Al Al-Bayt University

Prince Hussein bin Abdullah College of Information Technology

Computer Science Department

Intrusion Detection Using Feed-forward Neural Networks

By

Khattab Mejeal Ali Alobedy

2008

بسم الله الرحمن الرحيم

Al Al-Bayt University

Prince Hussein bin Abdullah College of Information Technology

Computer Science Department

Intrusion Detection Using Feed-forward Neural Networks

By

Khattab Mejeal Ali Alobedy

2008

Intrusion Detection Using Feed-forward Neural Networks

كشف الاختراق باستخدام الشبكات العصبونية ذات التغذية الأمامية

By

Khattab Mejeal Ali Alobedy

0620901007

Supervisor: Dr.Venus W. Samawi

A Thesis Submitted to the

Scientific Research and Graduate Faculty in partial fulfillment of the

Requirements for the Degree of Master of Science

in Computer Science

Members of the Committee	Approved
Dr.Venus W. Samawi	_____
Prof. Adnan Al-Smadi	_____
Dr. Jehad Nhood	_____
Dr. Iseam Al-Dawaid	_____

Al Al-Bayt University

Mafraq, Jordan

2008

بسم الله الرحمن الرحيم

قَالَ هَذَا مِن فَضْل رَبِّي لِيَبْلُوَنِي
أَأَشْكُرُ أَمْ أَكْفُرُ وَمَن شَكَرَ فَإِنَّمَا
يَشْكُرُ لِنَفْسِهِ وَمَن كَفَرَ فَإِنَّ رَبِّي
غَنِيٌّ كَرِيمٌ

صدق الله العظيم

النمل-

40

Dedication

My greatest appreciation awarded to my beloved parents who taught me the significance of education.

Acknowledgements

First of all, profuse thanks to Allah who helped me and gave me the ability to achieve this work.

I would like to express my thanks to Dr. Venus W. Samawi, the supervisor, for her able guidance, suggestion, and supervision during the development of this research. Without her help and contributions, this work could not have been accomplished.

I wish to thank professor Al-Smadi for his helpful suggestions and comments which contributed to the improvement of this thesis.

Appreciation is extended to all professors at Computer Science Department especially to Dr. Said Bani-Mohamed, the head of computer science department, Dr. Jehad Al-khaldi, and to Dr. Mamoun S. Al Rababaa for their support and cooperation. Thanks to the staff members of department of computer science.

Special thanks for the examination committee for their valuable and important scientific comments. In addition, I would like to express thanks to my friends Azait, Ayad, Marof and Ahmed for their great help and support.

Finally, Sincere thanks to my brothers Zaid, Harth, Maolod, Ali for their supper-sincere appreciation.

Khattab Ali Alobedy

Table of contents

List of Tables

List of Figures

List of Abbreviations

Abbreviations	Meaning
ANN	Artificial Neural Network
BP	Backpropagation
BN	Bayesian Networks
CART	Classification And Regression Trees
DARPA	Defense Advanced Research Project Agency
DOS	Denial-Of-Service
FNN	Fuzzy Neural Network
FNT	Flexible Neural Tree
FL	Fuzzy Logic
GA	Genetic Algorithm
HIDS	Host Intrusion Detection System
IDS	Intrusion Detection System
ID	Intrusion Detection
IT	Information Technologies
KDD	Knowledge Discovery Database
MARS	Multivariate Adaptive Regression Splines
MIT	Massachusetts Institute of Technology
NID	Network Intrusion Detection
NN	Neural Network
PC	Personal Computer
PHIDS	A parallel hierarchical Intrusion Detection System
R2L	Remote to local
RBF	Radial Basis Functions
SVM	Support Vector Machines
SC	Soft Computing
SCIDS	Soft Computing Intrusion Detection System
TCP	Transmission Control Protocol
U2R	User to Root

Abstract

Intrusion detection is an interesting approach that could be used to improve the security of network systems. An Intrusion detection system (IDS) detects suspected patterns of network traffic on the remaining open parts through monitoring user activities (runtime gathering of data from system operations), and the subsequent analysis of these activities. The main problem is the difficulty of distinguishing between natural behavior and abnormal behavior in computer networks due to the significant overlap in monitoring data. This detection process generate (False Alarms) resulting from the use of intrusion detection based on the (Anomaly Intrusion Detection Systems). The use of Fuzzy Set might reduce the amount of false alarm, where the degree of relationship to the use of any process for separation of this overlap could be used to define normal and abnormal behavior in computer networks. For that ***data fuzzification*** is needed before classification.

The purpose of this work is to contribute ideas of finding a solution to detect attacks (Intrusion) through building Intelligent detection system using feed-forward neural networks to detect attacks with low false negative rate (which is the most important point), and low false positive rate. To do so, two feed-forward neural networks architectures (one for non fuzzified data, the other for fuzzified data) are suggested, and their behaviors in detecting the attacks are studied.

To evaluate the performance of the proposed IDS, a standard set of data KDD (knowledge Discovery in Database) proposed by Massachusetts Institute of Technology's (MIT) Lincoln Labs is used. The dataset can be divided mainly into five categories (***Normal data***, ***Probing attack***, ***Dos attack***, ***U2R attack***, ***R2L attack***). The suggested neural networks were trained with reduced feature set (12 out of 41 features), different neural network architectures were tested and the most proper one was used.

In this research, the suggested IDS not only has the ability to distinguish if the access is normal or attack, but also capable of distinguishing the attack type. The suggested classifiers were tested over the entire dataset to evaluate real word performance. The preliminary results are promising at which the accuracy percentage is about (95.9303%) for Neural Network (NN) trained with non fuzzified dataset, while, the proportion of precision in the classification of the data after fuzzification is about (97.4890%).

Chapter One

Preface

1.1 Introduction

Now a days, networks and internets are heavily used due to the increasing need of people to gain information from different remote sources that are scattered all over the world as fast as possible. With the rapid growth of computer networks, security has become an essential problem for modern computer systems. Recent events have highlighted the need for fast reactionary in network security due to the increasing intrusion events.

The worldwide impact of malicious code attacks is estimated to be over $10 Billion annually. Novice attackers can easily acquire and use automated denial-of-service attack software. As Internet access becomes more affordable, not only does the number of attacks increase quickly, the attacks are also becoming more and more sophisticated. This is due to the increased understanding of how systems work. Network intruders (attackers) easily overcome the password authentication mechanism, which is considered as one of the major defense lines used for system protection. Intruders start to use patterns for intrusion that are difficult to trace and identify. They cover their tracks so that their system crack activity is not easily discovered. The major problem becomes *How to detect and respond to the attacks?*

Access control mechanism could be used as first line of defense although it dose not model or restrict what a subject may do with the object itself if it has the access to manipulate it. Access control therefore does not model and cannot prevent unauthorized information flow through the system because such flow can take place with authorized accesses to the objects. Moreover, in systems where access controls are discretionary, the responsibility of protecting data rests on the end user. This often requires that users understand the protection mechanisms offered by the system and how to achieve the desired security using these mechanisms [Den82, Kum95].

Access controls are not helpful against insider threats which historically, the vast majority of computer crimes were committed by people from within the organization. Therefore several methods are available to protect a computer system or network from attacks, a strong perimeter defense begin only one of them. Follows are the listings of the

most popular, nonexclusive approaches to anti-intrusion techniques (depicted in Figure (1.1)) [Pri97, Axe99]:

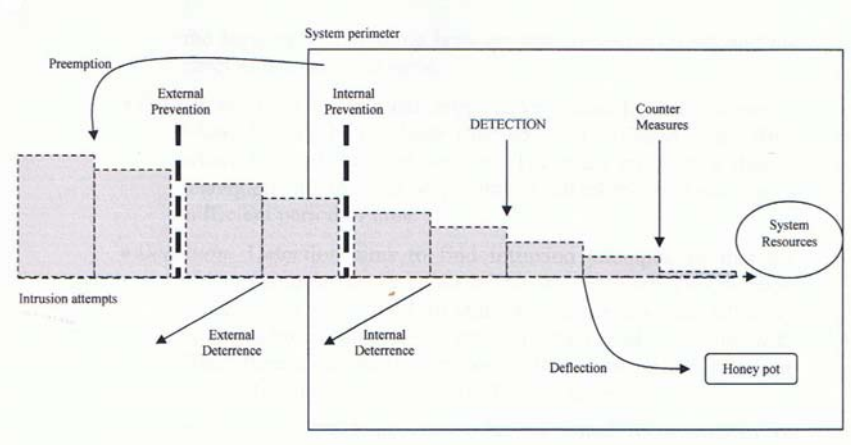


Figure (1.1) summary of anti-intrusion techniques [Pri97]

