system logs. On-line systems are designed to detect intrusions while they are happening, thereby allowing for quicker intervention.

2.2 Network Intrusions

2.2.1 Types of Attacks

In today's life each and every user of Internet will be facing some problem because of unknown attacks that occur while using the Internet on his computer. Attack types are mainly classified into four types.

Types of attacks:

- Probing: surveillance and other probing.
- DoS: denial of service.
- U2Su: unauthorized access to local super user (root) privileges.
- R2L: unauthorized access from a remote machine.

2.2.1.1 Probing

Probing is a class of attacks where an attacker scans a network to gather information or find known vulnerabilities. An attacker with a map of machines and services that are available on a network can use the information to look for exploits. There are different types of probes: some of them abuse the computer's legitimate features, some of them use social engineering techniques. This class of attacks is the most commonly heard and requires very little technical expertise.

2.2.1.2 Denial of Service Attacks

Denial of Service (DoS) is a class of attacks where an attacker makes some computing or memory resource too busy or too full to handle legitimate requests, thus denying legitimate users access to a machine [7]. There are different ways to launch DoS attacks:

- Abusing the computers legitimate features.
- Targeting the implementations bugs.
- Exploiting the system's misconfigurations.

DoS attacks are classified based on the services that an attacker renders unavailable to legitimate users.

Examples of DoS attack:

1. Ping broadcast - A ping request packet is sent to a broadcast network address where there are many hosts. The source address is shown in the packet to be the IP address of the computer to be attacked. If the router to the network passes the ping broadcast, all computers on the network will respond with a ping reply to the attacked system. The attacked system will be flooded with ping responses which will cause it to be unable to operate on the network for some time, and may even cause it to lock up. The attacked computer may be on someone else's network. One countermeasure to this attack is to block incoming traffic that is sent to a broadcast address.
2. Ping of death - An oversized ICMP datagram can crash IP devices.
3. Smurf - An attack where a ping request is sent to a broadcast network address with the sending address spoofed so many ping replies will come back to the victim and overload the ability of the victim to process the replies.
4. Land Exploit - is a DoS attack in which a program sends a TCP SYN packet where the target and source addresses are same and the port numbers are same. Most machines will

Attack Type	**Effect of the attack**
Process table	Denies new processes
Land	Freezes the machine
Smurf	Slows down the network
Teardrop	Reboots machine
Upstrom	Slows down the network
Ping of Death	Crash IP devices

Table 2.1: Examples of DoS attacks

2.2.1.3 User to Root Attacks

User to root exploits are a class of attacks where an attacker starts out with access to a normal user account on the system and is able to exploit vulnerability to gain root access to the system. Most common exploits in this class of attacks are regular buffer overflows, which are caused by regular programming mistakes and environment assumptions.

Attack Type	Effect of the attack
Eject	Gains root shell using Buffer Overflow
Xterm	Gains root shell using Buffer Overflow
Fdformat	Gains root shell using Buffer Overflow

Table 2.2: Examples of User to Root attacks

2.2.1.4 Remote to User Attacks

A remote to user (R2L) attack is a class of attacks where an attacker sends packets to a machine over a network, then exploits machine's vulnerability to illegally gain local access as a user. There are different types of R2U attacks; the most common attack in this class is done using social engineering.

Attack Type	Effect of the attack
Dictionary	Gain user access
Xlock	Spoof user to obtain password
Xnsoop	Monitor key strokes remotely

Table 2.3: Examples of user to Root attacks

2.2.2 Trojan Horses

A Trojan Horse is an illegal computer program disguised as legal, or hidden as part of a legal program. It can be described as a secret defect (or trap) that is intentionally inserted into legal software. The Trojan Horse can attack almost all programs, from basic systems software to users' application software. When the Trojan Horse is installed on the victims computer, it is often used to

- Propagate a virus or a worm
- Install a backdoor
- Destroy data

When it is installed, the Trojan Horse gives the intruder access to the data stored on the victim's computer. It can also give the attacker access to other computers if the victim's computer is in a local network.

2.2.3 Viruses and Worms

Even though a virus is not actually an attack method, it causes much damage and is expensive and time consuming so it should be mentioned. Viruses and worms are malicious codes made to do some damage on the infected system. 85% of the respondents in the FBI/CSI survey reported virus and worm outbreaks. Viruses and worms exploit vulnerabilities in the system, and large numbers of systems can be infected within a matter of hours.

Computer Viruses started to spread through floppy disks on Apple computers as early as in 1981. They started to appear in large number in 1987, apparently starting in Pakistan, Israel and Germany, and later appearing through the whole world. This caused thousands of computers to become unusable for short periods of time, hundreds of thousands computers to display spurious messages, tens of thousands of users to experience denial of services an several international networks to experience denial of service for a short period of time.

A decade ago, viruses were relatively easy to find and fix, and they spread slowly, generally by floppy disks or LANs. Now, however, increasingly 41 creative authors are

exploiting the Internet, open-source software, peer-to-peer technology, and other developments to write viruses and worms that invade computer systems in new ways, propagate around the world quickly, and wreak havoc to victims. During a virus' lifetime, it normally goes through 4 stages. These stages are:

- Dormant phase: The virus is idle, waiting to be activated.
- Propagation phase: Replicating itself to programs or disk.
- Triggering phase: The virus is activated to do its tasks by some event such as time, date, and number of replications.
- Execution phase: The function in the virus is performed.

The detection of new viruses has become very difficult. Virus writing has gone to a new level where the viruses are polymorphic, uses changing encryption and decryption, and can infect both Windows and Linux platforms. They infect machines not only by using their own code, but also by linking to and accessing malicious codes from newsgroups and Web sites. New software from different vendors is out now that requires users to define which actions they will and will not allow on a computer or network. Joe Hartman from Trend Micro said: "If a machine suddenly starts to send hundreds of e-mails, the software will know that something is wrong and notify the user or system administrator".

2.2.4 Honeynets

A non-profit organization called "The Honeynet Project" has dedicated them selves to find out more about intruders behavior and how they work. The group gathers information by deploying networks, called Honeynet. These networks are real networks with all the hardware that are needed. These honeynets lures the hackers to a system and then analyze their activities.

The intent is, for attackers to break into the system and have every action captured and controlled without them knowing it. Each computer in the Honeynet is called a honeypot. The concept of honeynets is pretty simple [6]. Honeynet is a type of honeypot, more specifically, a high interaction honeypot designed for research purposes where the typical honeypot is meant for deception and detection. The major difference with a honeynet is that it is a network of multiple systems and applications. By using multiple systems like

Solaris, Linux and Windows, it creates an environment much like a production network. Also, by having different systems, a honeynet can target a larger array of blackhats. The other difference between a honeynet and a honeypot is that a honeynet are real systems, not emulated like a honeypot.. Honeynets have two critical requirements. The following are the two requirements

- Data control: To ensure that once an attacker breaks into the honeynet system, the compromised system cannot be used to attack or harm other systems.
- Data capture: To ensure that all the attackers' activities, even if they are obfuscated or encrypted, are detected and captured.

CHAPTER 3: Artificial Neural Networks

INTORDUCTION

This chapter describes Artificial Neural Networks in detail, it discusses about the importance of neural networks to the intrusion detection system and the earlier research carried on the intrusion detection system.

3.1 Background

Neural network simulations appear to be a recent development. However, this field was established before the advent of computers, and has survived at least one major setback and several eras.

Many import and advances have been boosted by the use of inexpensive computer emulations. Following an initial period of enthusiasm, the field survived a period of frustration and disrepute. During this period when funding and professional support was minimal, relatively few researchers made important advances. Currently, the neural network field enjoys a resurgence of interest and a corresponding increase in funding.
The first artificial neuron was produced in 1943 by the neurophysiologist Warren McCulloch and the logician Walter Pits [7]. But the technology available at that time did not allow them to do too much.

Most of the ANN structures used commonly for many applications often consider the behavior of a single neuron as the basic computing unit for describing neural information processing operations. Each computing unit, i.e. the artificial; neuron in the neural network is based on the concept of an ideal neuron. An ideal neuron is assumed to respond optimally to the applied inputs.

3.2 How the Human Brain Learns

Artificial Neural Networks success at system modeling for highly complex physical processes can be attributed to the original architecture on which they are based, the

human brain. At present, brain function is not fully understood. A brain neuron collects signals from other neurons of the Central Nervous System (CNS), through structures called dendrites. The neuron sends out spikes of electrical activity through a long thin strand called an axon. This axon splits into thousands of branches. At the end of a branch, a structure called a synapse converts the activity from the axon into electrical effects. They may excite or inhibit activity in the connected neurons. When a neuron receives an excitatory input that is sufficiently large compared with its inhibitory input, it sends a spike of electrical activity down its axon [6].

Learning occurs by changing the effectiveness of the synapses so that the influence of one neuron on another changes. The structure of the human brain neuron is the template for artificial learning. However, lack of knowledge leads to approximations and assumptions of the general architecture of an artificial neural network. The knowledge of neurons is incomplete and computing power is limited so models are often idealizations of real networks of neurons.

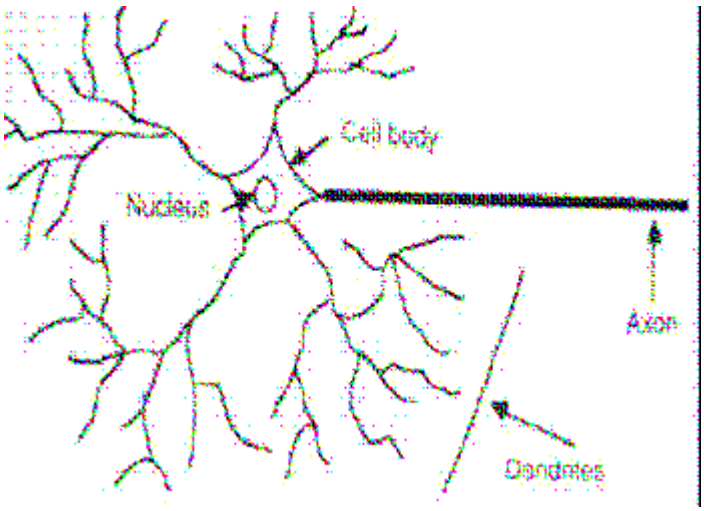

Figure 3.1: Biological neuron

The aim of neural networks is to mimic the human ability to adapt to changing circumstances and the current environment. This depends heavily on being able to learn from events that have happened in the past and to be able to apply this to future

situations. Artificial neural networks consist of many nodes, i.e. processing units analogous to neurons in the brain. Each node has a node function, associated with it, which along with a set of local parameters may alter the node function. An artificial neural network thus is an information-processing system. In this information-processing system, the elements called neurons, process the information. The signals are transmitted by means of connection links [8]. The link possesses an associated eight, which is multiplied along with the incoming signal for any typical neural net. The output signal is obtained by applying activations to the net input.

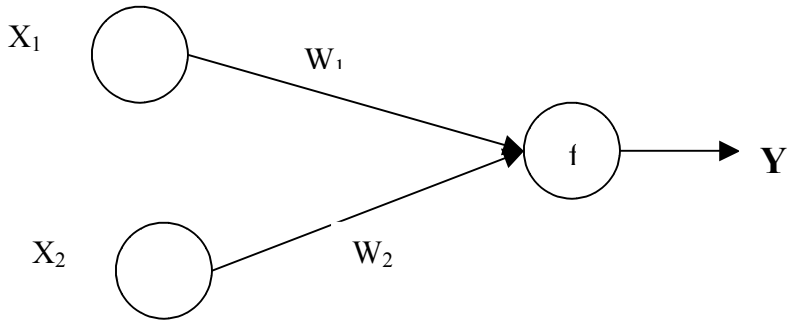

Figure 3.2: A Simple Artificial Neural Net

The neural net can generally be a single layer or a multi-layer net. The structure of the simple artificial neural net is shown in above figure. The figure 3.2 shows artificial neural net with two input neurons (x_1, x2) and one output neuron (y). The interconnected weights are given by w_1 and w_2. In a single layer net there is a single layer of weighted interconnections.

Artificial neural network (in the present context, multilayer, feed forward type networks) consists of a collection of highly interconnected processing elements to perform an input-output transformation. The actual transformation is determined by the set of weights associated with the links connecting elements. By modifying the connections between the nodes the network is able to adapt to the desired outputs.

The neural network gains knowledge about the transformation to be performed by iteratively learning from a sufficient training set of samples or input-output training pairs.

A well-trained network can perform the transformation correctly and also possess some generalization capability.

3.3 Learning Methods

In general, learning is a relatively permanent change in behavior brought about by experience. Learning in neural networks is a more direct process, and we typically can capture each learning step in a distinct cause-effect relationship. The knowledge of a neural network is stored in the synapses, which are the weights of the connections between the neurons. These weights between two layers of neuron can be represented as matrices. If the neural network with the proper learning algorithm, which is determined by the analysis and preprocessing of the data, can produce a reasonable prediction on how much it is going to rain the next year [9]. Another field, that is quite more interesting and way more challenging, is the stock market. Undoubtedly, investors will be interested in knowing how the value of a stock is going to develop in the short and long term. Based on the former developments of the stock for the passed years, a neural network model can be trained to predict when a stock will yield the largest profit. But the stock market is a very complex field and it is doubtful that such a model will be invented.

The definition of the learning process implies the following sequence of events:

1. The neural network is stimulated by an environment.
2. The neural network undergoes changes in its free parameters as a result of the stimulation.
3. The network responds in a new way to the environment due to the changes that occurred in its internal structure.

There are numerous algorithms available and as one would expect there is no unique algorithm for designing a neural network model. The difference between the algorithms lies in formulation of how to alter the weights of the neurons and in the relations of the neurons to their environment.

All learning methods can be classified into two major categories:

Supervised learning which incorporates an external teacher [10], so that each output unit is told what its desired response to input signals ought to be. During the learning process global information may be required. Paradigms of supervised learning include error-

correction learning, reinforcement learning and stochastic learning. Important issue concerning supervised learning is the problem of error convergence, i.e. the minimization of error between the desired and computed unit values. The aim is to determine a set of weights, which minimizes the error. One well-known method, which is common to many learning paradigms, is the least mean square (LMS) convergence.

Unsupervised learning uses no external teacher and is based upon only local information [10]. It is also referred to as self-organization, in the sense that it self-organizes data presented to the network and detects their emergent collective properties. Hebbian learning and competitive learning are the paradigms of unsupervised learning. We say that a neural network learns off-line if the learning phase and the operation phase are distinct. A neural network learns on-line if it learns and operates at the same time. Usually, supervised learning is performed off-line, whereas unsupervised learning is performed on-line.

3.4 Neuron Model

An elementary neuron with *inputs* is shown below. Each input is weighted with an appropriate *w*. The sum of the weighted inputs and the bias forms the input to the transfer function *f*. Neurons may use any differentiable transfer function to generate the output.