- Prevention: it means preventing the probability of a particular intrusion to happen. Internal prevention is under the system owner control, while external prevention takes place in the environment surrounding the system, such as a larger organization, or society as a whole.
- Deterrence: persuade an attacker to hold off his attack, or to break off an ongoing attack which can be done by increasing the perceived risk of negative consequences for the attacker. External prevention could be affected by the legal system, making laws against computer crimes.
- Deflection: attract an intruder into thinking that he has succeeded when, in fact, he has been shunted off, or tricked away, from where he could do real damage. The main problem is that of managing to deceive an experienced attacker, at least for a sufficient period of time.
- Detection: Detection aims to find intrusion attempts, so that the proper response can be evoked. Problems include the obvious: difficulty of defending against a hit-and-run attack and problems with false alarms, or failing to sound the alarm when someone secretly gains or attempts to gain access. All security incidents cannot be prevented for a number of reasons: systems contain vulnerabilities that cannot easily or quickly be found and fixed. Building secure systems without any vulnerability is extremely difficult, and secure systems are still vulnerable to

insider abuse and mistakes. Therefore, *detection of intrusive activity and other security incidents is an important line of defense*. Additionally, detection of attempted attacks, even when not successful, is important for a computer system's defense. *Monitoring the computer system to detect problems in a timely fashion minimizes damage from security incidents, establishes accountability, and deters threads*.

The key attributes of rigorous computer security are *confidentiality*, *integrity* and *availability* [Eri00]. The main approach to satisfy these requirements is intrusion prevention. *Intrusion prevention is an attempt to eliminate various security vulnerabilities and resolve ambiguities in passive network monitoring by insertion of intrusion detection system (IDS) in-line*. The main goal of intrusion detection is to detect unauthorized use, misuse and abuse of computer systems by both system insiders and external intruders [And80, Man02]. The costs of temporary or permanent damages caused by unauthorized access of the intruders to computer systems have urged different organizations to increasingly implement systems that monitor data flow in their networks [Kem02]. These systems are generally referred to as Intrusion Detection Systems (IDSs). IDS becomes necessary for computer security to enhance existing defenses.

1.2 Intrusion Detection Systems

An intrusion can be defined as ''an act of a person of proxy attempting to break into or misuse a system in violation of an established policy'' [Mal02]. So to protect systems from intruders, intrusion detection system is needed. ID is software and/or hardware system for monitoring and detecting data traffic or user behavior to identify attempts of illegitimate accessing system manipulation through a network by malware and/or intruders (crackers, or disgruntled employees). ID has been used to protect information systems along with prevention-based mechanisms such as authentication and access control. An ID cannot directly detect attacks within properly encrypted traffic.

Intrusion detection systems can be classified as *network-based* and *host-based* according to the information source of the detection. Network-based IDS monitors the network traffic and looks for network-based attacks, while host-based IDS is installed on host and monitors the host audit trail. Intrusion detection systems can be roughly classified

as *anomaly detection* and *misuse detection*. Anomaly detection is based on the normal behavior of a subject (e.g., a user or a system). Any action that significantly deviates from the normal behavior is considered intrusive. Misuse detection is based on the characteristics of known attacks or system vulnerabilities, which are also called *signatures*. Any action that matches the signature is considered intrusive. Both anomaly detection and misuse detection have their limitations.

Misuse-base detection detects attacks based on signatures (known attacks signatures), at which the traffic pattern compared with these signatures, if a match is found, then it is reported as an attack, otherwise it is not. So misuse detection cannot detect novel attacks. On the other hand, anomaly-based detection depends on monitoring system activity and classifying it as either normal or anomalous. The classification is based on heuristics or rules, rather than patterns or signatures, and will detect any type of misuse that falls out of normal system behavior.

The strength of the anomaly detection approach is that prior knowledge of the security flaws of the target systems is not required. Thus, it is able to detect not only known intrusion but also unknown intrusion. In addition, this approach can detect the intrusion that is achieved by the abuse of legitimate users or masqueraders without breaking security policy [Den87, Por92]. However, it has several limitations, such as high false positive detection error, the difficulty of handling gradual misbehavior and expensive computation [Myk94]. In contrast, the misuse detection approach detects only previously known intrusion signatures. The advantage of this approach is that it rarely fails to detect previously notified intrusions [Den87]. However, this approach cannot detect new intrusions that have never previously been monitored. Furthermore, this approach is known to have other drawbacks such as the inflexibility of misuse signature rules and the difficulty of creating and updating intrusion signature rules [Por92, Kum95]. These strengths and limitations of the two approaches imply that effective IDS should employ an anomaly detector and a misuse detector in parallel [Myk94]. However, most available commercial IDS's use only misuse detection because most developed anomaly detector still cannot overcome the limitations described above. This trend motivates many research efforts to build anomaly detectors for the purpose of ID [Kim02]. However, the nature of current and

future threats in conjunction with ever larger Information Technologies (IT) system systems urgently requires the development of automated and adaptive defensive tools.

1.3 Problem Definition

Recently, the problem of computer systems intrusions grows and intruders become intelligent and flexible agents. The reason is that, new automated hacking tools appear every day, and these tools, along with various system vulnerability information, are easily available on the web. This problem can be solved by using appropriate software which is designed to detect and prevent intrusions.

Intrusion detection (ID) is an appealing concept since the current techniques used in computer security are not able to cope with dynamic and increasingly complex nature of computer systems and their security. The intrusion detector learning task is to build a predictive model (i.e. a classifier) capable of distinguishing between "bad" connection, called intrusion or attack, and "good" normal connections depending on special attributes (features) that are collected from the packet header and audit trail files (behavior during the connection). Such classifiers could be built using different approaches (statistical approaches, genetic algorithms, fuzzy systems, or neural networks).

To evaluate any intrusion detection system, dataset collected by Defense Advanced Research Project Agency is used. This dataset is a version of the 1999 DARPA intrusion detection evaluation data set prepared and managed by MIT Lincoln Labs. In this data set, 41 attributes that describe the different features of the corresponding connection (22 of these features describe the connection itself and 19 of them describe the properties of connections to the same host in last two seconds).

There are 39 different attack types presented and falls exactly into one of the following four categories [Sto00]:

1. Probing (surveillance and other probing): Probing is a class of attack where an attacker scans a network to gather information or find known vulnerabilities.
2. DOS (denial-of-service): it is a class of attack where an attacker makes a computing or memory resource too busy or too full handles legitimate requests thus denying legitimate users access to a machine.
3. U2R (User to root): unauthorized access to local super user (root) privileges exploits. It is 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.

4. R2L (A remote to local): unauthorized access from a remote machine. It is a class of attack where an attacker sends packets to a machine over a network, then exploits the machine's vulnerability to illegally gain local access as a user.

Another problem, current IDS examine all data features to detect intrusion or misuse patterns, use all the feature adds extra burden on the system, and extra features can increase computation time, and can impact the accuracy of IDS. Therefore, it is important to reduce number of features (attributes) that would be used to train the classifier. One of the major problems is to select the proper attributes (from the total 41 attribute in dataset) that have the best discrimination ability (between normal and attack connections). It is also important to choose the suitable classifier with high classification rate.

1.4 Research Objectives

One way to detect illegitimate use in network system is through monitoring unusual user activity by using IDS. Different methods used to build intrusion detection system (such as statistical, genetic, fuzzy genetic, neural networks etc). This research aims to study the classification ability of feed-forward neural network with actual data and feed-forward neural network with fuzzified data and compare their results from distinguishing accuracy point of view. Also, use reduced number of attributes (12 out of 41 attributes suggested by Chebrolu [Che05]) since some of these features will not affect the classification accuracy (or even may negatively affect it) at which their values are the same in different attack types or normal one. The main goal of this research is to improve classification rate of the discrimination ability (i.e. discriminate attacks from normal behavior).

In additional, most of the previous studies focused on classification of records in one of the two general classes-normal and attack, this research aim's to solve a multi-class problem at which the type of attack is also detected by the suggested intrusion detector. Using the reduced data sets, 5-class classifier is built (normal data belongs to class 5, probe belongs to class 1, denial of service belongs to class 2, user to super user belongs to class 3, remote to local belongs to class 4). The dataset is partitioned into 3 different parts (validation part, training part, and testing part). The evaluation of the system accuracy will depend on the testing results.

1.5 Literature Survey

With respect to the research on IDS's, the intrusion detector and neural networks attracted a growing number of computer scientists and they have proposed several different intelligent systems. Various types of classifier were used by the researchers to detect intrusions; this work is concerned with using ***neural networks with fuzzified data*** and ***neural networks with non-fuzzified data*** as classifiers. Following are some works related to the above ideas:

1. Dipankar Dasgupta and Fabio A. Gonzalez [Das01] proposed a linear representation of tree structures in order to evolve complex fuzzy rules sets for solving classification problems. In particular, linguistic rules are evolved, where the condition part of a rule can have an arbitrary structure of conjunctions and disjunctions. They describe an efficient rule representation scheme, which uses a GA. The method is tested with a number of benchmark datasets. Their experiments showed that the proposed representation works well in a wide variety of classification problems. Despite the fact that only five values for the linguistic variables were used, the accuracy of the evolved classifier rules 94.8% for specific data set.
2. Adhity Chittur [Chi01] presented a novel approach to detect intrusion by using GA, which was used to learn how to separate malicious intrusions from normal use. The algorithm was then tested in a real-world simulation to measure its effectiveness under unpredictable conditions. This model was then tested over previously unseen data to gauge its real-world performance. The result shows that the GA was successfully able to generate an accurate empirical behavior model from training data and then able to successfully apply this empirical knowledge to data never seen before. The final model produced had an overall accuracy level of 97.8% which showed both a high detection rate and an extremely low false positive rate.
3. J.T Yao, S.L. Zhao, L.V. Saxton [Yao02] proposed a dynamic approach that tries to discover known or unknown intrusion detection patterns. A dynamic fuzzy boundary is developed from labeled data for different levels of security needs. In additional, a method to develop a dynamic decision boundary based on SVM (support vector machines) and fuzzy logic has been introduced. The experiment results show that users can adjust dynamic

boundary easily for different requests of accuracy and computation complexity. Furthermore, using SVM provides the ability of handling a large number of features efficiently. It is also possible to build a dynamic decision boundary, using other popular artificial intelligence techniques such as neural networks, decision tree and Bayesian Networks.

4. Jonatan Gomez, Fabio Gonzalez, and Dipankar Dasgupt [Gom03] presents a new technique that allows generating a set of fuzzy rules that characterize the non-self space (abnormal) using as input only self (normal) samples. This work extended a previous work that used crisp rules as detectors. The experiments results show that the proposed approach performs better than the previous one and is comparable with other results reported in the literature. The main advantages of this approach are: providing better definition of the boundary between normal and abnormal, shows an improved accuracy on the anomaly detection problem (this can be attributed to the fuzzy representation of the rules which reduce the search space), and allowing the EA to find better solutions.

5. Srinivas Mukkamalaa, Andrew H. Sunga, Ajith Abraham [Muk04] addresses using an ensemble approach of different soft computing and hard computing techniques for intrusion detection. Due to increasing incidents of cyber attacks, building effective intrusion detection systems are essential for protecting information systems security, and yet it remains as vague goal and a great challenge. They studied the performance of Artificial Neural Networks (ANNs), Support Vector Machines (SVMs) and Multivariate Adaptive Regression Splines (MARS). They found that an ensemble of ANNs, SVMs and MARS is superior to individual approaches for intrusion detection in terms of classification accuracy. Finally, the importance of using ensemble approach for modeling IDSs is clearly shown. An ensemble helps to indirectly combine the synergistic and complementary features of the different learning paradigms without any complex hybridization.

6.Chunlin Zhang, Ju Jiang, Mohamed Kamel [Zha05] use two hierarchical IDS frameworks using Radial Basis functions (RBF), a serial hierarchical IDS (SHIDS) is proposed to identify misuse attack accurately and anomaly attacks adaptively. A parallel hierarchical IDS (PHIDS) is proposed to enhance the SHIDS's functionalities and performance. Furthermore, there are two objectives: the first objective is to find a suitable method, which can be applied to intrusion detection with less training time, high detection rates and less

false positive rates. Because of the many advantages of neural networks, BPL and RBF algorithms are applied to train neural network based intrusion detectors (classifiers) for IDSs. The second objectives of the paper are to design IDS with the abilities of detecting both misuse and anomaly, and updating its structure for novel attacks.

7.Ajith Abraham, Ravi Jain, Sugata Sanyal, and Sang Yong Han [Abr05] evaluates three fuzzy rule based classifiers for IDS and the performance is compared with decision tree, support vector machines and linear genetic programming. Further, Soft Computing (SC) based IDS (SCIDS) is modeled as an ensemble of different classifiers to build lightweight and more accurate (heavy weight) IDS. Empirical results clearly show that SC approach could play a major role for ID. They have illustrated the importance of SC paradigms for modeling IDSs. For real time IDSs, linear Genetic programming would be the ideal candidate as it can be manipulated at the machine code level. Overall, the fuzzy classifier (FR2) gave 84.396% accuracy for all attack types using all the 12 attributes. The proposed hybrid combination of classifiers requires only 12 input variables.

8. Srilatha Chebrolu, Ajith Abraham, and Johnson P. Thomas [Che05] this study is to identify important input features in building an IDS that is computationally efficient and effective. The performance of two feature selection algorithm involving Bayesian networks (BN) and classification and Regression Trees (CART) and an ensemble of BN and CART are investigated. A hybrid architecture involving ensemble and base classifiers for intrusion detection were proposed. From the empirical results, it is seen that by using the hybrid model Normal, Probing and DOS could be detected with 100% accuracy and U2R and R2L with 84% and 99.47% accuracies, respectively. Finally, some data sources can be eliminated using feature selection. A support vector machine technique is used to select the important features (12 features were selected).

9. Yuehui Chen, Ajith Abraham, Bo Yang [Che07] try to identify important input features in building an IDS that is computationally efficient and effective. This article proposes an IDS model based on a general and enhanced flexible neural tree (FNT). Based on the predefined instruction/operator sets, a flexible neural tree model can be created and evolved. This framework allows input variables selection, over layer connections, and different activation functions for the various nodes involved. The (FNT) structure is developed using an evolutionary algorithm, and the parameters are optimized by a particle

swarm optimization algorithm. The performance using different reduced data sets were demonstrated. The proposed flexible neural tree approach seems to be very promising. Using 41 variables, the FNT model gave the best accuracy for the detection of most of the classes (except U2R). Although the hybrid model seems to work very well for most of the attack classes, the direct NN classifier outperformed the FNT approach for U2R attack. For the 12-variable reduced data set, the direct NN approach outperformed the FNT model for DOS, U2R, and R2L attacks.

10. Ray-I Chang, Liang-Bin Lai, Wen-De Su, Jen-Chieh Wang, Jen-Shiang Kouh [Cha07] propose a new learning methodology towards developing a novel intrusion detection system (IDS) by Backpropagation neural networks (BPN) with sample-query and attribute-query. They test the proposed method by a benchmark intrusion dataset to verify its feasibility and effectiveness. Results show that choosing good attributes and samples will not only have impact on the performance, but also on the overall execution efficiency. The proposed method can significantly reduce the training time required. Additionally, the training results are good. It provides a powerful tool to help supervisors analyze, model and understand the complex attack behavior of electronic crime.

1.6 Chapters Overview

In this section, the contents of individual chapters of this research are briefly reviewed.

- ***Chapter Two*** (Theoretical Background) consists of three parts: Part one describes the ID, classification and data processing techniques used in IDS. Part two describes the artificial neural networks, classification, major Components of an Artificial Neurons and feed-forward Neural Network Backpropagation (BP). Part three describes Crisp set and Fuzzy set.
- ***Chapter Three*** (Design and Implementation) consists of three parts: Part one describes the data source, data set, feature set and pre-processing data. Part two describes the fuzzy membership and type of alarms. Part three describes Architecture the artificial neural network with non-fuzzified data and Architecture with fuzzified data.

- ***Chapter Four*** (Assessment Results) consists of three part: Parts one describes the General Data Splitting Method. Part two describes how to improved generalization. Part three describe the Experiment Results(analysis and comparison), train and test neural network with non-fuzzified data, result the train and test neural network with non-fuzzified data and result train and test neural network with fuzzified data.
- ***Chapter Five*** (Conclusion and future works) concentrates on conclusions with recommendations for future works.