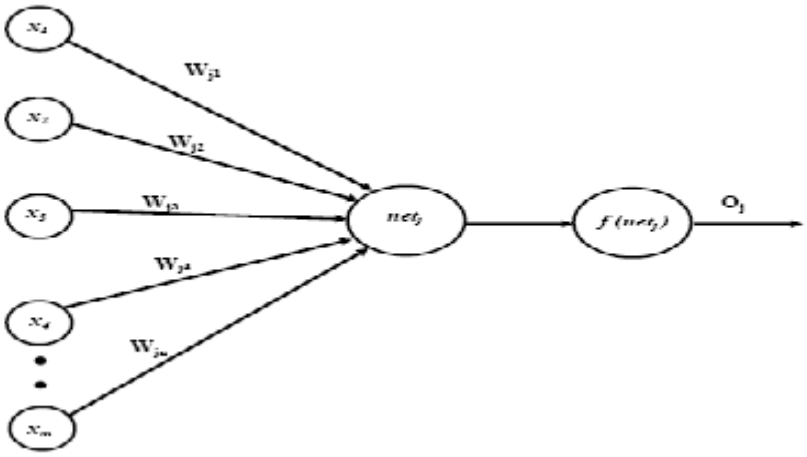

Figure 3.3: Generic model of ANN

Multilayer networks often use the log-sigmoid transfer function logsig [12]

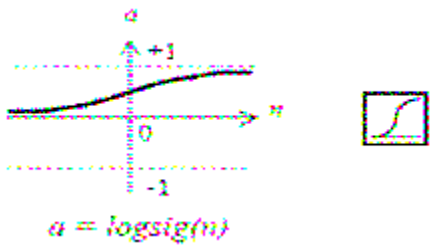

Figure 3.4: Log-Sigmoid Transfer Function

The function logsig generates outputs between 0 and 1 as the neuron's net input goes from negative to positive infinity. Alternatively, multilayer networks may use the tan-sigmoid transfer function tansig.

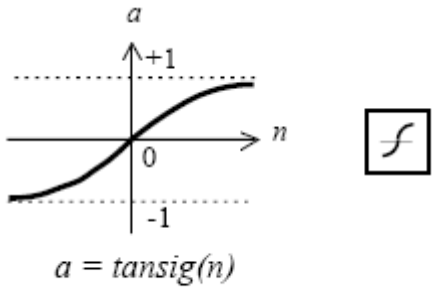

Figure 3.5: Tan-Sigmoid Transfer Function

Occasionally, the linear transfer function purelin is used in backpropagation networks

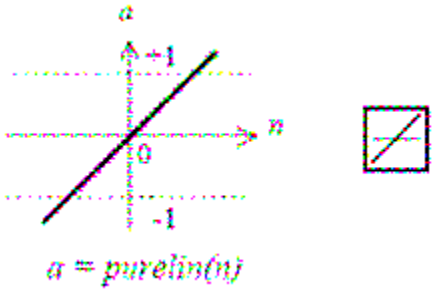

Figure 3.6: Linear Transfer Function

3.5 Feedforward Network

The feedforward neural network was the first and arguably simplest type of artificial neural network devised. In this network, the information moves in only one direction, forward, from the input nodes, through the hidden nodes (if any) and to the output nodes. There are no cycles or loops in the network [9].

3.5.1 Single-layer Perceptron

The earliest kind of neural network is a single-layer perceptron network, which consists of a single layer of output nodes, the inputs are fed directly to the outputs via a series of weights. In this way it can be considered the simplest kind of feed-forward network. The sum of the products of the weights and the inputs is calculated in each node, and if the value is above some threshold (typically 0) the neuron fires and takes the activated value (typically 1); otherwise it takes the deactivated value (typically -1). Neurons with this kind of activation function are also called artificial neurons.

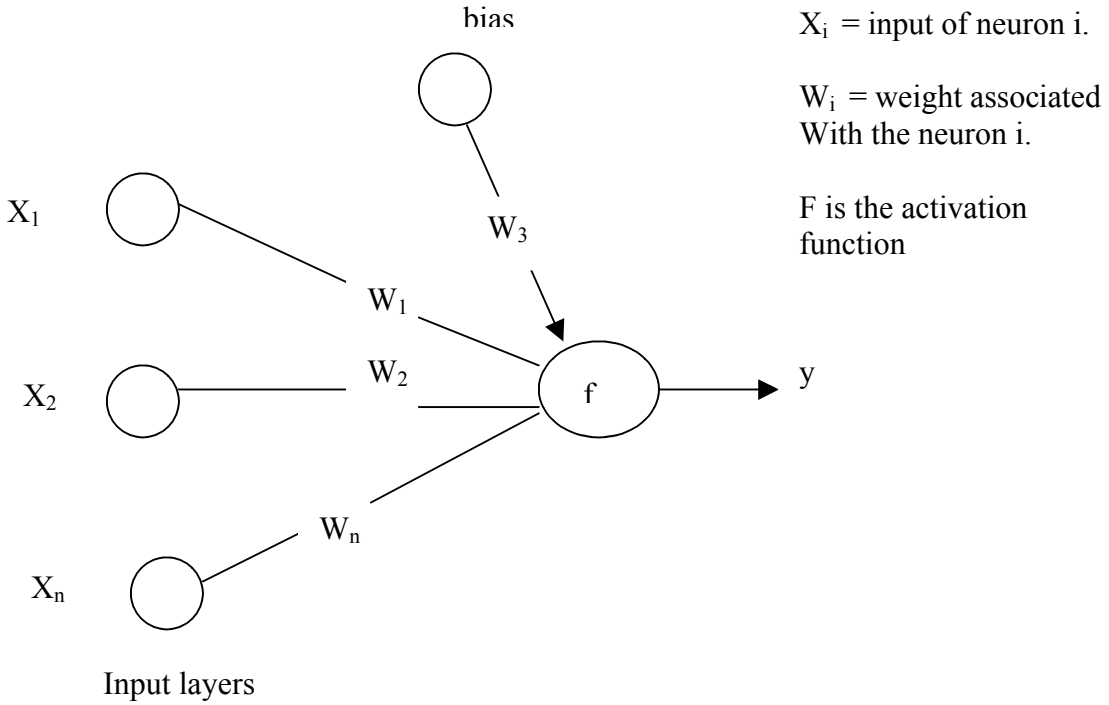

Figure 3.7: Single-layer Feedforward Architecture

3.5.2 Multi-layer Perceptron

This class of networks consists of multiple layers of computational units, usually interconnected in a feed-forward way. Each neuron in one layer has directed connections to the neurons of the subsequent layer. In many applications the units of these networks apply a sigmoid function as an activation function. Multi-layer networks use a variety of learning techniques, the most popular being Back propagation [13]. Here, the output values are compared with the correct answer to compute the value of some predefined error-function. By various techniques, the error is then fed back through the network. Using this information, the algorithm adjusts the weights of each connection in order to reduce the value of the error function by some small amount. After repeating this process for a sufficiently large number of training cycles, the network will usually converge to some state where the error of the calculations is small. In this case, one would say that the network has learned a certain target function.

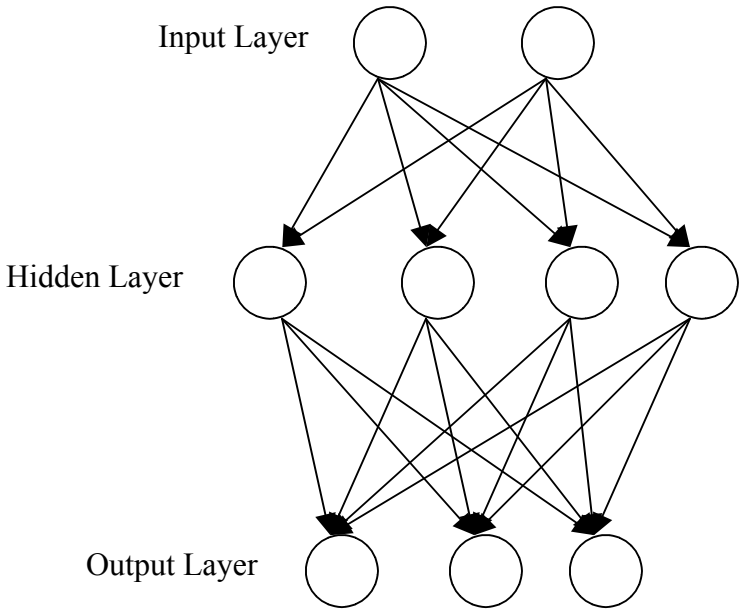

Figure 3.8: Multi-layer Feedforward Architecture

3.6 Neural Network-based Intrusion Detection Systems

The NNID anomaly intrusion detection system is based on identifying a legitimate user based on the distribution of commands she or he executes. This is justifiable because

different users tend to exhibit different behavior, depending on their needs of the system. Some use the system to send and receive e-mail only, and do not require services such as programming and compilation. Some engage in all kinds of activities including editing, programming, e-mail, Web browsing, and so on. However, even two users that do the same thing may not use the same application program. For example, some may prefer the "vi" editor to "emacs", favor "pine" over "elm" as their mail utility program, or use "gcc" more often than "cc" to compile C programs. Also, the frequency with which a command is used varies from user to user.

The set of commands used and their frequency, therefore, constitutes a 'print' of the user, reflecting the task performed and the choice of application programs, and it should be possible to identify the user based on this information. It should be noted that this approach works even if some users have aliases set up as shorthand for long commands they use frequently, because the audit log records the actual commands executed by the system. Users' privacy is not violated, since the arguments to a command do not need to be recorded. That is, we may know that a user sends e-mail five times a day, but we do not need to know to whom the mail is addressed. Building NNID for a particular computer system consists of the following three phases:

- Collecting training data: Obtain the audit logs for each user for a period of several days. We collected DARPA intrusion data.
- Training: Train the neural network to identify the user based on these command distribution vectors.
- Performance: Let the network identify the user for each new command distribution vector. If the network's suggestion is different from the actual user, or if the network does not have a clear suggestion, signal an anomaly.

The first advantage in the utilization of a neural network in the detection of instances of misuse would be the flexibility that the network would provide. A neural network would be capable of analyzing the data from the network, even if the data is incomplete or distorted.

Similarly, the network would possess the ability to conduct an analysis with data in a non-linear fashion. Both of these characteristics are important in a networked environment where the information, which is received, is subject to the random failings of the system [15]. Further, because some attacks may be conducted against the network in a coordinated assault by multiple attackers, the ability to process data from a number of sources in a non-linear fashion is especially important.

The inherent speed of neural networks is another benefit of this approach. Because the protection of computing resources requires the timely identification of attacks, the processing speed of the neural network could enable intrusion responses to be conducted before irreparable damage occurs to the system. Because the output of a neural network is expressed in the form of a probability the neural network provides a predictive capability to the detection of instances of misuse.

A neural network might be trained to recognize known suspicious events with a high degree of accuracy. While this would be a very valuable ability, since attackers often emulate the "successes" of others, the network would also gain the ability to apply this knowledge to identify instances of attacks, which did not match the exact characteristics of previous intrusions. The probability of an attack against the system may be estimated and a potential threat flagged whenever the probability exceeds a specified threshold.

3.7 Application of Neural Networks in Intrusion Detection

While there is an increasing need for a system capable of accurately identifying instances of misuse on a network there is currently no applied alternative to rule-based intrusion detection systems [16]. This method has been demonstrated to be relatively effective if the exact characteristics of the attack are known.

However, network intrusions are constantly changing because of individual approaches taken by the attackers and regular changes in the software and hardware of the targeted systems. Because of the infinite variety of attacks and attackers even a dedicated effort to constantly update the rule base of an expert system can never hope to accurately identify the variety of intrusions.

The constantly changing nature of network attacks requires a flexible defensive system that is capable of analyzing the enormous amount of network traffic in a manner, which is

less structured than rule-based systems. A neural network-based misuse detection system could potentially address many of the problems that are found in rule-based systems.

3.8 Earlier research on Intrusion Detection System

10 years ago Internet was not a utility in any sense. It is now supporting 10 – 15 percent of the Gross Domestic Product of the industrialized world. This makes the needs for better security, and the threats are higher than ever.

Gil Raanan from Sanctum explained why businesses are so in need for better security [30]: "Companies are using their networks for confidential and mission critical functions. In addition, businesses today are more vulnerable than in the past because they are sharing information, such as financial and sales data, over internal and external networks". William A. Wulf, president of 8 National Academy of Engineering, explained how important Internet is today [26]: "We are so dependent on the cyber infrastructure now. We can't do financial transactions without it. Even the larger infrastructure – the power grid, gas pipelines – depends on it".

But just implementation of security components is not enough. Gene Spafford, Director of CERIAS at Purdue University said: "Security comes from understanding systems, goals and methods. Strong tools applied in the wrong way for the wrong reasons don't help, and may even confound other defenses". The person in charge of the security should not just buy some security components, implement these, and think that most of the work is done. An analysis of the system and its need should be done first to find out what the security components should protect, and how important these data are. But the most important work is done after implementation. This is to stay up to date on attack methods, and trim the security components when needed.

There are a few groups that have attempted and successfully used neural networks for intrusion detection, each promoting a different approach. Feature Selection for Intrusion Detection using Neural Networks and Support Vector Machines by Srinivas Mukkamala1

& Andrew H. Sung [17]. This paper concerns using CI-type learning machines for intrusion detection, which is an SVM-based IDS for class-specific detection.

A Neural Network Based System for Intrusion Detection and Classification of Attacks by Mehdi Moradi and Mohammad Zulkernine [18]. This paper presents a neural network approach to intrusion detection. A Multi Layer Perceptron (MLP) is used for intrusion detection based on an off-line analysis approach. The results show that the designed system is 87% accuracy with one hidden layer of neurons in the neural network.

There has also been research on other using soft computing techniques in intrusion detection. S. B. Cho showed in his report that the use of hidden Markov models and attempts to detect intrusions by noting significant deviations from the model can be used with success in anomaly Intrusion Detection Systems [20]. In this experiment he used systems call, process and file access as parameters for the intrusion detection. Experiments with the use of Self-Organizing Maps in intrusion detection have also been done. The 25 parameters that were used in this experiment were username, host, type of connection and time session started.

A Neural Network Based Intelligent Intrusion Detection System by Robert Birkely [11]. In this work he proposed and investigated a neural network based intelligent Intrusion Detection System that can promptly detect attacks, either they are known or never seen before. He got a classification rate of 86% on known and unknown attacks.

In 2004 Moazzam Hosssain worked on intrusion detection with neural networks in his work he used backpropagation algorithm to implement his work [15].

"A Data Mining based Adaptive Intrusion Detection Model (DMAIDM) is presented by Tian-Qing Zhu. The DMAIDM applies a fast heuristic clustering algorithm for mixed data (FHCAM) to distinguish intrusions from legal behaviors efficiently and an attribute-constrained based fuzzy mining algorithm (ACFMA) to construct intrusion Pattern-database automatically [20]**.**

Jake Ryan “Intrusion Detection with Neural Networks”. In his work A backpropagation neural network called NNID (Neural Network Intrusion Detector) was trained in the identification task and tested experimentally on a system of 10 users. The system was 96% accurate in detecting unusual activity, with 7% false alarm rate.

CHAPTER 4: Research Methodology

INTRODUCTION

This chapter presents in detail the research methodology, the importance of neural network to tackle this problem is discussed and overview of how the problem is solved, it also provides sample sessions of normal traffic and what parts of data sessions are used to build the system.

4.1 Overview

The work Objective is to design a behavior based system to detect intruders in a computer network. Operation of the system is divided into four phases:

- Input Data Collection and Preprocessing
- Training
- Detection.
- Testing performance of different ANN training algorithm for IDS.

In preprocessing phase, network traffic is collected and processed for use as input to the system. The data collected is converted into binary so that the input is given to the neural network. In the second phase, this system gathers knowledge about the normal behavior of the network users from the preprocessed input data, and store the acquire knowledge.
In detection phase, the system detects attacks based on the knowledge, which is achieved during the training phase. The main task is to generalize and classify user behavior and detect intruders from this classification. The backpropagation neural network is used to accomplish this task. Backpropagation is a supervised learning method i.e. training the neural net with help of teacher. In this work the training and testing phases has been carried out and also compared the train functions for the IDS.

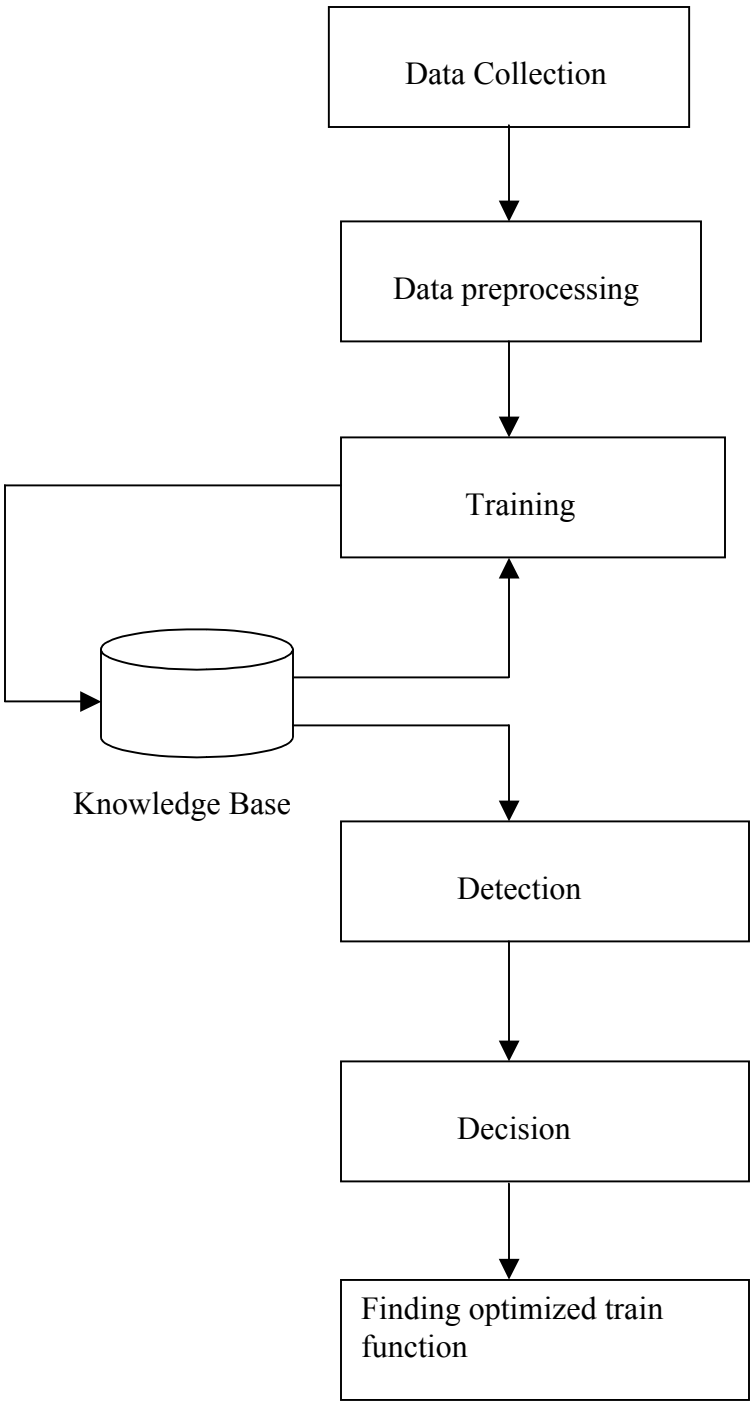

Figure 4.1: Block diagram of system overview.

4.2 Why use neural networks?

Neural networks, with their remarkable ability to derive meaning from complicated or imprecise data, can be used to extract patterns and detect trends, which are too complex to be noticed by either humans or other computer techniques. A trained neural network can be thought of as an "expert" in the category of information it has been given to analyze. This expert can then be used to test with the new situations. Other advantages include:

- Adaptive learning: An ability to learn how to do tasks based on the data given for training or initial experience.
- Self-Organization: An ANN can create its own organization or representation of the information it receives during learning time.
- Real Time Operation: ANN computations may be carried out in parallel, and special hardware devices are being designed and manufactured which take advantage of this capability.
- Fault Tolerance via Redundant Information Coding: Partial destruction of a network leads to the corresponding degradation of performance. However, some network capabilities may be retained even with major network damage.

4.3 Input files to the neural network

Network traffic data called DARPA intrusion detection evaluation, collected, has some sessions. For the training of the neural network 220 sessions of traffic is used. Out of these there were 112 sessions with normal traffic and 108 sessions with attacks. When the learning is over, the system is validated with the network traffic that contains attacks and normal data. 100 sessions of traffic to test the trained network is used, Out of these 50 sessions with normal traffic and 50 sessions with attacks.

Each session from the network traffic is used as an input pattern of the system. To suit the input format of the backpropagation neural network, converted these traffic patterns into binary form. Fig 4.2 shows a sample representation of 15 network sessions.

1 06/19/1998 10:31:22 00:00:01 ftp-data 20 15183 197.218.177.069 172.016.112.207 0 -
2 06/19/1998 10:31:22 00:00:01 domain/u 1106 53 192.168.001.010 172.016.112.020 0 -
3 06/19/1998 11:03:42 00:00:01 smtp 23122 25 194.027.251.021 172.016.114.168 0 -
4 06/19/1998 11:03:42 00:00:01 domain/u 1402 53 192.168.001.010 172.016.112.020 0 -
5 06/19/1998 11:22:57 00:00:12 smtp 21937 25 172.016.112.149 195.115.218.108 0 -
6 06/19/1998 11:22:57 00:00:01 domain/u 1873 53 192.168.001.010 172.016.112.020 0 -
7 06/19/1998 15:35:34 00:00:01 http 19974 80 196.037.075.158 172.016.114.050 0 -
8 06/19/1998 15:35:34 00:00:01 ftp-data 20 19975 172.016.114.148 135.008.060.182 0 -
9 06/19/1998 15:49:05 00:03:00 telnet 21004 23 172.016.112.207 194.027.251.021 0 -
10 06/19/1998 15:49:05 00:00:01 domain/u 53 53 192.168.001.010 192.168.001.020 0 -
11 06/19/1998 16:04:58 00:00:04 smtp 23705 25 172.016.113.084 135.008.060.182 0 -
12 06/19/1998 16:04:58 00:00:01 http 23706 80 172.016.117.132 131.084.001.031 0 –
13 06/19/1998 10:29:00 00:00:01 smtp 20150 25 195.073.151.050 172.016.113.084 0 -
14 06/19/1998 10:29:00 00:00:01 smtp 20512 25 195.073.151.050 172.016.113.105 0 -
15 06/19/1998 10:31:22 00:00:01 domain/u 1095 53 192.168.001.010 172.016.112.020 0 -

Figure 4.2: Sample sessions of network traffic

Session Number: Session serial number is irrelevant, as it has nothing to do with user's behavior. So, it can be avoided.

Start Date: As for every single day in a year, there is a unique start date. So, it can be avoided.

Session Server Name: Session server name is also represented by the destination port number. So, it is redundant.

Start time: Start time shows the time when particular session starts. User's habit of using specific types of protocol in some specific time can be determined from this piece of information.

Time Duration: It shows the total time duration of the corresponding session.

Source and Destination Ports: The source and destination port numbers range from decimal 0 to 65535, which can be represented by 16 binary digits.

Source and Destination IP Addresses: The size of IP address in ipv4 is 32 binary bits. These addresses are normally represented as four parts, where each part is represented by 3 decimal digits. It means each part of an IP addresses needs 8 binary bits to be converted into binary form. Here, IP addresses are represented in decimal form. So, they have to be converted back into binary form, which consists of 32 binary bits.

Attack Bit: The state of attack can be either 'yes' or 'no'. It means, an attack can simply be represented by a '1' and normal traffic can be represented by a '0', hence, it requires only one binary digit.

4.4 Training

As stated in previous chapter backpropagation neural network is used for behavior classification. MATLAB Neural Network Toolbox was used for the implementation of the backpropagation neural networks. Using this tool one can define specifications like number of layers, number of neurons in each layer, activation functions of neurons in different layers, and number of training epochs. Then the training feature vectors and the corresponding desired outputs can be fed to the neural network to begin training. The neural network that is going to construct for this work will be of 121 units in the input layer and one unit in the output layer. The network consists of only one hidden layer and can choose the number of hidden nodes in the hidden layer. In this work the network has been trained and tested by changing the training functions and hidden units.

4.5 Criteria for training termination

When using a backpropagation neural network, the usual criteria for termination of the training is that the RMS error is reduced to an acceptable level. There is no standard for the RMS error, but usually the lower it is, the better the classification rate is. But a too low RMS error could also over train the neural network causing the network to detect things that are exactly identical to the training data. In this work 0.0001RMS error is considered as performance goal.

4.6 Architecture of the network

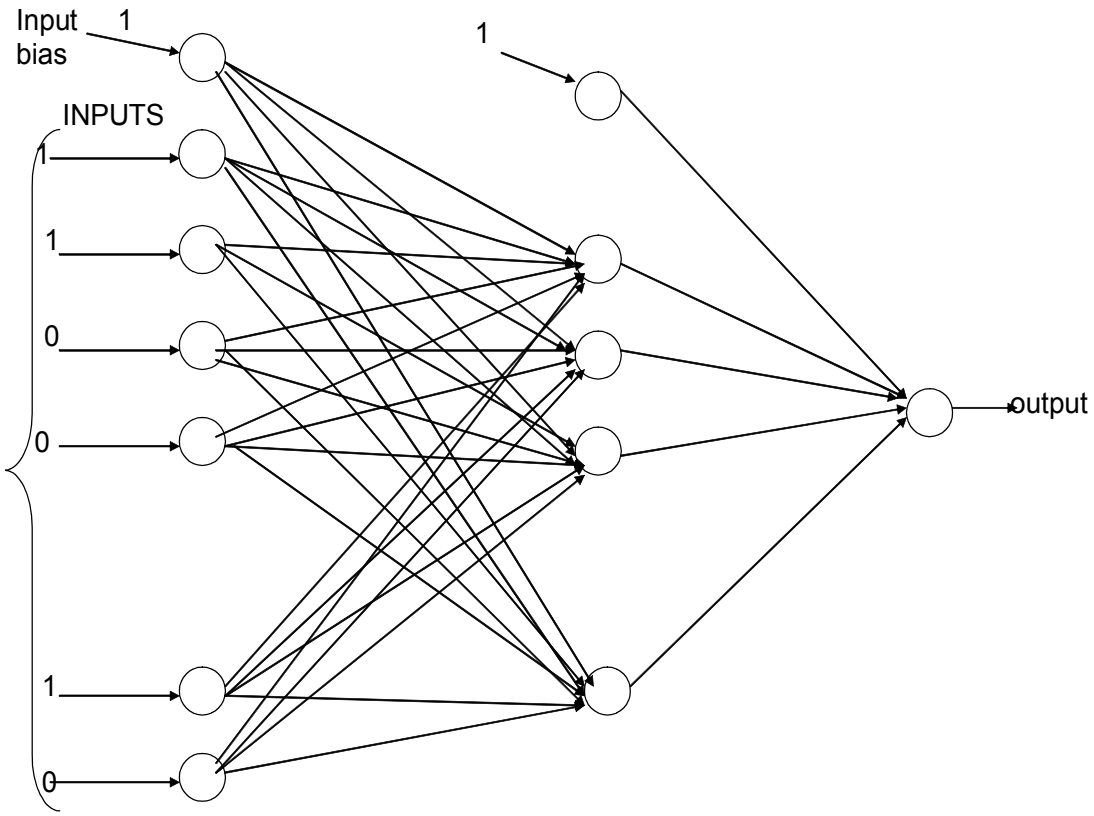

Figure 4.2: Neural Network Architecture

As shown in above figure the input vector of the network is 122 units. Among these 122 units, first unit is the bias, which is always '1'. The other 121 units are taken from the first 121 units of the session vector and the 122^{th} i.e. last unit of session vector is used as the target output. There are many variations available for the backpropagation network. The simplest implementation of backpropagation learning updates the network weights and biases in the direction in which the performance function decreases most rapidly.

4.7 Parameters for the Neural Network

A backpropagation neural network was used in work. The parameter setting for the training of the neural network is of great importance. Number of inputs tells how many units are there in the input vector. Number of outputs tells how many bits there are in the second part of the training pair. Number of training pairs tells how many sessions are used in the training of the neural network. In this work 220 training pairs were used, of these, 112 sessions are with normal traffic and 108 sessions are with attacks. RMS-error is a value that can be adjusted, so that the neural network trained as needed. But a too low RMS error could also over train the neural network.

Parameters	Value
Number of inputs	121
Number of outputs	1
Number of hidden units	10
Number of Training Pairs	220
RMS-error	0.0001

Table 4.1: Used parameters for Neural Network

4.8 Detection

In this phase, the system is already trained with the normal behavior of user and from that knowledge it will detect the abnormal behavior of authorized users. In the training phase the input pattern and output pattern is given to the network but in this phase only the input pattern is given. This is because, the network itself predicts the possible output pattern for each given input pattern from the acquired knowledge through training. This output will show us whether the given input pattern represents normal behavior or an attack.

CHAPTER 5: Implementation

INTRODUCTION

This chapter provides details about the training algorithm that has been used. The architecture of the neural network, converting the data sessions into binary format and the parameters that has been used to build the system are also included in this chapter.

5.1 Parameters used

Start Time - 7 bits Duration - 18 bits Src. Port - 16 bits Dst. Port - 16 bits	**Src. IP Address - 32 bits** **Dst. IP Address - 32 bits** **Attack/Normal - 1 bit**

Figure 5.1: Parameters used in this experiment

5.2 Input data processing

For converting the sessions into binary form MS-Excel is used in this work. As conversion of a decimal number using excel is possible for which the binary output consists of only 12 bits, but for source and destination port the maximum value will be of 65535. It is not possible to convert this huge number into binary through Excel. So, Morefunc tool is used for this conversion. In this tool CHBASE function is used, which converts a value from one base to the other base.