Chapter Two
Theoretical Background

2.1 Introduction

The world is witnessing the development in both wide computer systems and networks. This development made the Internet available and simple. Unfortunately, this development has increased the number of attacks and intrusions (informal or unauthorized access). For this reason, the need to develop defense systems that detect these intruders becomes essential.

Various processing techniques could be used to implement IDS depending on the technique used in intrusion detection such as experts systems, signature analysis, statistical analysis, computer immunology, or neural networks. In this work, the concentration is on using neural networks. This chapter is concerned with exploring intrusion detection systems, categories, types, in addition to illustration of neural network and fuzzy concepts.

2.2 Overview of Intrusion Detection System

To understand what is ID, the meaning of intrusion should be declared. An intrusion can be defined as [Kum95, Bac01]: "*Any set of actions that attempt to compromise the integrity, confidentiality, or availability of a resource*". Intrusion detection becomes essential security mechanism to protect systems and networks. It attempts to detect improper or inconsistent activity in a computer network, or on a host, by the exploration of certain kinds of data through monitoring. Such activities may be initiated from external intruder or from internal misuse. According to the monitored system, IDS could be categorized into [Tao07, Sun02, Cro03, and Kaz04]:

- ***Network-based IDS***: is an independent platform that monitors the network backbones and look for attack scenarios by analyzing, examining, and monitoring network traffic data. Network Intrusion Detection Systems (NIDS) gain access to network traffic by connecting to a hub, network switch configured for port mirroring, or network tap. The NIDS reassemble and analyze all network packets that reach the network interface card. They do not only deal with packets going to a specific host – since all the machines in a network segment benefit from the

protection of the NIDS. Network-based IDS can also be installed on active network elements, for example on router.

- ***Host-based IDS***: reside on a particular computer and tries to detect malicious activity and provide protection for a specific computer system by monitoring the operating and file systems for signs of intrusion. This can be done through an agent placed on that host to identify intrusions by analyzing system calls, application logs, file-system modifications (binaries, password files) and other host activities and state.
- ***Hybrid*** of HIDS and NIDS, Host agent data is combined with network information to form a comprehensive view of the network. the main reason for introducing such hybrid IDS is the need to work online with encrypted networks and their data destined to the single host (only the source and destination can see decrypted network traffic).

The fact is that intrusion detection systems are not organs of what regulations deter more control and monitor, alarm any detect process will determine the intruder and where breach occurred, when and what the response depends on the design of the system. In addition, the alarm does not provide system security itself; it only to indicate that some sort of potentially malicious activity is being attempted.

2.2.1 Classification of IDS

ID is a network security tool that concerned with the detection of illegitimate actions. This network security tool uses one of two main techniques [Tao07, Cro03, and Kaz04]:

- ***Anomaly detection***, explores issues in ID associated with deviations from normal system or user behavior. It is based on the assumption that the characteristics of attacks are significantly different from normal behavior. Anomaly detection is capable of detecting unknown attacks or variants of known attacks if such attacks significantly change the monitored characteristics of the system. And deviations of normal usage of programs regardless of whether the source is a privileged internal user or an unauthorized external user. The disadvantage of the anomaly detections approach is that well-known attacks may not be detected, particularly if they fit the established profile of the user. Another drawback of many anomaly detection

approaches is that a malicious user who knows that he or she is begin profiled can change the profile slowly over time to essentially train the anomaly detection system to learn the attacker's malicious behavior as normal.

- The second employs ***Misuse*** (Signature detection) refers to known attacks that exploit the known vulnerabilities of the system to discriminate between anomaly or attack patterns (signatures) and known ID signatures. The main disadvantage of misuse detection approaches is that they will detect only the attacks for which they are trained to detect (i.e. not capable of detecting novel or unknown attacks).

The IDS can operate as **standalone, centralized** application or integrated applications that create a **distributed system**. One may categorize IDSs in terms of behavior i.e., they may be ***Passive*** (those that simply generate alerts and log network packets). They may also be ***active*** which means that they detect and respond to attacks, attempt to patch software holes before getting hacked or act proactively by logging out potential intruders, or blocking services.

IDSs can run on either a continuous or periodic feed of information (Real-Time IDS and Interval-base IDS respectively) and hence they use two different ID approaches. ***Audit trail analysis*** is the prevalent method used by periodically operated systems. In contrast, the IDS deployable in real-time environments are designed for online monitoring and analyzing system events and user actions.

With ***on the fly*** processing, an ID performs verification of system events. Generally, a stream of network packets is constantly monitored. With this type of processing, ID uses the knowledge of current activities over the network to sense possible attack attempts (it does not look for successful attacks in the past). Figure (2.1) shows the classification of IDS from different point of views [Kaz04]:

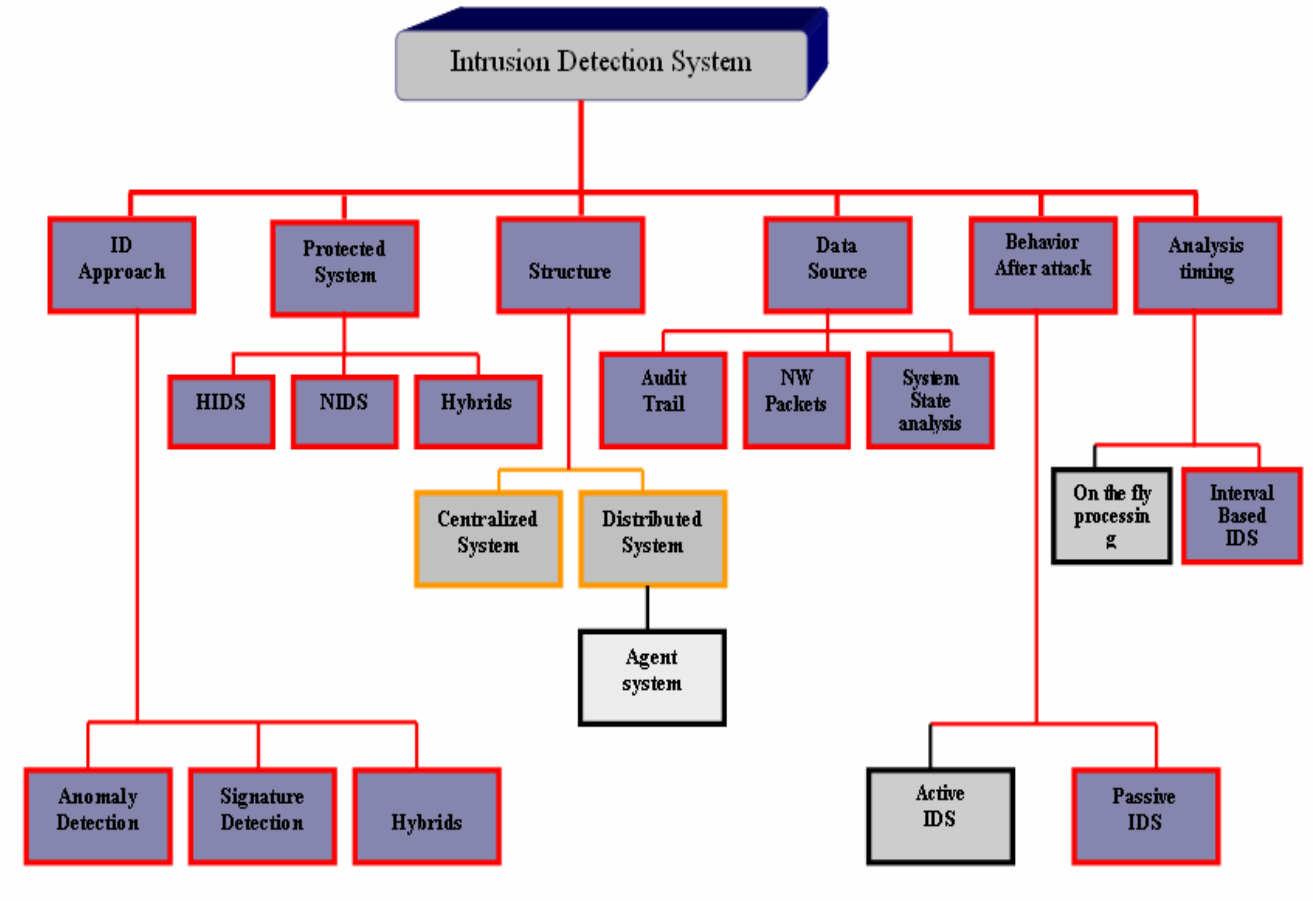


Figure (2.1) the classification of IDS from six different points of views

2.2.2 IDS: Data Processing Techniques

Depending on the type of approach taken in intrusion detection, various processing mechanisms (techniques) are employed for data to reach IDS. Several systems are described briefly below [Kaz04]:

- **Expert systems**, these work on a previously defined set of rules describing an attack. All security related events incorporated in an audit trail are translated in terms of if-then-else rules.
- **Signature analysis**, similarly to expert system approach, this method is based on the attack knowledge. They transform the semantic description of an attack into the appropriate audit trial format. Thus, attack signatures can be found in logs or input data streams in a straightforward way. An attack scenario can be described, for example, as a sequence of audit events that a given attack generates or patterns of searchable data that are captured in the audit trail. This method uses abstract equivalents of audit trail data. Detection is accomplished by using common text string matching mechanisms.
- **Colored Petri Nets**, the Colored Petri Nets approach is often used to generalize attacks from expert knowledge bases and to represent attacks graphically. Purdue University's IDIOT system uses Colored Petri Nets. With this technique, it is easy for system administrators to add new signatures to the system. However, matching a complex signature to the audit trail data may be time-consuming.
- **State-transition analysis,** here an attack is described with a set of goals and transitions that must be achieved by an intruder to compromise a system. Transitions are represented on state-transition diagrams.
- **Statistical analysis approach**, this is a frequently used method. The user or system behavior (set of attributes) is measured by a number of variables over time. Examples of such variables are: user login, logout, number of files accessed in a period of time, usage of disk space, memory, CPU etc. The frequency of updating can vary from a few minutes to, for example, one month. The system stores means values for each variable used for detecting exceeds that of a predefined threshold. Yet, this simple approach was unable to match a typical user behavior model. Approaches that relied on matching individual user profiles with aggregated group

variables also failed to be efficient. Therefore, a more sophisticated model of user behavior has been developed using short and long-term user profile. These profiles are regularly updated to keep up with the changes in user behaviors. Statistical methods are often used in implementations of normal user behavior profile-based Intrusion Detection System.

- **Neural Networks**, Neural networks use their learning algorithm to learn about the relationship between input and output vectors and to generalize them to extract new input/output relationships. With the neural network approach to intrusion detection, the main purpose is to learn the behavior of actors in the system (e.g., users, daemons). It is known that statistical methods partially equate neural networks (as will be discussed later in this chapter).
- **User intention identification**, this technique models normal behavior of users by the set of high-level tasks they have to perform on the system (in relation to the users' function). These tasks are taken as series of actions, which in turn are matches to the appropriate audit data. The analyzer keeps a set of tasks that are acceptable for each users. Whenever a mismatch is encountered, an alarm is produced.
- **Machine learning**, this is an artificial intelligence technique that stores the user-input stream of commands in a vectorial form and is used as a reference of normal user behavior profile. Profiles are then grouped in a library of user commands having certain common characteristics.
- **Computer immunology**, Analogies with immunology has lead to the development of a technique that constructs better models.
- **Data mining** generally refers to a set technique that use the process of extracting previously unknown but potentially useful data from large stores of data. Data mining method excels at processing large system logs (audit data). However they are less useful for stream analysis of network traffic.

2.3 Artificial Neural Network

The idea of Artificial Neural Network (ANN) came from the idea of working human brain, the first step toward artificial neural networks came in 1943 when Warren

McCulloch, a neurophysiologist, and a young mathematician, Walter Pitts, wrote a paper on how neurons might work. Think scientists in a way which can simulate the process, which occur in the human mind, and came to the knowledge of Neural Network, which falls under science artificial intelligence, so as to make computers intelligent devices, they can gain knowledge of the same way that acquires the rights of knowledge, they control the way weights during the learning. In addition, on the structural side large number of highly interconnected processing elements (*neurons*) working together. The neuron is the basic information processing unit of a Neural Network (NN); it consists of: A set of links, describing the neuron inputs, with weights *W1, W2, …,Wm,* An adder function (linear combiner) for computing the weighted sum of the inputs (real numbers) [Zur96]:

$$U_j = \sum_{i=1}^{p} W_{ji}\, x_i \qquad (2.1)$$

And an activation function (squashing function) for limiting the amplitude of the neuron output.

Tan-Sigmoid function = 2/ (1+exp (-2×n))-1 (2.2)

In addition, there is extra weight value considered which is corresponding to the constant bias (extra input). The bias is an external parameter of the neuron; it can be modeled by adding an extra input. Figure (2.2) shows the neuron and bias, while Figure (2.3) shows the biological neuron:

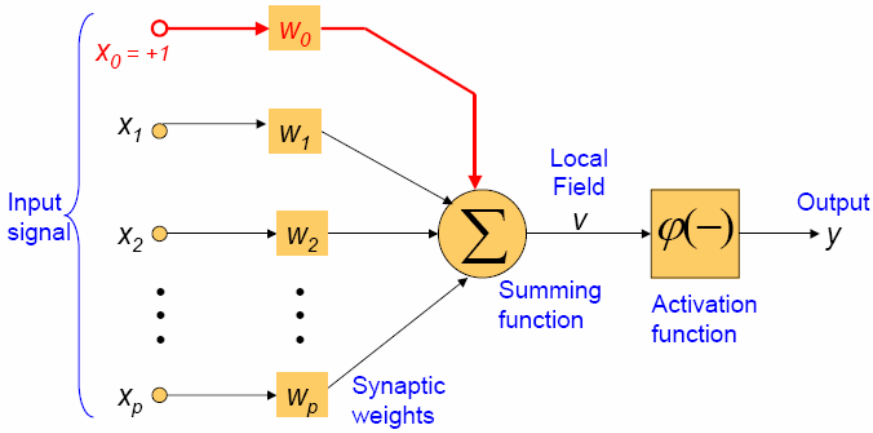


Figure (2.2) the artificial neuron [Mth07].

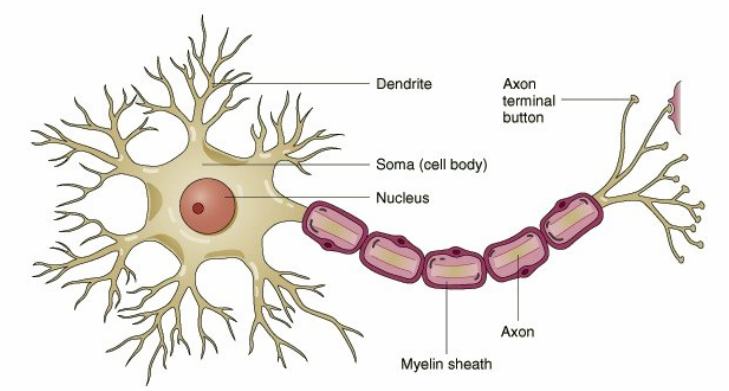


Figure (2.3) biological neuron

2.3.1 Classification of Neural Network

Neural networks can be classified into dynamic and static categories. Static (Feed-forward) networks have no feedback elements and contain no delays; the output is calculated directly from the input through feed-forward connections. The training of static networks will be discussed in Backpropagation. In dynamic networks, the output depends not only on the current input to the network, but also on the current or previous inputs, outputs, or states of the network.

Dynamic networks can also be divided into two categories: those that have only feed-forward connections, and those that have feedback, or recurrent, connections.

Generally classified neural networks on the basis of either training (learning) or architectures. There are two approaches to training-supervised and unsupervised. Supervised training involves a mechanism of providing the network with the desired output either by manually "distinguish" the network's performance or by providing the desired outputs with the inputs. Unsupervised training is where the network has to make sense of the inputs without outside help. From architecture point of view, three main classes of network architectures will be illustrated [Fau94]:

– Single-layer Perceptron (SLP).

– Multi-layer Perceptron (MLP).

– Recurrent (Feedback).

2.3.1.1 Supervised Training

In supervised training, both the inputs and the outputs are provided. Then the weights are adjusted according to the used leaning algorithm depending on the propagating differences between resulting outputs against the desired outputs back through the net. This process occurs over and over as the weights are continuously adjusted. The set of data which enables the training is called the "training set". During the network training phase the same set of data is processed many times as the connection weights are ever refined. Many problems are resolved such as classification, recognition, diagnostic and regression. Different supervised model exist such as perceptron, Adaline, feed-forward and radial basis function [Fau94].