Screenshot of Morefunc:

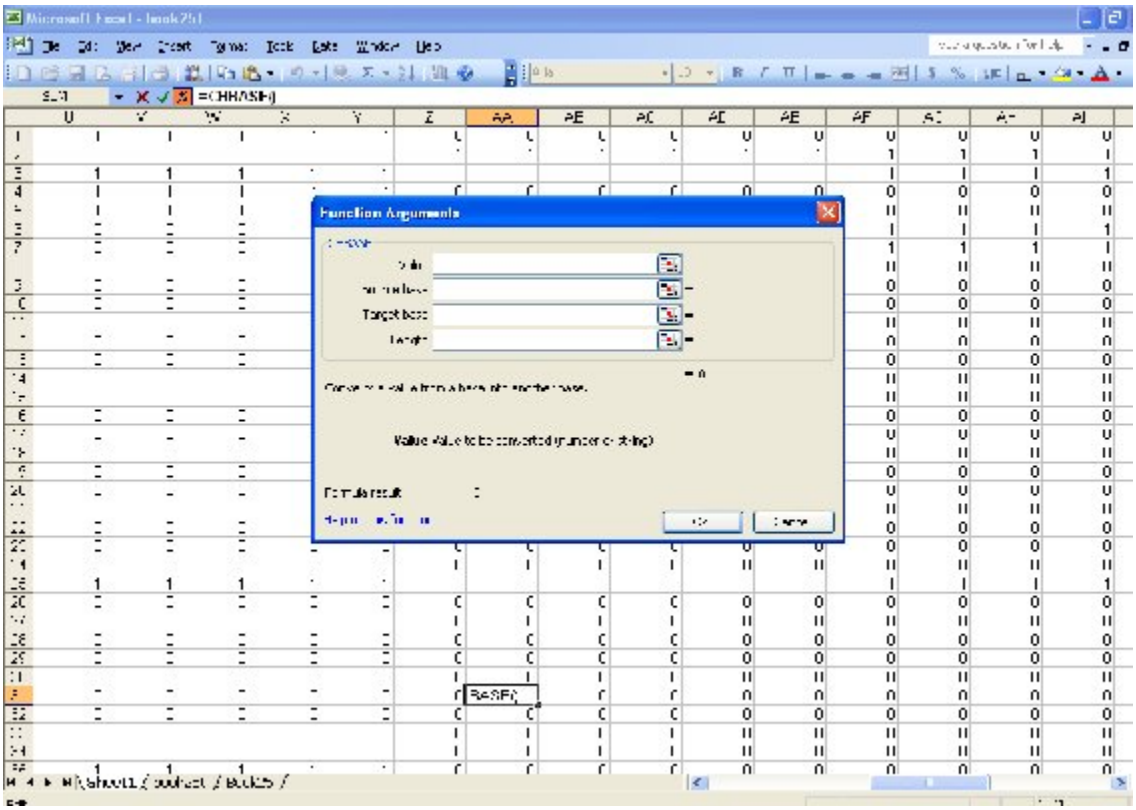

Figure 5.2: Screenshot of Morefunction

Start time: In this work it has been considered that each time slot is of 15 minutes duration. Each slot consists of fifteen minutes, it means there are four slots in an hour and hence, instead of representing the specific start time of the session, it has been represented in the binary representation of the time slot number where it belongs to. For an example 10:31:22 the decimal number equivalent of this is 10*4+3 = 42, added 2 because 30-45 will belong to Slot 3, ignored seconds. Maximum value that can get from this is 96, seven binary bits are enough to represent.

Time Duration: It represents the time duration of the corresponding session. In case of hours, the maximum number is 23 and can be represented by 6 bits and in case of minutes as well as seconds, the maximum number is 59 which can also be represented by 6 bits. It

makes total of 18 bits to represent time duration. 00:00:01 can be represented as 000000000000000001 in binary format.

Source and Destination ports: As we know that the port number ranges from 0 to 65535 which can be represented by 16 bits. 15183 can be represented as 0011101101001111.

Source and Destination addresses: Address can be represented by 32 bits. 197.218.177.069 can be represented by 11000101110110101011000101000101.

Attack Bit: Normal traffic can be represented by a '0' and for attack '1'. Hence it requires only one bit to represent this.

5.3 Training

The training process requires a set of examples of proper network behavior - network inputs p and target outputs t. In previous chapters it has been already mentioned that the backpropagation neural network is used for behavior classification. In this work neural network is constructed with 121 units in the input layer and one unit in the output layer. Only one hidden layer is used in this work and there is option to choose different number of hidden nodes for the system. This can be simply done by changing the value of the variable for hidden units in the implementation. The inputs to the neural network are the input pattern consists of 121 binary bits. The first 7 units in the input layer get input from the time slot session. Similarly, next 18 units get inputs from the time duration of a session and so on. For each pair of input units and hidden units connection exists between them. Similarly, each hidden layer is connected to the output unit as we have only one output unit. Figure 5.3 shows how the input vector connected to the input layer and the output produced by the network. A weight value is associated with each of the connections. The output of the neural network will be 0 if the input pattern is normal and the output of the neural network will be 1 if the input pattern represents an attack.

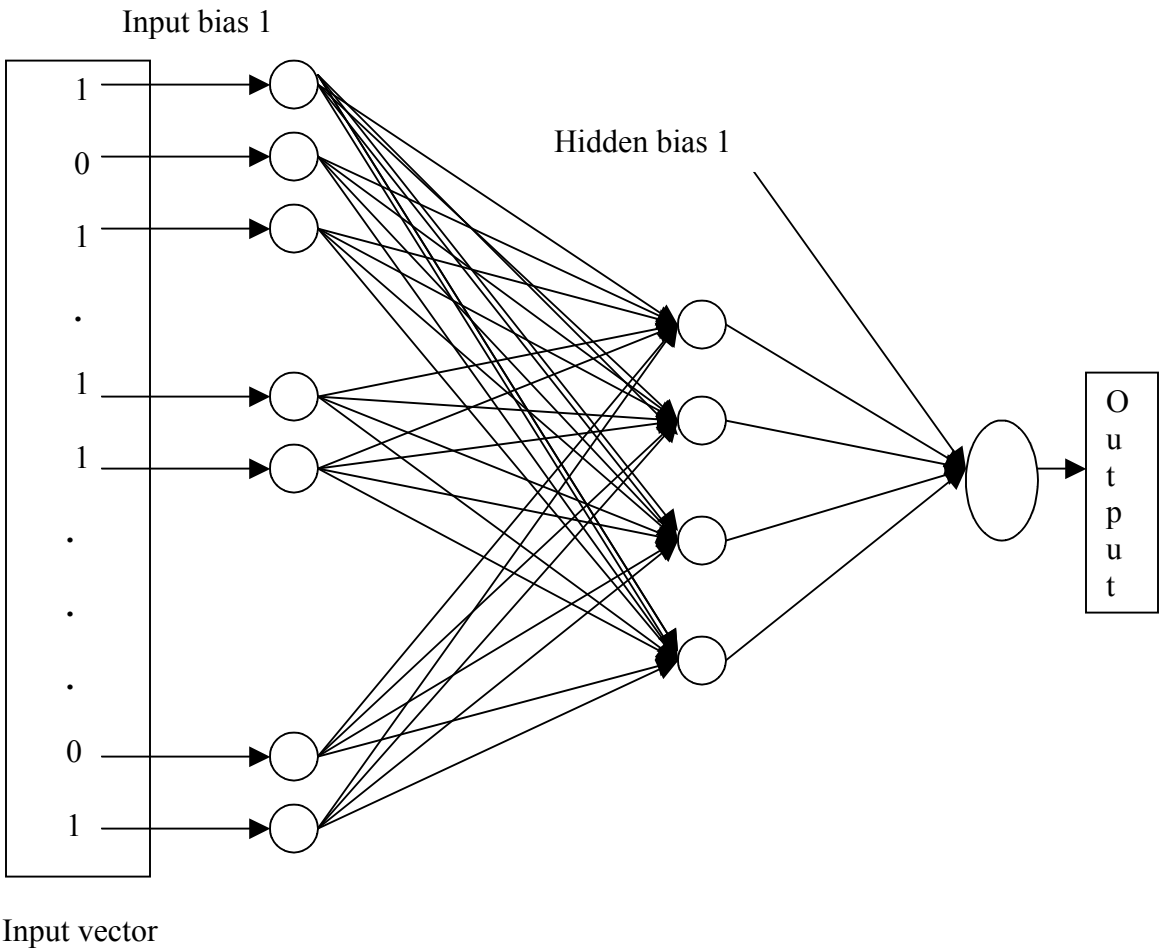

Figure 5.3: Input vector, Bias, Output and Neural net

The training algorithm of backpropagation involves four stages is as follows:

- Initialization of weights
- Feed forward
- Back propagation of errors
- Updation of the weights and biases.

There are two types of learning in back propagation: sequential learning and batch learning. In sequential learning a given input pattern is propagated forward, the error is determined and back propagated, and the weights are updated. In batch learning the weights are updated only after the entire set of training network has been presented to the network. Thus the weights update is only performed after every epoch. To train the network, the proposed training algorithm used in the backpropagation algorithm is given in Figure 5.4. In this work batch learning has been used.

5.3.1 Backpropagation Algorithm

Algorithm:

Initialize the weights in the network (often randomly)

Do

For each example e in the training set

O = neural-net-output (network, e); forward pass

T = teacher output for e

Calculate error (T - O) at the output units

Compute delta_wi for all weights from hidden layer to output layer; backward pass

Compute delta_wi for all weights from input layer to hidden layer; backward pass continued

Update the weights in the network

Until all examples classified correctly or stopping criterion satisfied

Return the network

Figure 5.4: Backpropagation Algorithm

Figure illustrates the flowchart of the error backpropagation training algorithm for a basic two layer network. The learning begins with the feedforward recall phases. After a single pattern vector z is submitted at the input, the layer responses y and o are computed in this phase. Then the error signal computation phase follows. Note that the error signal vector must be determined in the output layer first, and then it is propagated toward the network input nodes. The k*j weights are subsequently adjusted within the matrix w. Finally, j*I weights are adjusted within the matrix v. the final error value for the entire training cycle is calculated after each completed pass through the training set[z_1,z_2 ,z_3------z_p]. The learning procedure stops when the final error value below the upper bound e_{max} is obtained. In the previous chapters it has been mentioned that the performance goal is 0.0001(RMS error).

5.3.2 Backpropagation Flowchart

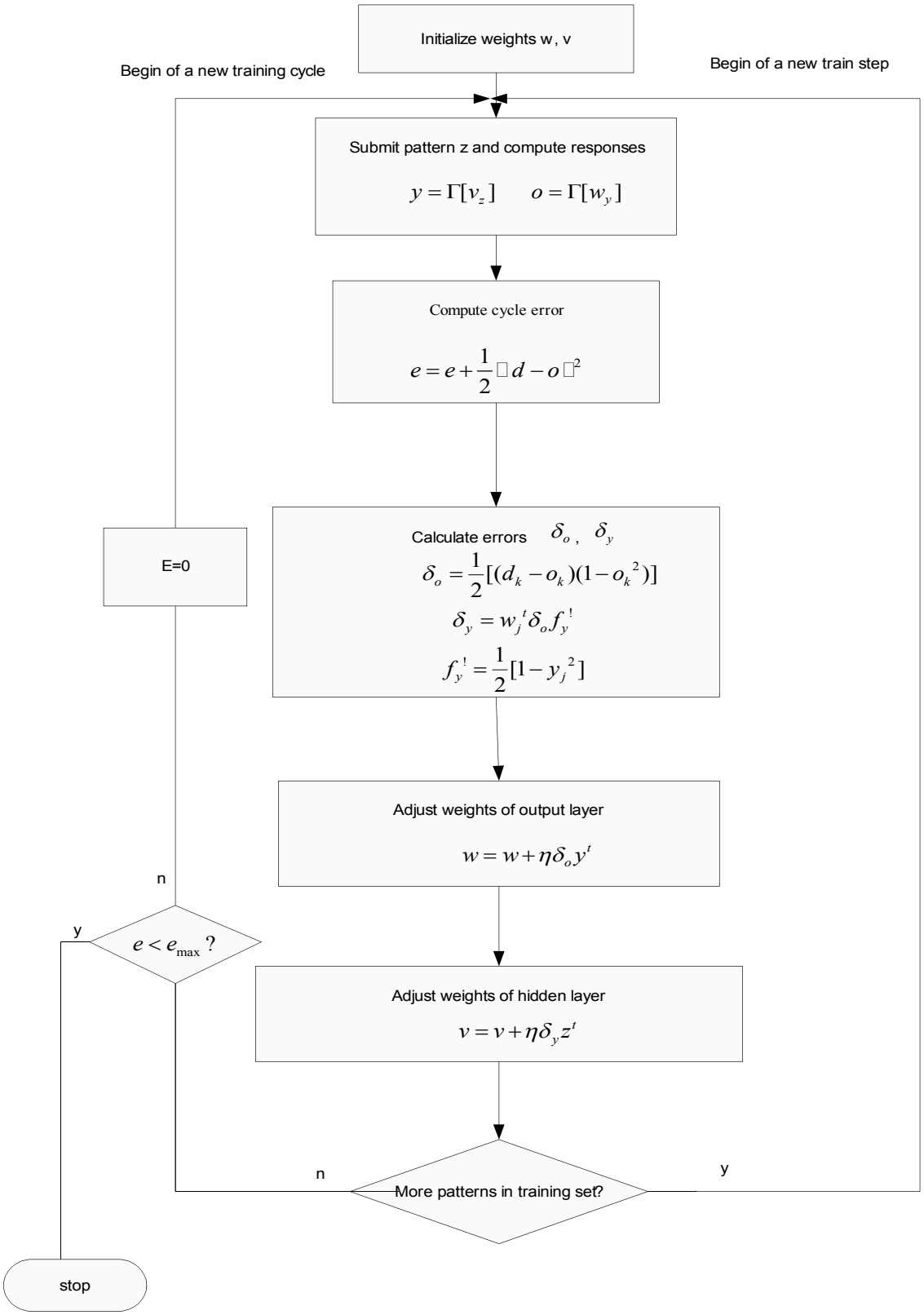

Figure 5.5 Backpropagation Flowchart

5.4 Validation

The system is already trained with the normal behavior of user and from that knowledge it will detect the abnormal behavior of authorized users. In the training phase the input pattern and output pattern is given to the network but in this phase only the input pattern is given. After initialization of the weights, the input units of input layer are activated with the input patterns that have been taken from the input file. The outputs of the input layer are propagated towards the output layer similarly as training. The calculated output from the output unit of output layer is considered as the desired output pattern. This output will show us whether the given input pattern represents normal behavior or an attack.

In the present work, 5 different kinds of files are managed, dataset1 containing training input patterns of normal traffic and known attacks, dataset2 containing training output pattern of normal traffic and known attacks, dataset3 containing test input patterns of normal traffic, datset4 containing test input patterns of known attacks, dataset5 containing test input patterns of unknown attacks.

In this work different tests are performed, the preliminary experiment is performed to find out the minimum number of hidden nodes needed to train the neural network to detect attacks. This test gave the background for choosing the number of hidden units for the training of the neural network for the last two experiments. In the second experiment 4 hidden units were considered and different test runs were made for the training functions shown in table 5.1. The final experiment was with 10 hidden units.