2.3.1.2 Unsupervised Training

In unsupervised training, the network is provided with input vector but no target vectors are specified (i.e. without the need for desired output vector). The net adjust the weights so that the most similar input vector are assigned to the same output for cluster (i.e. the system itself must then decide what features it will use to group the input data). This is often referred to as self-organization or adaptation. The problems are resolved such as clustering and content addressable memory. The most popular models of unsupervised NN is self organizing neural network.

2.3.1.3 Single-layer Perceptron (SLP)

This reflects one of the oldest structures in neural networks, which consists of one layer of connection weights. The neurons are input neurons, which receive singles from the outside world, and output units, from which the response of the net can be read. On the basis that the computational input layer does not undertake any operations of arithmetic. The input layer is fully connected to the output layer. Figure (2.4) shows the single- layer networks[Fau94]:

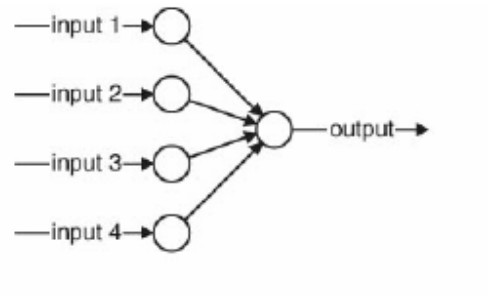


Figure (2.4) a single-layer linear model [Fau94].

2.3.1.4 Multi-layer Perceptron (MLP)

In MLP, there is input layer and output layer, in addition to many hidden layers. If the lines of communication between cells of the input layers connected with hidden layers and then with the output layer, this structure is called Feed-forward (feed-forward Architecture). Figure (2.5) shows the multi–layer networks [Fau94]:

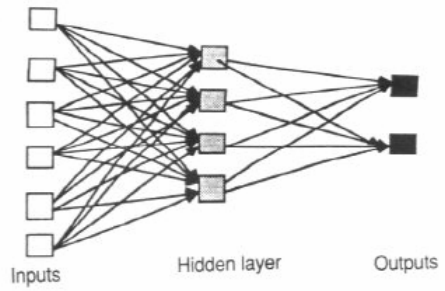


Figure (2.5) a multi - layer liner model [Fau94].

2.3.1.5 Recurrent (Feedback)

Recurrent networks: can be unstable, or oscillate, or exhibit chaotic behavior e.g., given some input values, can take a long time to compute stable output and learning is made more difficult. However, can implement more complex agent designs and can model systems with state the Figure (2.6) below shows recurrent networks [Fau94]:

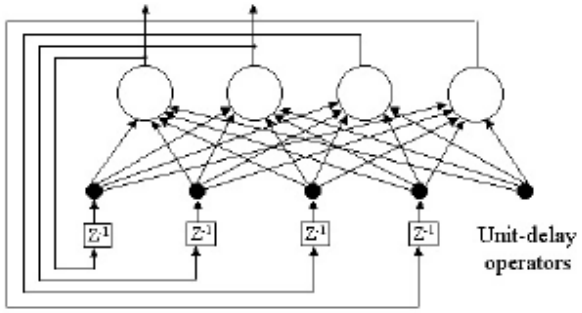


Figure (2.6) a recurrent network with hidden neurons [Fau94].

2.3.2 Major Components of an Artificial Neuron

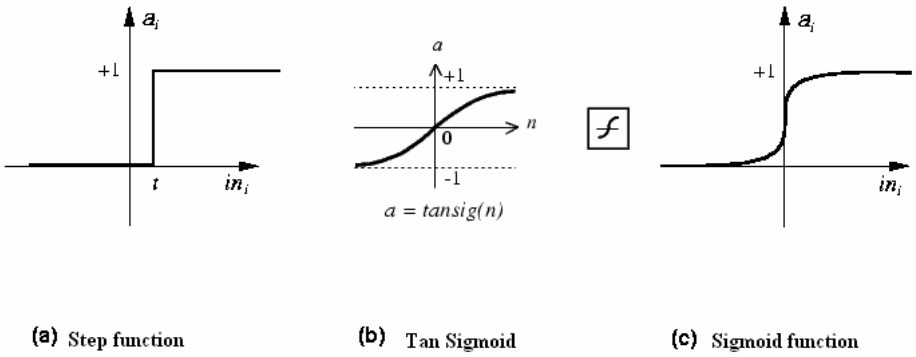
This section describes the seven major components which make up an artificial neuron. These components are valid whether the neuron is used for input, output, or is in one of the hidden layers [Zur96]:

a) **Weighting Factors**: A neuron usually receives many simultaneous inputs. Each input has its own relative weight which gives the input the impact that it needs on the processing element's summation function. These weights perform the same type of function as do the varying synaptic strengths of biological neurons. In both cases, some inputs are made more important than others so that they have a greater effect on the processing element as they combine to produce a neural response.

b) **Summation Function:** The first step in a processing element's operation is to compute the weighted sum of all of the inputs. Mathematically, the inputs and the corresponding weights are vectors which can be represented as (i_1, i_2 . . . i_n) and (w_1, w_2 . . . w_n). The total input signal is the dot (inner) product of these two vectors.

c) **Transfer Function:** The result of the summation function, almost always the weighted sum, is transformed to a working output through an algorithmic process known as the transfer function. In the transfer function the summation total can be compared with some threshold to determine the neural output. If the sum is greater than the threshold value, the processing element generates a signal. If the sum of the input and weight products is less than the threshold, no signal (or some inhibitory signal) is generated. Both types of responses are significant. In addition, the threshold, or transfer function is generally non-linear. Linear (straight-line) functions are limited because the output is simply proportional

to the input. Linear functions are not very useful. The Figure (2.7) below shows the activation function:



(a) Step function (b) Tan Signoid (c) Signoid function

Figure (2.7) sample transfer functions [Mth07].

d) **Scaling and Limiting:** After the processing element's transfer function, the result can pass through additional processes which scale and limit. This scaling simply multiplies a scale factor times the transfer value, and then adds an offset.

e) **Output Function (Competition):** Each processing element is allowed one output signal which may output to hundreds of other neurons. This is just like the biological neuron, where there are many inputs and only one output action. Normally, the output is directly equivalent to the transfer function's result.

f) **Error Function and Back-Propagated Value:** In most learning networks the difference between the current output and the desired output is calculated. This raw error is then transformed by the error function to match particular network architecture.

g) **Learning Function:** The purpose of the learning function is to modify the variable connection weights on the inputs of each processing element according to some neural based algorithm. This process of changing the weights of the input connections to achieve some desired result can also be called the adaptation function, as well as the learning mode.

2.3.3 Feed-forward Neural Network: Backpropagation (BP):

Most popular training method for neural networks, the generalized delta rule [Rum86], also known as Backpropagation algorithm. The explanation intended to give an outline of the process involved in Backpropagation algorithm. The (NN) explained here contains three layers. These are ***input***, ***hidden***, and ***output*** layer. During the training phase, the training

data is fed into to the input layer. The data is propagated to the hidden layer and then to the output layer. This is called the ***forward pass*** of the Backpropagation algorithm. In ***forward pass,*** each node in hidden layer gets input from all the nodes from input layer, which are multiplied with appropriate weights and then summed. The output of the hidden node is the nonlinear transformation of this resulting sum. Similarly each node in output layer gets input from all the nodes of the hidden layer, which are multiplied with appropriate weights and then summed. The output of this node is the non-linear transformation of the resulting sum. The output values of the output layer are compared with the ***target output values.*** The ***target output values*** are used to teach network. The error between actual output values and target output values is calculated and propagated back toward hidden layer. This is called the ***backward pass*** of the Backpropagation algorithm. The error is used to update the connection strengths between nodes, i.e. weight matrices between input-hidden layers and hidden-output layers are updated. During the testing phase, no learning takes place i.e., weight matrices are not changed. Each test vector is fed into the input layer. The feed-forward of the testing data is similar to the feed-forward of the training data. Backpropagation architecture was developed in the early 1970s by several independent sources (Werbor; Parker; Rumelhart, Hinton and Williams). There are many laws (algorithms) used to implement the adaptive feedback required to adjust the weights during training. The most common technique is backward-error propagation, more commonly known as back propagation. The Backpropagation algorithm searches for weight values that minimize the total error of the network over the set of training examples (training set). Backpropagation consists of the repeated application of the following two passes:

- Forward pass: in this step the network is activated on one example and the error of (each neuron of) the output layer is computed.
- Backward pass: in this step the network error is used for updating the weights (credit assignment problem).