5.5 Training algorithms

Different test runs were made for each of the following training algorithm and also tried with different number of hidden nodes. The different training functions that have been used in this work are given in the table 5.1 [19].

Function	Description
Traingd	Basic gradient descent. Slow response, can be used in incremental mode training.
Traingdm	Gradient descent with momentum. Generally faster than Traingd. Can be used in incremental mode training.
Traingdx	Adaptive learning rate. Faster training than Traingd, but can only be used in batch mode training.
Trainrp	Resilient back propagation. Simple batch mode training algorithm with fast convergence and minimal storage requirements.
Traincgf	Fletcher-Reeves conjugate gradient algorithm. Have smallest storage requirements of the conjugate gradient algorithms.
Traincgp	Polak-Ribiere conjugate gradient algorithm. Slightly larger storage requirements than Traincgf. Faster convergence on some problems.
Traincgb	Powell-Beale conjugate gradient algorithm. Slightly larger storage requirements than Traincgp. Generally faster convergence.
Trainscg	Scaled conjugate gradient algorithm. The only conjugate gradient algorithm that requires no line search. A very good general purpose training algorithm.
Trainbfg	BFGS quasi-Newton method. Requires storage of approximate Hessian matrix and has more computation in each iteration than conjugate gradient algorithms, but usually converges in less iteration.
Trainoss	One step secant method. Compromise between conjugate gradient methods and quasi-Newton methods.
Trainlm	Levenberg-Marquardt algorithm. Fastest training algorithm for networks of moderate size. Has memory reduction feature for use when the training set is large.

Table 5.1 Descriptions of Different Neural Network Training Functions

CHAPTER 6: Evaluation

INTRODUCTION

This chapter highlights on experimental issues. First of all we discuss how to proceed with the training and testing, the files that are used for testing. It also highlights the affect of output when different train function used to train the network.

Experiments

From the preliminary experiment it can be examined that the number of hidden layers that were needed to train the neural network and the different training functions that has been used to train with the same dataset. The different training functions are compared in this work. In these experiments normal, known attacks and unknown attacks are used in different files for testing. The following experiments has been performed to detect intrusions from collected datasets using nntool of the matlab environment

6.1 Experiment 1

As preliminary experiment Trainlm training function is used with 2 hidden units. The network has been trained till the RMS error value is reduced to an acceptable level. For training we used sessions that were collected from the DARPA data set. The results of training normal traffic have been shown in table 6.1. In the table number of hidden units denotes the number of neurons in the first layer. Number of epochs is nothing but number of iterations needed to converge the network with desired accuracy. Performance goal status represents that desired goal is achieved or not. We have been divided test data into two files for testing. One consists of normal and known test data other consists of unknown test data.

The table 6.1 shows the training of data set evaluation. First tried with 2 hidden units it taken 12 epochs to train the network and the classification rate is 97% for the normal traffic and 98% for the attacks. In this work 112 sessions of normal traffic and 108 sessions of attack traffic are used to train the network. By considering this results into

account that the network with 2 hidden units not trained up to extent. It has been observed from the table that by increasing the number of hidden layers the performance of the network has been increased.

Hidden Units	No. of Epochs	Performance goal status	Classification rate for Training		Classification rate (%) for Training	
			Normal	Attack	Normal	Attack
2	12	met	108/112	105/108	97	98
4	14	met	112/112	108/108	100	100
6	17	met	112/112	108/108	100	100
12	15	met	112/112	108/108	100	100

Table 6.1: Preliminary Experiment results

6.2 Experiment 2 (training)

In this work 220 sessions of traffic are used for training of the neural network. Out of these there were 112 sessions with normal traffic and 108 sessions with attacks. The results shown in table 6.2 are obtained for the training of network with 4 hidden units and with different train functions of the matlab environment.

Function	No. of epochs	Classification rate for training set Normal	known Attacks	Classification rate (%) for training set Normal	known Attacks
Traingd	76637	112/112	108/108	100	100
Traingdm	90000	112/112	108/108	100	100
Traingdx	49318	112/112	108/108	100	100
Trainrp	63	112/112	108/108	100	100
Traincgf	84	112/112	108/108	100	100
Traincgp	83	112/112	108/108	100	100
Traincgb	55	112/112	108/108	100	100
Trainscg	83	112/112	108/108	100	100
Trainbfg	31	112/112	108/108	100	100
Trainoss	208	112/112	108/108	100	100
Trainlm	11	112/112	108/108	100	100

Table 6.2: Results of training ANN for Normal and Known attacks

The results shown in table 6.3 are obtained for the training of the network with 10 hidden units and with different train functions of the matlab environment.

Function	No. of epochs	Classification rate for training set Normal	known Attacks	Classification rate (%) for training set Normal	known Attacks
Traingd	13961	112/112	108/108	100	100
Traingdm	14841	112/112	108/108	100	100
Traingdx	64956	112/112	108/108	100	100
Trainrp	109	112/112	108/108	100	100
Traincgf	120	112/112	108/108	100	100
Traincgp	98	112/112	108/108	100	100
Traincgb	91	112/112	108/108	100	100
Trainscg	98	112/112	108/108	100	100
Trainbfg	47	112/112	108/108	100	100
Trainoss	220	112/112	108/108	100	100
Trainlm	4	112/112	108/108	100	100

Table 6.3: Results of training ANN for Normal and Known attacks

6.2.1 Traingd

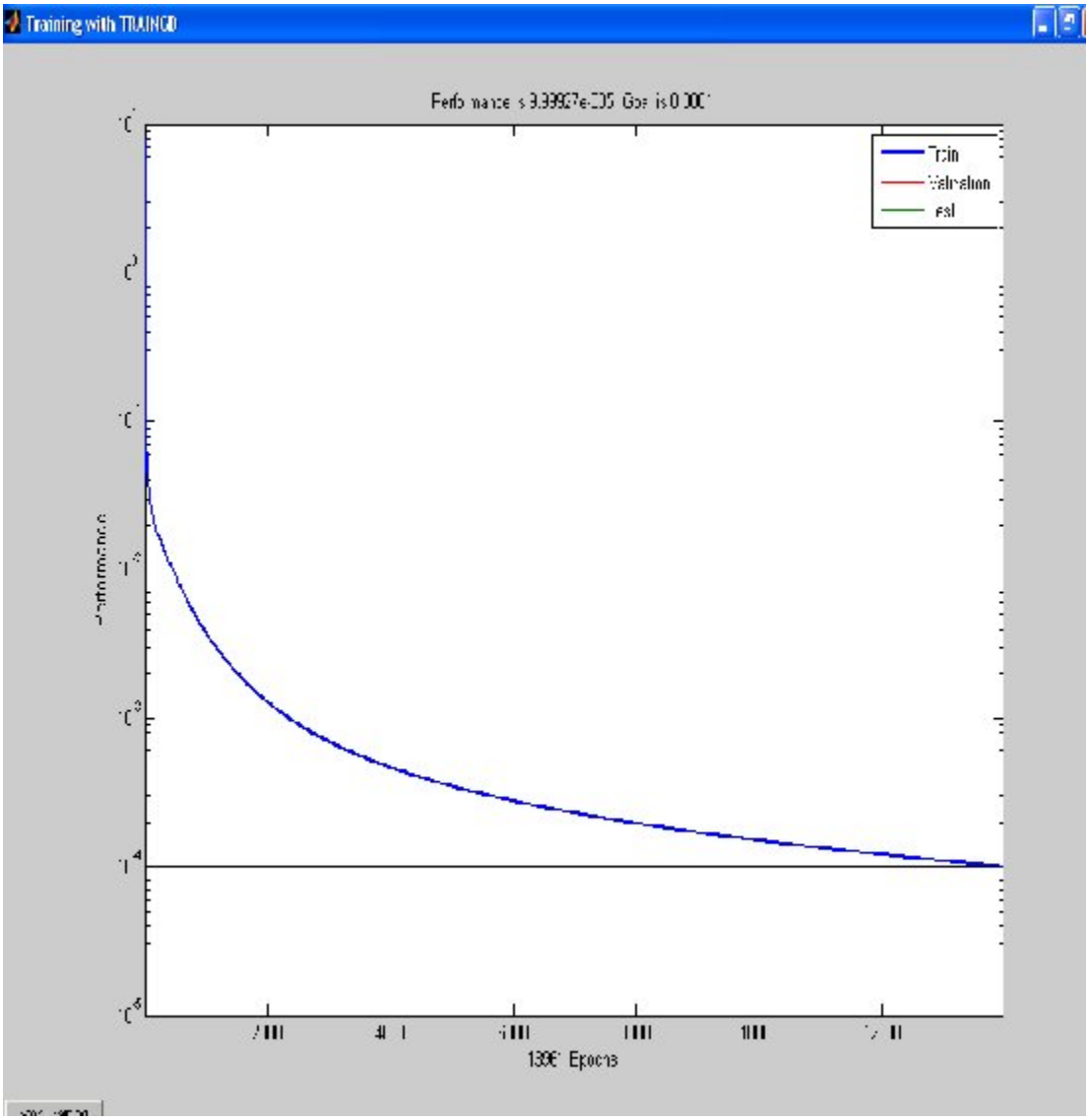

Figure 6.1: Result of training ANN using Traingd

The graph shown in figure 6.1 represents the output of the training of the network and 13961 epochs have been taken to get trained the network using the traingd train function. In this case the performance gal of the network has been achieved.

6.2.2 Trainlm

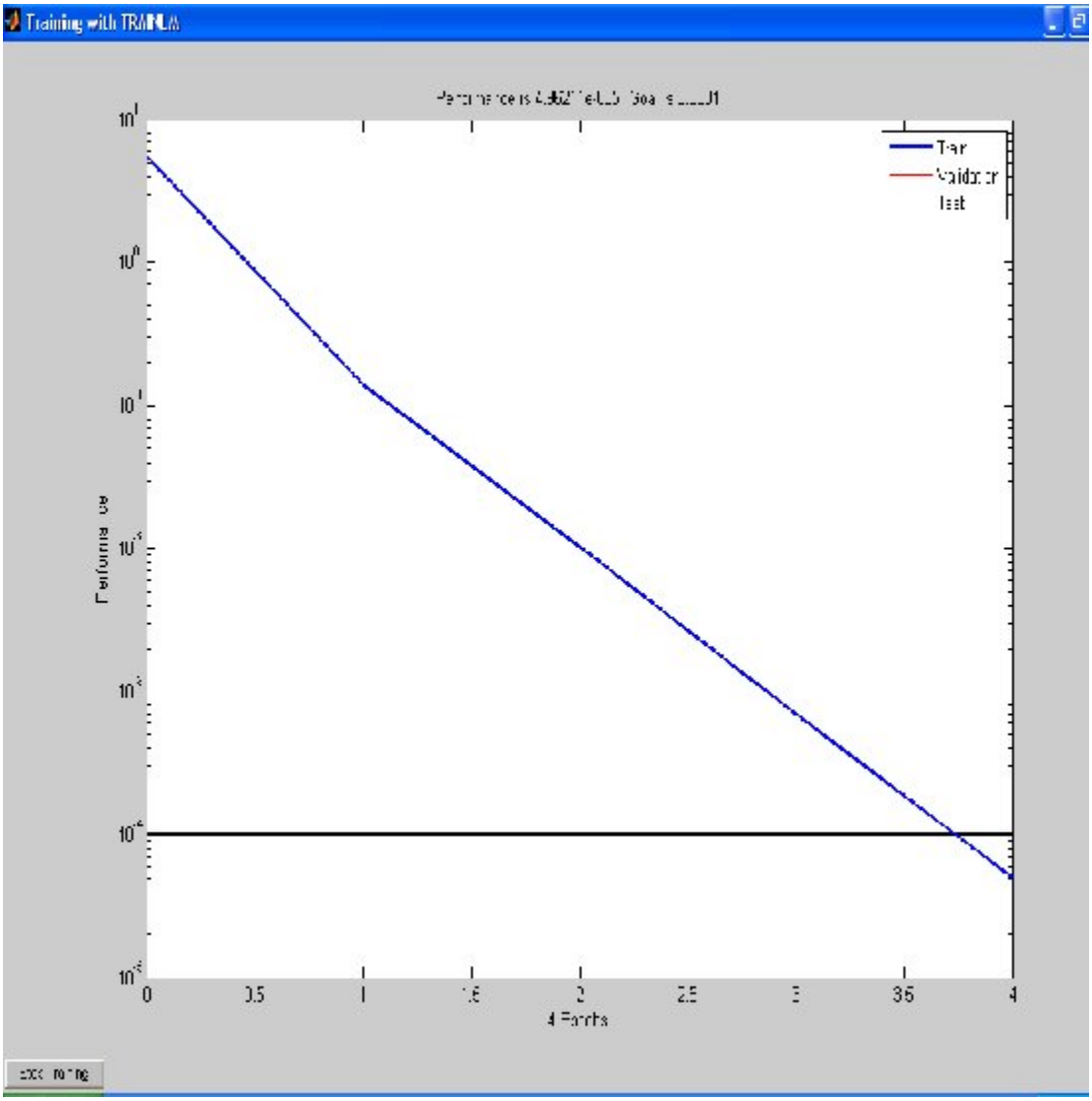

Figure 6.2: Result of training ANN using Trainlm

The graph shown in figure 6.2 represents the output of the training of the network and 4 epochs have been taken to get trained the network using the trainlm train function. In this case the performance gal of the network has been achieved.

6.2.3 Trainbfg

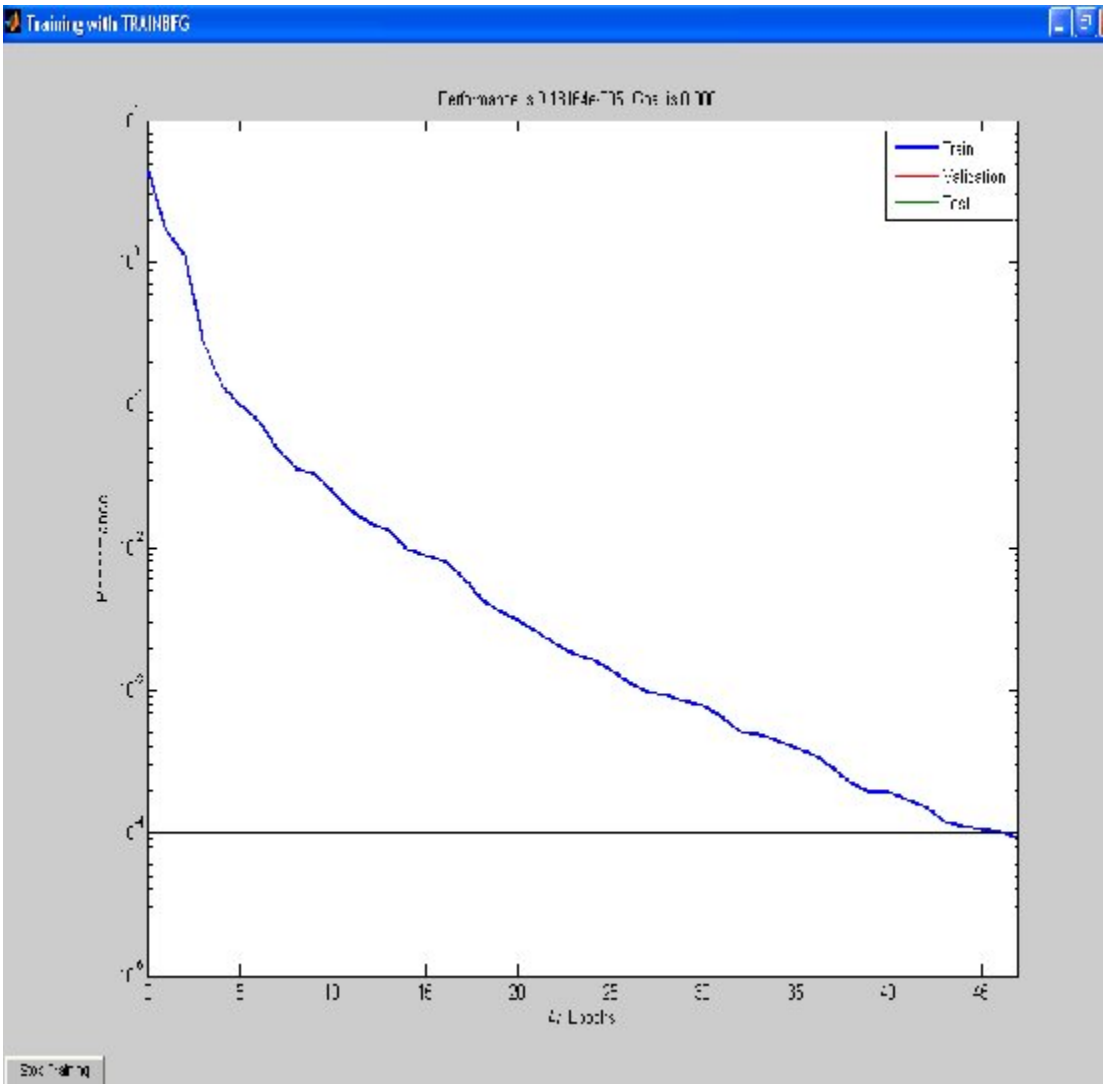

Figure 6.3: Result of training ANN using Trainbfg

The graph shown in figure 6.3 represents the output of the training of the network and 47 epochs have been taken to get trained the network using the trainbfg train function. In this case the performance gal of the network has been achieved.

6.2.4 Trainoss

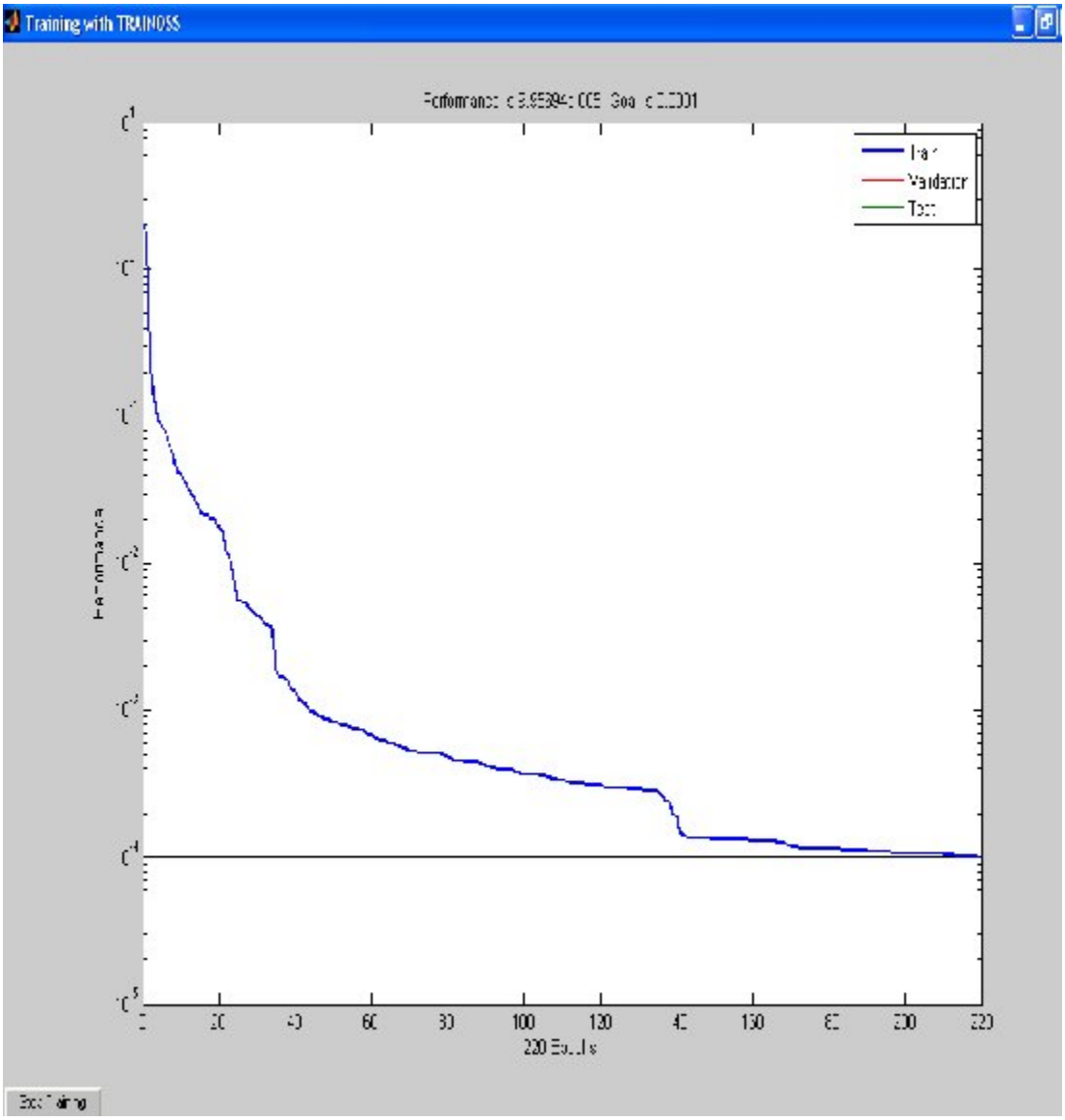

Figure 6.4: Result of training ANN using Trainoss

The graph shown in figure 6.4 represents the output of the training of the network and 220 epochs have been taken to get trained the network using the trainoss train function. In this case the performance gal of the network has been achieved.

6.2.5 Trainscg

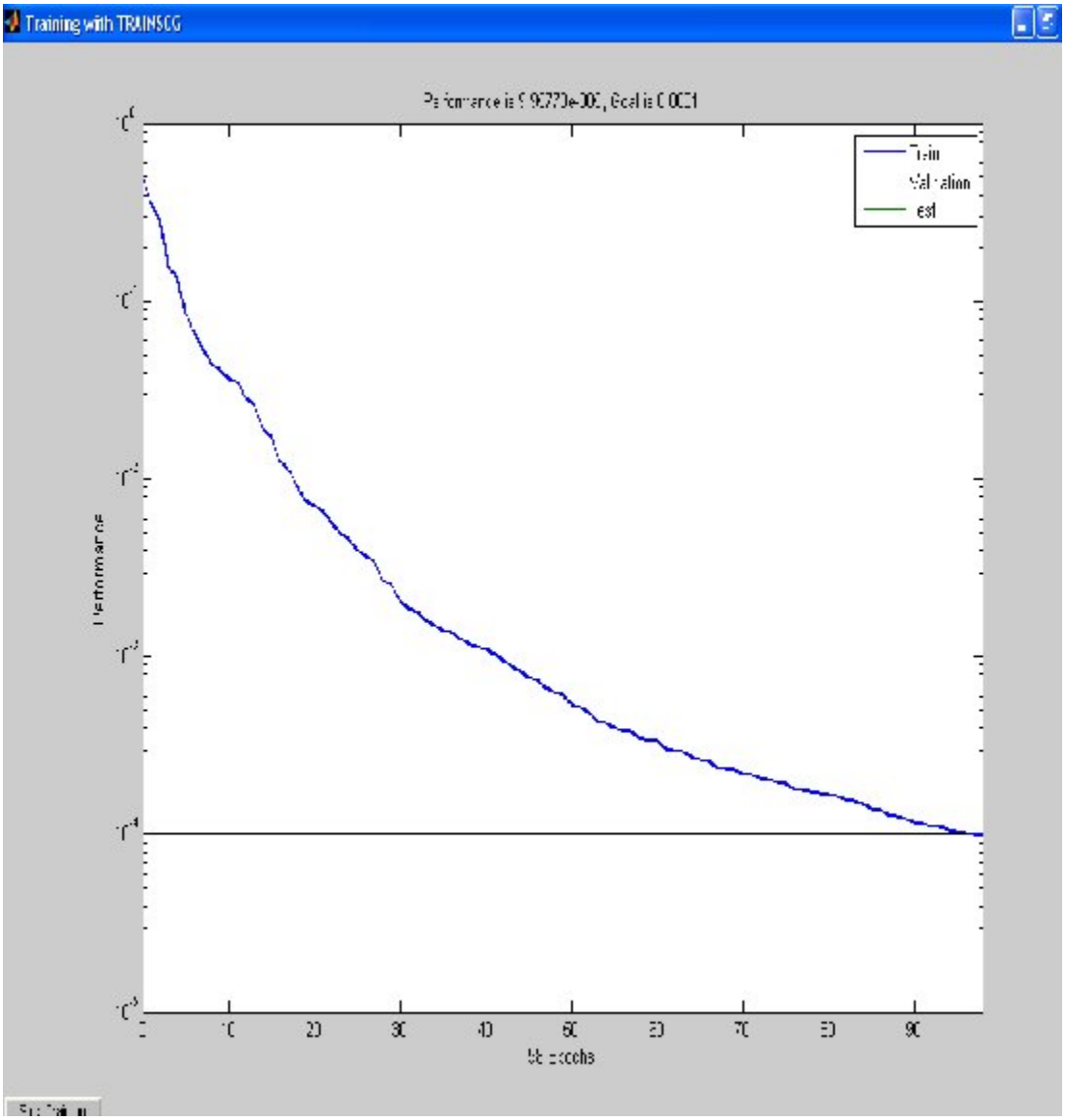

Figure 6.5: Result of training ANN using Trainscg

The graph shown in figure 6.5 represents the output of the training of the network and 98 epochs have been taken to get trained the network using the trainscg train function. In this case the performance gal of the network has been achieved.

6.2.6 Traincgb

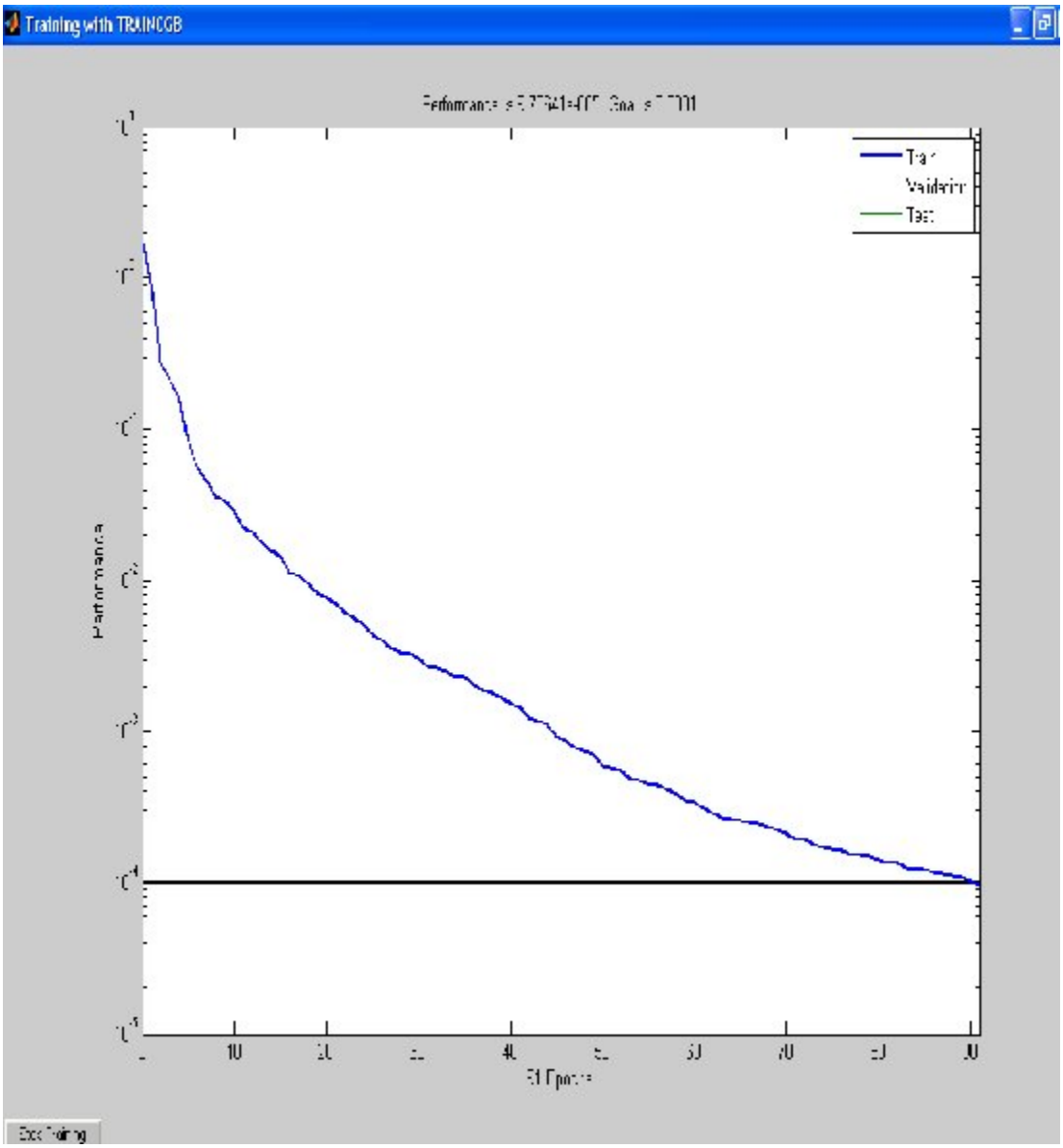

Figure 6.6: Result of training ANN using Traincgb

The graph shown in figure 6.6 represents the output of the training of the network and 91 epochs have been taken to get trained the network using the traincgb train function. In this case the performance gal of the network has been achieved.