Therefore, starting at the output layer, the error is propagated backwards through the network, layer by layer. This is done by recursively computing the local gradient of each neuron. Here, a simple example shows the work of Backpropagation algorithm, uses supervised training, if the output is not correct, the weight are adjusted according to the formula:

$$W_{new} = W_{old} + \alpha \text{ (desired} - \text{output)} \times \text{input.} \qquad (2.3)$$

α is the learning rate, (in this example assume α = 1).

Assume output threshold = 1.2.

Figure (2.8) shows the neural network:

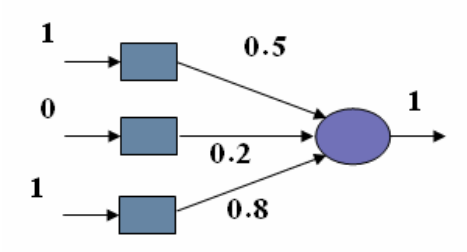


Figure (2.8) sample neural network

To apply the above:

$= 1 \times 0.5 + 0.2 \times 0 + 1 \times 0.8$
$= 1.3$

$1.3 < 1.2$

Then adjust the weight:

$W_{1new} = 0.5 + 1 \times (0 - 1) \times 1 = -0.5$

$W_{2new} = 0.2 + 1 \times (0 - 1) \times 0 = 0.2$

$W_{3new} = 0.8 + 1 \times (0 - 1) \times 1 = -0.2$

$= 1 \times (-0.5) + 0.2 \times 0 + 1 \times (-0.2)$

$= (-0.5) + (-0.2)$

$= -0.7$ ⟶ $-0.7 < 1.2$

Threshold, this is the edge and identify the value in the neural network.

2.4 Crisp set and Fuzzy set

Fuzzy logic (FL) was introduced by Dr. Lotfi Zadeh in the 1960 as a means to model the uncertaints of natural language [Zad65]. There are many motivations that encouraged scientists to develop the science of fuzzy logic or crisp logic, with the development of computer and software have the desire to reinvent or programmable systems can deal with inaccurate information along the lines of human, but this problem

was born as the computer can not deal only with specific and accurate information, and has resulted in this trend known as expert systems or artificial intelligence. Knowledge of fuzzy logic is one of the theories through which they could build such systems. General, fuzzy mainly concerned with fuzzy logic and fuzzy set. Fuzzy Logic (FL) is a multi valued logic, that allows middle values to be defined between conventional evaluations like true/false, yes/no, high/low, etc, this concept is sufficient for many areas of applications, but it can easily be seen, that it lacks in flexibility for some applications like classification of remotely sensed data analysis, with the fuzzy logic use the rule. In addition, the main drawbacks of crisp set are that the membership function of crisp logic fails to distinguish between members of the same set. Figure (2.9) below shows the process:

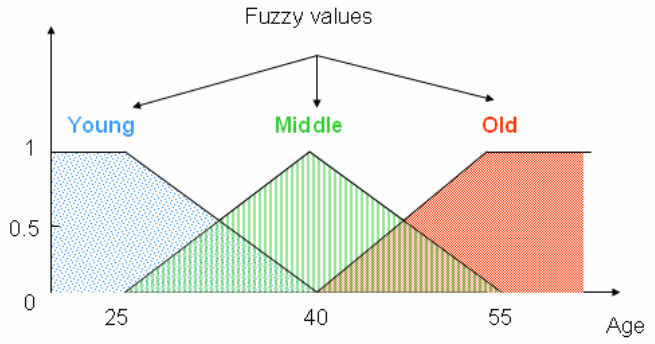


Figure (2.9) the fuzzy logic process [Zad65].

We can note from Figure (2.10) the meaning of the Fuzzification scales and maps input variables to fuzzy sets, but the Defuzzification convert fuzzy output values to control signals. While in fuzzy sets an object can partially be in a set, the membership degree takes values between 0 and 1, 1 mean entirely in the set, 0 means entirely not in the set, other values mean partially in the set. The fuzzy sets are in the range [0.0, 1.0], with 0.0 representing absolute falseness and 1.0 representing absolute truth. This does not deal with rule but deal with the membership degree. We will deal with only fuzzy set, we will fuzzy the data only. Fuzzy set play an important role in recognizing dangerous events and reducing false alarms level [Yao02]. Interest from the use the fuzzy set if the dataset are huge and complex (overlapping), the fuzzification resolve this overlap. The fuzzy set deals with a linguistic variable associate's words or sentences with a measure of belief functions, also called membership function (use the natural language). The set of values that it can

take is called term set, the figure (2.9) shows simple example of membership relation of age.

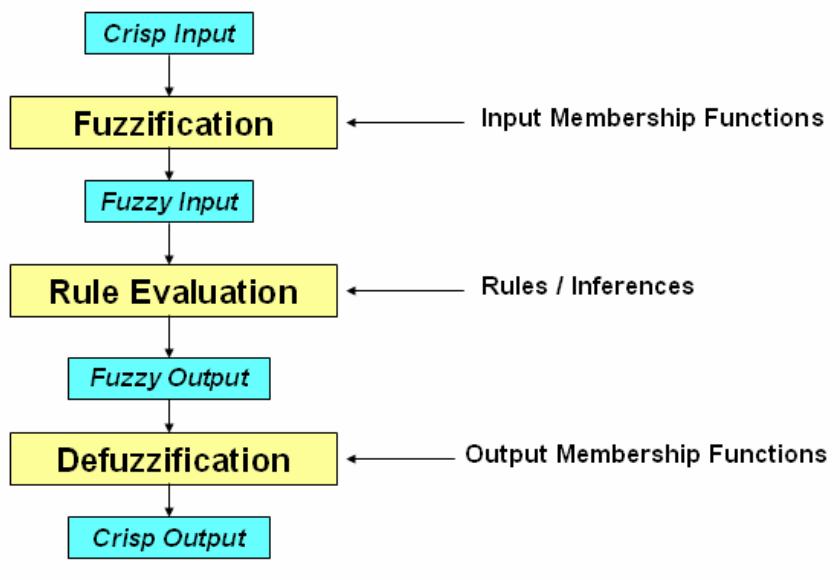


Figure (2.10) membership function

Through example, we can note the degree of affiliation of each and every one. Each value in the set is a fuzzy variable defined over a base variable.

Chapter Three
Design and Implementation

3.1 Introduction

The aim of this work is to implement two neural network architectures (suggested for intrusion detection purposes) and evaluate their applicability and performance. In order to evaluate the performance of the suggested ID approach, a standard set of data to be audited, which includes a wide variety of intrusions simulated in a network environment is needed. It was found that this sort of data was prepared and managed by Massachusetts Institute of Technology's (MIT) Lincoln Laboratory.

This chapter starts with a description of the dataset used for training the intrusion detection and the preprocessing steps needed before training the neural nets, followed by the architecture description of the two networks and steps used for the *training phase*. The chapter also explores the feature-sets and the chosen subset most appropriate for intrusion detection

3.2 Data Source

In this work, an ANN was designed to be run over the 1999 Knowledge Discovery in Database KDD Cup data, provided by the Defense Advanced Research Projects Agency DARPA and the MIT Lincoln Labs. The data set was generated via a simulated U.S. Air Force local-area network set up at Lincoln Labs, which was run and operated similarly to a standard Air Force network, excepting for planned and recorded attacks.

Originally, the data consisted of nine weeks of raw Transmission Control Protocol (TCP) dump data from the network. The dataset contains normal and attacks, has five different classes. We performed a five-class binary classification. The Normal data belongs to class (5), and 39 attack types that could be classified into four main categories , Probe belongs to class (1), DOS belongs to class (2), U2R belongs to class (3), R2L belongs to class (4).

3.3 Dataset

In this work, the used dataset is extracted from KDD dataset; it contains about 311029 connection records. Each connections record has 41 different, 22 of these features describe the connection itself and 19 of them describe the properties of connections to the same host in last two seconds. The different features of the corresponding connection, and the value of the connection is labeled either as an attack with one specific attack type, or as normal.