6.2.7 Traincgp

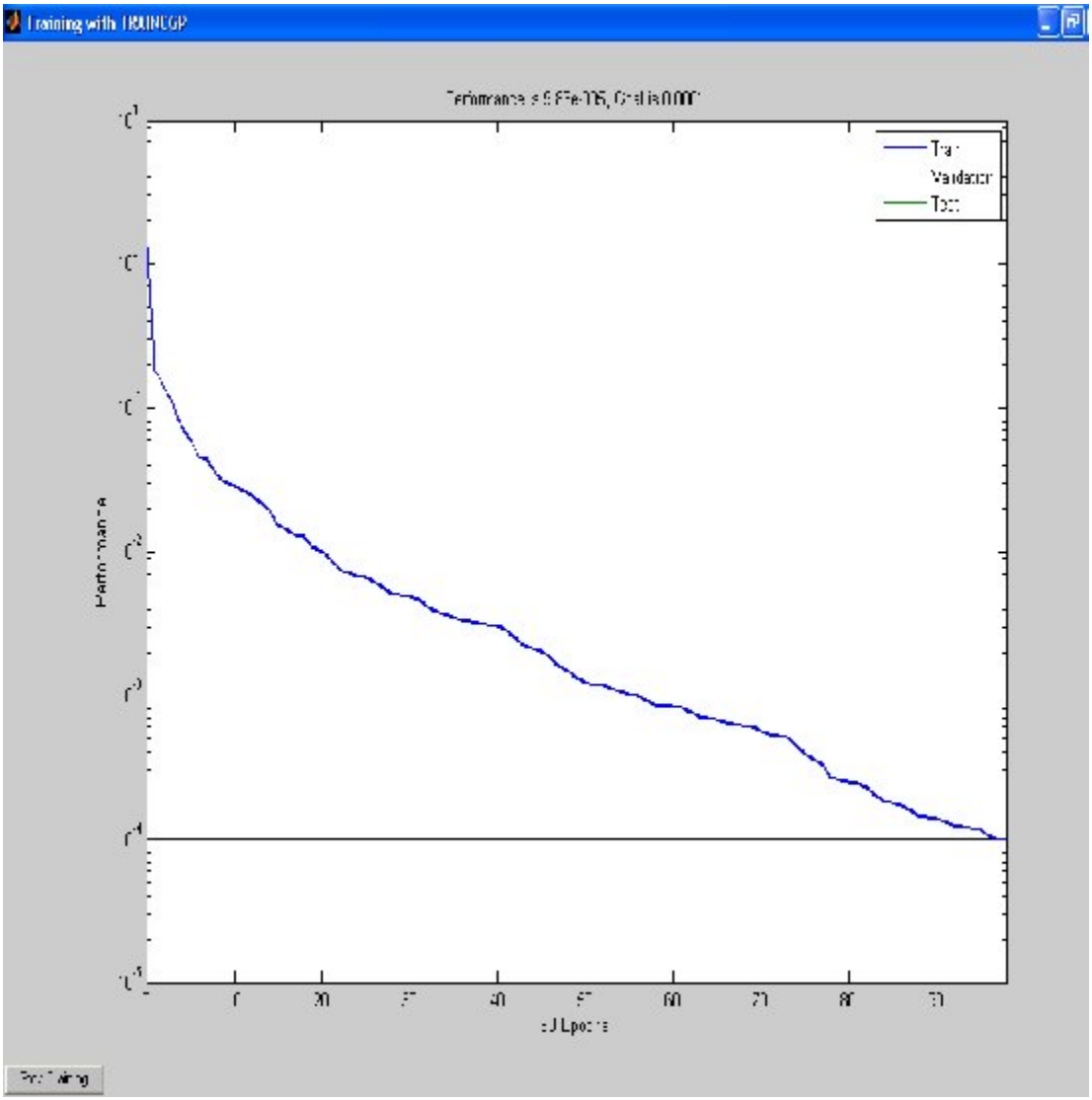

Figure 6.7: Result of training ANN using Traincgp

The graph shown in figure 6.7 represents the output of the training of the network and 98 epochs have been taken to get trained the network using the traincgp train function. In this case the performance gal of the network has been achieved.

6.2.8 Traincgf

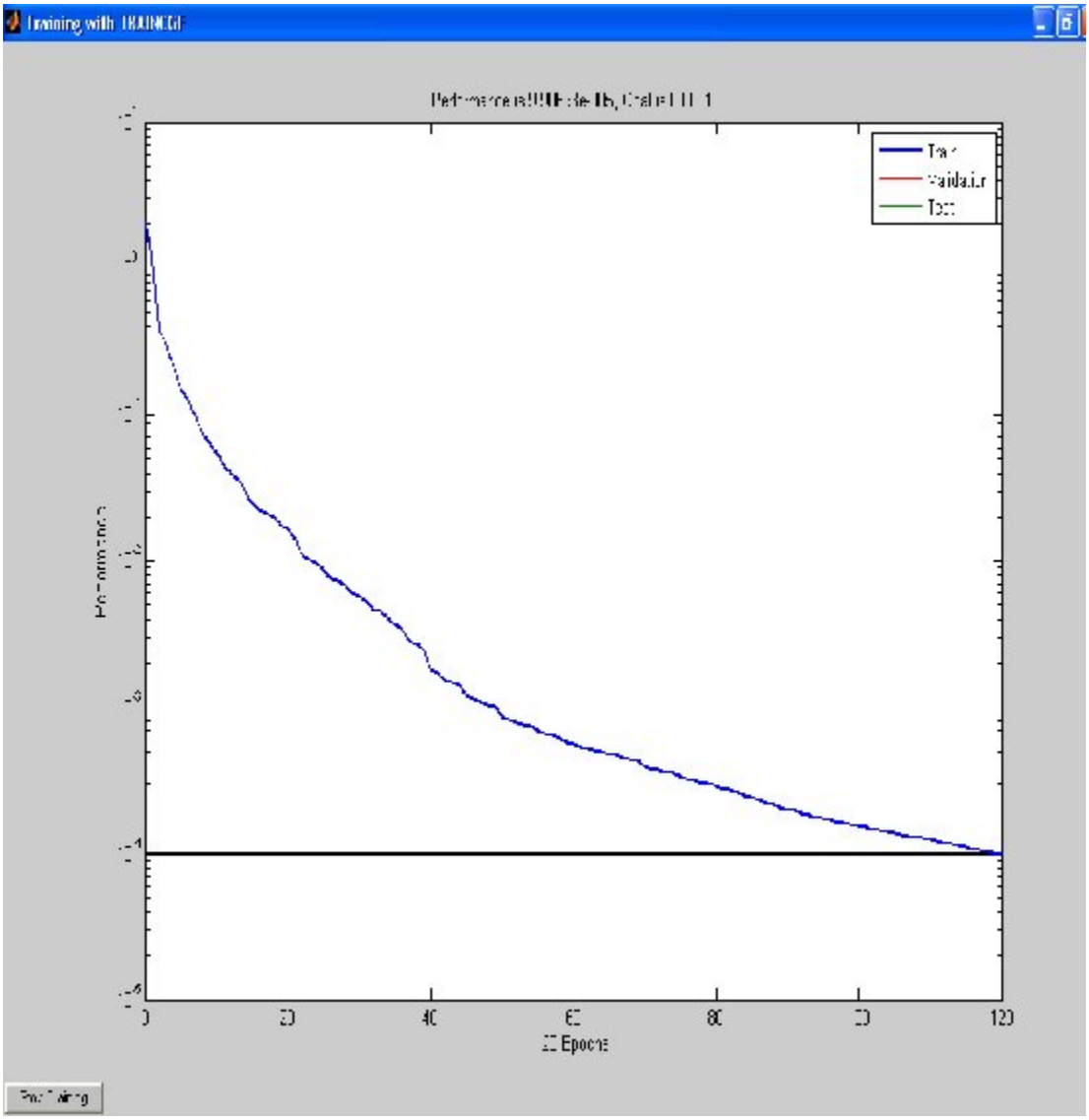

Figure 6.8: Result of training ANN using Traincgf

The graph shown in figure 6.8 represents the output of the training of the network and 120 epochs have been taken to get trained the network using the traincgf train function. In this case the performance gal of the network has been achieved.

6.2.9 Trainrp

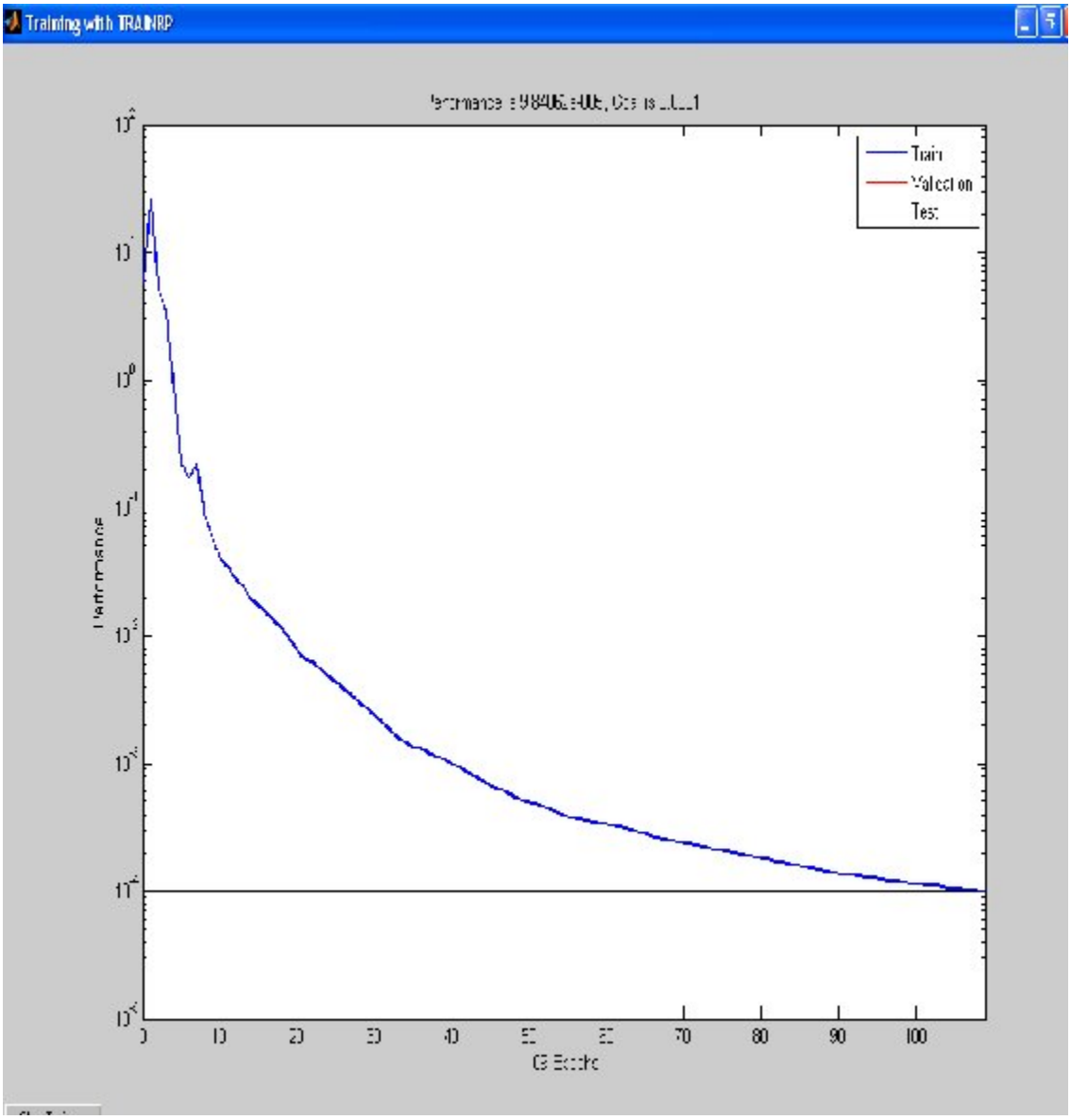

Figure 6.9: Result of training ANN using Trainrp

The graph shown in figure 6.9 represents the output of the training of the network and 109 epochs have been taken to get trained the network using the trainrp train function. In this case the performance gal of the network has been achieved.

6.2.10 Traingdx

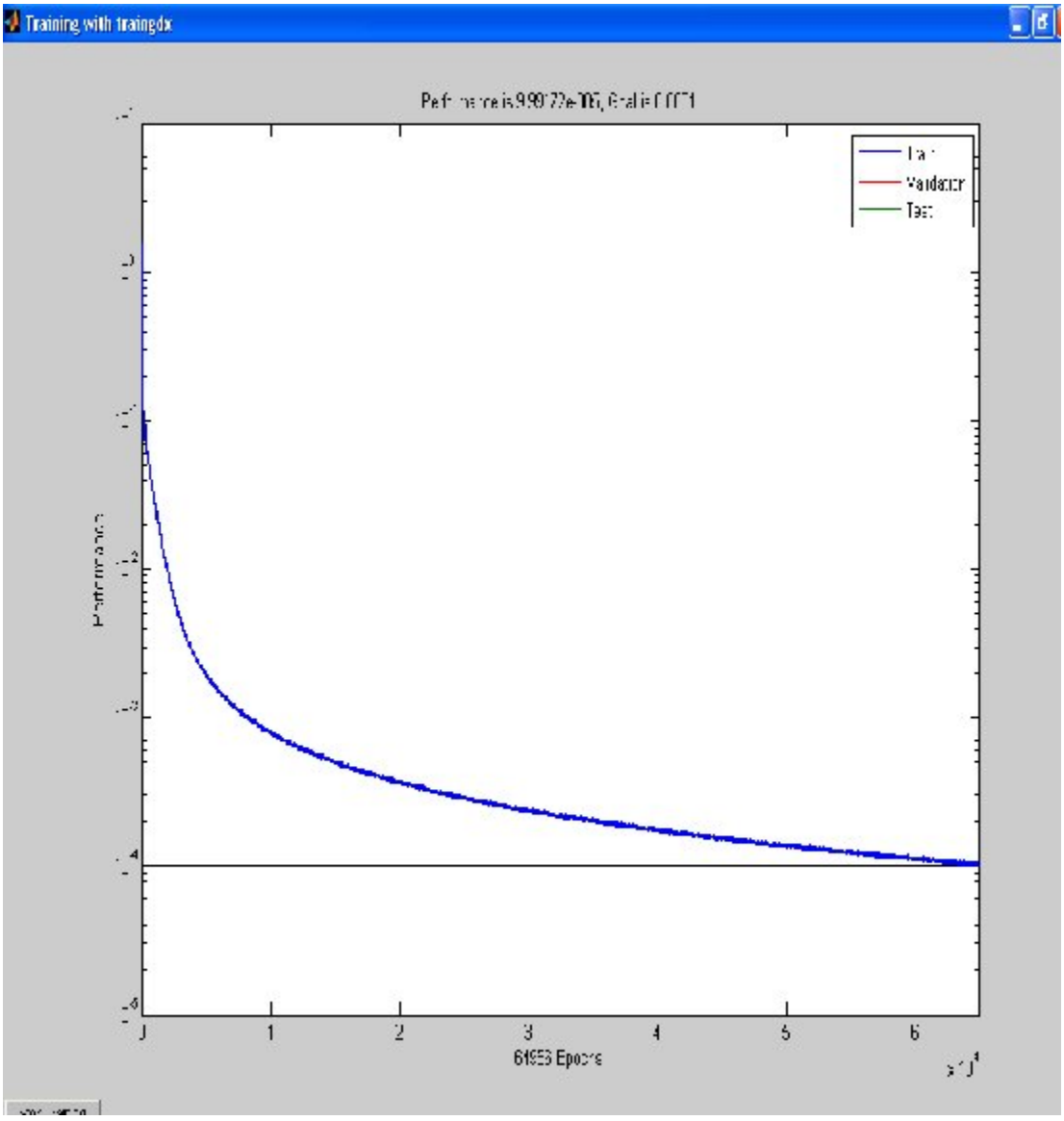

Figure 6.10 Result of training ANN using Traingdx

The graph shown in figure 6.10 represents the output of the training of the network and 64956 epochs have been taken to get trained the network using the traingdx train function. In this case the performance gal of the network has been achieved.

6.2.11 Traingdm

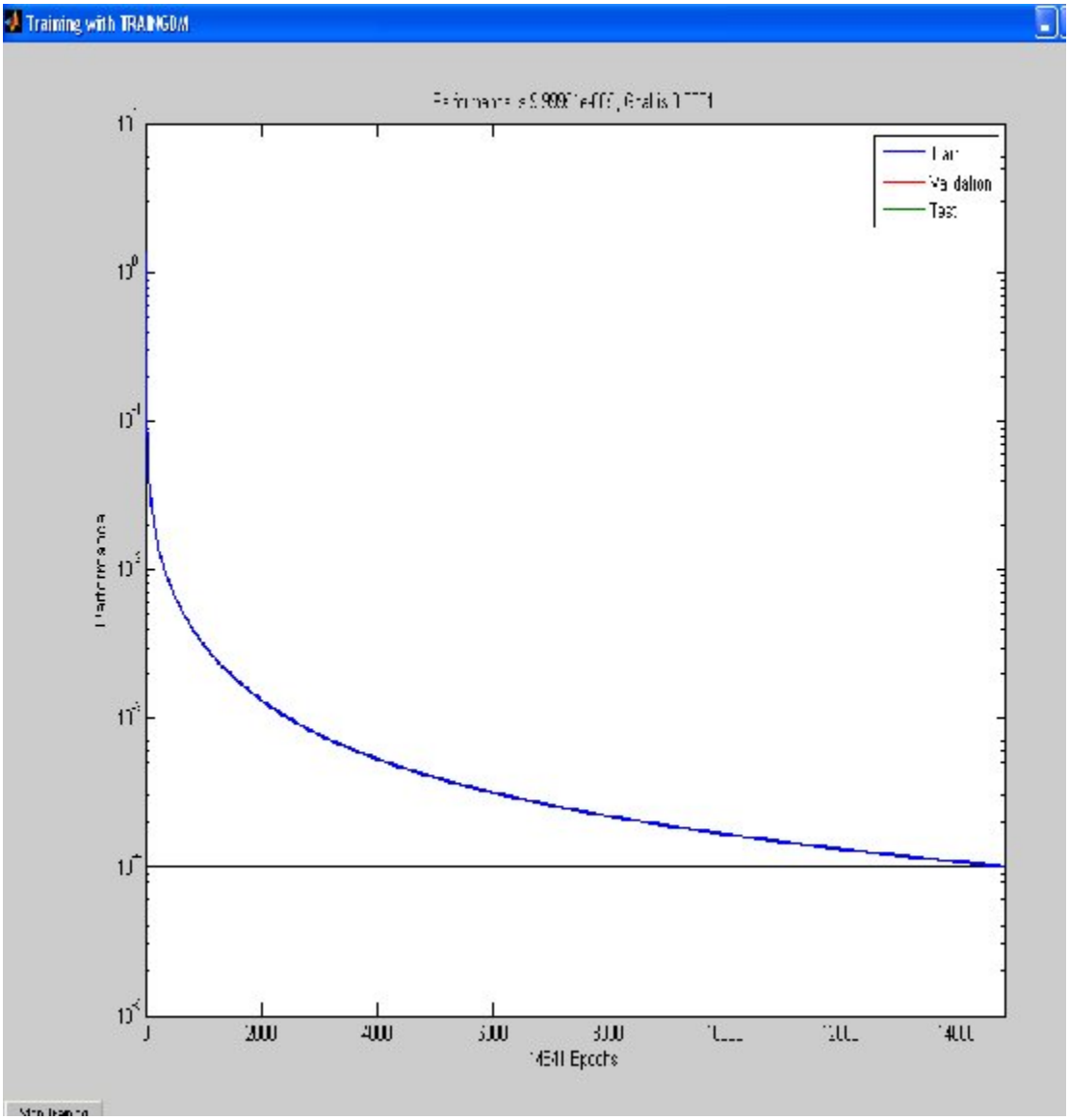

Figure 6.11: Result of training ANN using Traingdm

The graph shown in figure 6.1 represents the output of the training of the network and 14841 epochs have been taken to get trained the network using the traingdm train function. In this case the performance gal of the network has been achieved.

6.3 Experiment 3 (Testing)

6.3.1 Normal traffic and Known attacks

For the testing with normal traffic the sessions that were classified as normal traffic and known attack traffic in the DARPA data sets are used for testing. Here are the results from the experiment with normal traffic and known attacks. In this work 100 sessions of traffic were used to test the network. Out of these 100 sessions, 50 sessions with normal traffic, 25 sessions with known attacks and 25 sessions with unknown attacks. The results shown in 6.4 are of when 4 hidden units are used.

Function	No. of epochs	Classification rate for testing set Normal	known Attacks	Classification rate (%) for testing set Normal	known Attacks
Traingd	76637	46/50	23/25	92	92
Traingdm	90000	47/50	23/25	94	92
Traingdx	49318	43/50	23/25	86	92
Trainrp	63	47/50	22/25	94	88
Traincgf	84	43/50	25/25	86	100
Traincgp	83	48/50	23/25	96	92
Traincgb	55	43/50	24/25	86	96
Trainscg	83	49/50	24/25	98	96
Trainbfg	31	49/50	24/25	98	96
Trainoss	208	44/50	21/25	88	84
Trainlm	11	48/50	22/25	96	88

Table 6.4: Validation of ANN for Normal and Known attacks

6.3.2 Unknown attacks

Unknown attacks are attack types that the neural network has not seen before. Here are the results from the experiment with unknown attacks. The results shown in 6.5 are of when 4 hidden units are used.

Function	Classification rate for testing set Unknown Attacks	Classification rate (%) for testing set Unknown Attacks
Traingd	15/25	60
Traingdm	10/25	40
Traingdx	10/25	40
Trainrp	12/25	48
Traincgf	19/25	76
Traincgp	19/25	76
Traincgb	8/25	32
Trainscg	16/25	68
Trainbfg	15/25	60
Trainoss	10/25	40
Trainlm	16/25	68

Table 6.5: Validation of ANN for Unknown attacks

6.3.3 Normal traffic and Known attacks

For the testing with normal traffic the sessions that were classified as normal traffic in the DARPA data sets are used. Known attacks are attack types that the neural network has been trained with. The sessions that were classified, as known attack traffic in the DARPA data sets are used for testing the known attacks. Here are the results from the experiment with normal traffic and known attacks. The results shown in 6.6 are of when 10 hidden units are used.

Function	**No. of epochs**	**Classification rate for testing set**		**Classification rate (%) for testing set**	
		Normal	**known Attacks**	**Normal**	**known Attacks**
Traingd	13961	47/50	18/25	94	72
Traingdm	14841	46/50	21/25	92	84
Traingdx	64956	45/50	21/25	90	84
Trainrp	109	47/50	22/25	94	88
Traincgf	120	47/50	24/25	94	96
Traincgp	98	50/50	25/25	100	100
Traincgb	91	41/50	23/25	82	92
Trainscg	98	45/50	21/25	90	84
Trainbfg	47	49/50	24/25	98	96
Trainoss	220	50/50	25/25	100	100
Trainlm	4	45/50	20/25	90	80

Table 6.6: Validation of ANN for Normal and Known attacks

6.3.4 Unknown attacks

Unknown attacks are attack types that the neural network has not seen before. Here are the results from the experiment with unknown attacks. The results shown in 6.7 are of when 10 hidden units are used.

Function	Classification rate for testing set Unknown Attacks	Classification rate (%) for testing set Unknown Attacks
Traingd	20/25	80
Traingdm	15/25	60
Traingdx	15/25	60
Trainrp	16/25	64
Traincgf	23/25	92
Traincgp	20/25	80
Traincgb	20/25	80
Trainscg	22/25	88
Trainbfg	21/25	84
Trainoss	24/25	96
Trainlm	17/25	68

Table 6.7: Validation of ANN for Unknown attacks

CHAPTER 7: Conclusion and Future work

INTRODUCTION

This chapter discusses about the conclusions that are made from the results obtained in the previous chapter and also about the Limitations and future work**.**

7.1 Conclusion

Here, a neural network based intrusion detection system, intended to classify the normal and attack patterns has been presented. In the present work, we have presented different training algorithm used to train ANN for modeling Intrusion Detection System. As shown in this work, neural networks can be successfully be used as a method for training and testing an intrusion detection system. In this work, the ability of a backpropagation neural network to classify normal traffic correctly and to detect known and unknown attacks is tested successfully

.

As shown in the previous chapter that to classify 25 out of 25 known attacks using Traincgp, Trainoss training algorithms are very promising results and for unknown attacks classifying rate is 96% i.e 24 out of 25 unknown attacks has been detected using Trainoss, Traincgf training algorithms. In this work different training algorithms has been tested and from these results it can be observed that the performance of trainoss train function is well compare to the other train functions for IDS. The classification rate using trainoss train function is 100% for normal traffic i.e. 50 out of 50, 100% for known attacks i.e. 25 out of 25,for unknown attacks 96% i.e. 24 out of 25 which are very promising results. Thus the aim of this work to design anomaly based intrusion detection system and to select the optimized ANN train algorithm is achieved.

7.2.1 Limitations and Future work

In this work there is need to regular update of the signature database, need to consider know and unknown attack. Only detect, cannot prevent the intrusions and it's an offline system. In the future work this system can be extended to an online system by little effort. There has been a lot of research on intrusion detection, and also on the use of neural networks in intrusion detection. As showed in this thesis, backpropagation neural networks can be used successfully to detect attacks on a network. The same experiments should also be conducted with other types of neural networks to see if these types can improve the detection rate we got from the experiments with a backpropagation neural network. In an online system, the real network is directly connected with the intrusion detection system the network traffic can be checked by the system periodically for every hour or so.

References

[1]. J. Ryan, M. J. Lin and R. Miikkulainen, “Intrusion Detection with Neural Networks”, Advances in Neural Information Processing Systems, Vol. 10, Cambridge, MA: MIT Press, 1998, pages 943-949.

[2]. Y. Wang, G.X.Huang, and D.G. “Model of Network Intrusion Detection System based on BP Algorithm”, in proceedings of Industrial Electronics and Applications, 2006 1ST IEEE Conference on Peng, pages 1-4.

[3]. Aurobindo Sundaram ”An Introduction to intrusion detection”, Crossroads, Issue 2.4, computer security 1996 http://www.acm.org/crossroads/xrds2-4/intrus.html.

[4].Jean-Philippe PLANQUART, “Application of Neural Networks to Intrusion Detection Introduction”, SANS Institute, 2001.

[5]. “The Computer Technology Documentation Project” a site that share information http://www.comptechdoc.org/independent/security/recommendations/secattacks.html.

[6]. "Honeypots: Definitions and Value of Honeypots" by Lance Spitzner.

[7]. sandeep sharma “Application of Neural Networks to Intrusion Detection Classification and detection of computer intrusions “by -2005.

[8]. Introduction to NEURAL NETWORKS by Christos Stergiou and Dimitrios Siganos http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html.

[9]. Introduction to Artificial Neural System by Jack M. Zaruda.

[10]. Introduction to neural networks using MATLAB by SN Sivanandan, S. Sumathi.

[11]. Robert Birkely “A Neural Network Based Intelligent Intrusion Detection System” (2003), Maser thesis, School of Information Technology, Griffith University. http://docs.ksu.edu.sa/PDF/Articles34/Article340530.pdf pages 22-40.

[12]. Howard Demuth, Mark Beale, Martin Hagan, “Neural Network Toolbox™ 6”, User Guide, COPYRIGHT 1992–2008 by The MathWorks, Inc.

[13]. “Neural networks in intrusion detection system” by A. Vessley, D. Brechlerova pages 35-39.

[14]. Lawrence, Jeanette (1994) “Introduction to Neural Network”, California Scientific Software Press.

[15].”Intrusion detection with artificial neural networks “ by moazzam hossain February 10, 2004, pages 25-40.

[16]. Debar, H.Becker, and Siboni, D. (1992). “A neural network component for an intrusion detection system” In Proceedings of the 1992 IEEE.

[17]. Srinivas Mukkamala1 & Andrew H. Sung “ Feature Selection for Intrusion Detection using Neural Networks and Support Vector Machines”, Pages 8-14. http://www.icasa.nmt.edu/Content/publication/feature.pdf.

[18]. Mehdi Moradi and Mohammad Zulkernine “A Neural Network Based System for Intrusion Detection and Classification of Attacks”, presented in the Natural Sciences and Engineering Research Council of Canada (NSERC). Pages 1-6.

[20]. S. B. Cho, “Incorporating Soft Computing Techniques Into a Probabilistic Intrusion Detection System”, IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews, Vol. 32, No. 2, pp. 154-160, May 2002.

[21]. In 2005 “A Data Mining based Adaptive Intrusion Detection Model (DMAIDM) is presented by Tian-Qing Zhu pages 2390-2395.

[22].Srinivas mukkarnala, Andrew sung, intrusion detection using neural networks and supportvector machines,http://www.cs.nmt.edu/it/papers/hawaii7.pdf pages 3-6.

[23]. “Intrusion detection using an ensemble of intelligent paradigms” by Srinivas Mukkamala, Andrew H. Sung and Ajith Abraham.

[24]. Elaine Rich, Kevin Knight, Artificial Intelligence Tata-McGraw Hill Publishing company Limited, New Delhi.

[25]. DARPA Intrusion Detection Evaluation, Lincoln Laboratory, Massachusetts Institute of Technology, http://www.ll.mit.edu/IST/ideval/index.html.

[26]. L. D. Paulson,” Wanted: More Network-Security Graduates and Research”, IEEE Computer, Vol. 35, No. 2, pp. 22-24, February 2002.

[27]. Mukherjee, B., Heberlein, L.T., Levitt, K.N. (May/June, 1994). “Network Intrusion Detection”. IEEE Network. Pages 28-42.

[28]. Frank, Jeremy. “Artificial Intelligence and Intrusion Detection”. In Proceedings of the 17th National Computer Security Conference, 1994, pages 166-169.

[29]. Sung A. H. (1998) “Ranking Importance of Input Parameters of Neural Networks,” Expert Systems with Applications, Vol. 15, pages 405-41.

[30]. L. D. Paulson, “Stopping Intruders Outside the Gate”s, IEEE Computer, Vol. 35, No. 11, pp. 20-22, November 2002

Paper Communicated

1. Gnana Prasad. M, V. P. Singh **"Modeling Intrusion Detection System by Optimized Selection of ANN Training Algorithms"** at national conference on "Emerging trends in Information Technology", NCEIT-2008, Institute of Engineering and Emerging Technologies, Baddi, Solan. [Communicated]