The main task for the developed classifier (using ANN), which trained and tested using KDD Cup 1999, that act as a predictive model able to distinguish between legitimate (normal) and illegitimate (called intrusion or attack) connection in a computer network. The ANN was run over a percent subset of the data, called the training data, and then tested over the entire data set to test real-world performances. The task was to predict the value of each connection (normal or one of the above attack categories) for each of the connection record of the test dataset that containing 311029.

3.4 Feature Set

Stolof [Sto00] defined higher-level features that help in distinguishing normal connection from attacks. There are 39 different attack types present in the datasets falls exactly into one of the following four categories Table (3.1):

1. Probing: surveillance and other probing.
2. DOS: denial-of-services.
3. U2R: unauthorized access to local super user (root) privileges.
4. R2L: unauthorized access from a remote machine.

Table (3.1) summarized each the 39 attacks and their categories [Bou04].

Probing	DOS	U2R	R2L
ipsweep, mscan, nmap, portsweep, saint, satan.	apache2, back, land, mailbomb, neptune, pod, processtable, smurf, teardrop, udpstorm.	buffer_overflow httptunnel, loadmodule perl, ps, rootkit, sqlattack, xterm.	ftp_write, guess_passwd, imap, multihop, named, phf, send-mail, snmpgetattack, snmpguess, spy, warezclient, warezmaster, worm, xlock, xsnoop.

There are several categories of derived features:

- The "same host" features examine only the connections in the past seconds that have the same destination host as the current connection, and calculate statistics related to protocol behavior, services, etc.
- The similar "same services" features examine only the connections in the past two seconds that have the same services as the current connection.
- "Same host" and "same service" features are together called time-based traffic features of the connection records.
- Some probing attacks scan the hosts (or ports) using a much larger time interval than two seconds, for example once per minute. Therefore, connection records were also sorted by destination host, and features were constructed using a window of 100 connections to the same host instead of a time window.
- Unlike most of the DOS and probing attacks, there appear to be no sequential patterns that are frequent in records of R2L and U2R attacks. This is because the DOS and probing attacks involve many connections to some hosts(s) in a very short period of time, but the R2L and U2R attacks are embedded in the data portions of packets, and normally involve only a single connection.

A complete listing of the set of features defined for the connection records is given in table (3.2) below:

Table (3.2) features of KDD Cup 1999

	Feature name	Description	Type
1	**Duration**	**Length (number of seconds) of the connection**	**Continuous**
2	**Protocol-type**	**Type of the protocol e.g. tcp, udp, etc.**	**Discrete**
3	**Service**	**Network service on the destination e.g., http, telnet, etc.**	**Discrete**
4	**Flag**	**Normal or error status of the connection**	**Discrete**
5	**Src_bytes**	**Number of data bytes from source to destination**	**Continuous**

6	Dst_bytes	Number of data bytes from destination to source	Continuous
7	Land	1 if connection is from/to the same host/port; 0 otherwise	Discrete
8	Wrong fragment	Number of “wrong” fragments	Continuous
9	Urgent	Number of urgent packets	Continuous
10	Hot	Number of “hot” indicators	Continuous
11	Num_failed_logins	Number of failed login attempts	Continuous
12	Logged_in	1 if successfully logged in ; 0 otherwise	Discrete
13	Num_compromised	Number of “compromised” conditions	Continuous
14	Root_shell	1 if root shell is obtained; 0 otherwise	Discrete
15	Su_attempted	1 if “su root” command attempted; 0 otherwise	Discrete
16	Num_root	Number of “root” accesses	Continuous
17	Num_file_creation	Number of file creation operation	Continuous
18	Num_shells	Number of shell prompts	Continuous
19	Num_access_files	Number of operations on access control files	Continuous
20	Num_outbound_cmds	Number of outbound commands in an ftp session	Continuous
21	Is_hot_login	1 if the login belongs to the “host” list; 0 otherwise	Discrete

22	Is_guest_login	1 if the login is a “guest” login;0 otherwise	Discrete
23	Count	Number of connections to the same host as the current connection in the past two seconds	Continuous
		Note: the following feature refer to these same_host connections.	
24	Srv_count	Number of connections to the same service as the current connection in the past two second	Continuous
25	Serror_rate	% of connection that have “SYN “ error	Continuous
26	Srv_serror_rate	% of connections that have “SYN” error	Continuous
27	Rerror_rate	%of the connections that have “REJ” errors	Continuous
28	Srv_rerror_rate	% of connection that have “REJ” errors	Continuous
		Note: the following feature refer to these same_service connections.	
29	Same_srv_rate	% of connections to the same service	Continuous
30	Diff_srv_rate	% of the connections to different service	Continuous

31	Srv_diff_host_rate	% of connection to different hosts	Continuous
32	Dst_host_count	Number of connection to the same host	Continuous
33	Dst_host_srv_count	Number of connection to the same service for the same host destination	Continuous
34	Dst_host_same_srv_rate	% of the connections of same services to the same host destination	Continuous
35	Dst_host_diff_srv_rate	% of connections of different services to the host destinations	Continuous
36	Dst_host_same_scr_port_rate	% of connection from same source port number to host destination	Continuous
37	Dst_host_srv_diff_host_rate	% connection of different service to host destination	Continuous
38	Dst_host_serror_rate	% connection that have “SYN” error to the host destination	Continuous
39	Dst_host_srv_serror_rate	% connection that have “SYN” service to the host destination	Continuous
40	Dst_host_rerror_rate	% connection that have “REJ” error to the host destination	Continuous
41	Dst_host_srv_rerror_rate	% connection that have “REJ” service to the host destination	Continuous

Table (3.2) above describes the 41 features of each connection record in the DARPA KDD cup 1999. The fields with blue color are features that have been considered in this research based on previous studies suggested by Chebrolu [Che05].

3.5 Pre-Processing Data

The original data set used in this work includes symbolic attributes, such as "protocol type", with values "UDP", "TCP", "ICMP" , feature "services", with values " PRIVATE", "ECR_I", "HTTP", ect……, and other feature "Flag", with values "SF", "REJ", "RSTR", "SO". At the beginning, these features must be converted to the numerical form before any operation on the dataset (encoding), such as UDP = 0, TCP =1, icmp =2, and private =0, ecr_i =1, http =2, ect……, Sf =0, Rej =1, Rstr =2, So=3, in addition to plus class label, such as snmpgetattack=0, Xlock=1, Normal=39, ect,…this numerical representation is necessary because the features vector fed to the input of the neural network has to be numerical.

Two preprocessing operations applied to the original dataset: ***uniform distribution*** of pattern class and ***normalization of numerical attributes***. ***Uniform distribution*** in this process divided the dataset into (10) sets each set contain (30,000) records and final dataset contains (11029) records. The selection is implemented using random generator of math-lab (called Random permutation) which performs random generation with uniform distributed ability at which a number is generated randomly, each number represents the number of the record, on other hand, creates a dataset from the original dataset with the following property: if the samples number of pattern k is m and the original dataset has n samples, then the probability to find a sample of class k in the first n/m samples of the final data set is 1.0. Therefore, each portion of the final data set has almost the same distribution of the full data set [Gom02].

In the ***normalization,*** each numerical value in the data set is normalized between 0.0 and 1.0 according to the following equation [Gom02]:

$$X = \frac{x - Min}{Max - Min} \qquad (3.1)$$

where, x is the numerical value at which $0 \leq x \leq 1$, Min is the minimum value for the attribute that x belongs to, and Max is the maximum value for the attribute that x belong to

[Gom02]. Furthermore, assign value to each record in the dataset which represents the target type of the record, such as normal [1 1 1 1 1], Probe [-1 1 -1 -1 -1], DOS [-1 -1 1 -1 -1], U2R [-1 -1 -1 1 -1] and R2L [-1 -1 -1 -1 1] after that the dataset is fed to the ANN. In the ANN at first the available data is divided into three subsets. The first subset is the train set (0.50), the second subset is the validation set (0.25), and the third subset is the test set (0.25).

The main purpose from taking some dataset validation in the training part, to handle the over-fitting problem that may occur during neural network training is over-fitting. When the ANN begins to over-fit the data, the error on the validation set will typically begin to rise. When the validation error increases for a specified number **of iterations**, the training is stopped. Another process in the train part decode to the 39 attack to the four major class such as, Probe =1, DOS =2, U2R =3 and R2L =4, in addition to the Normal =5. The figure (3.1) below shows the flow chart of the pre-processing:

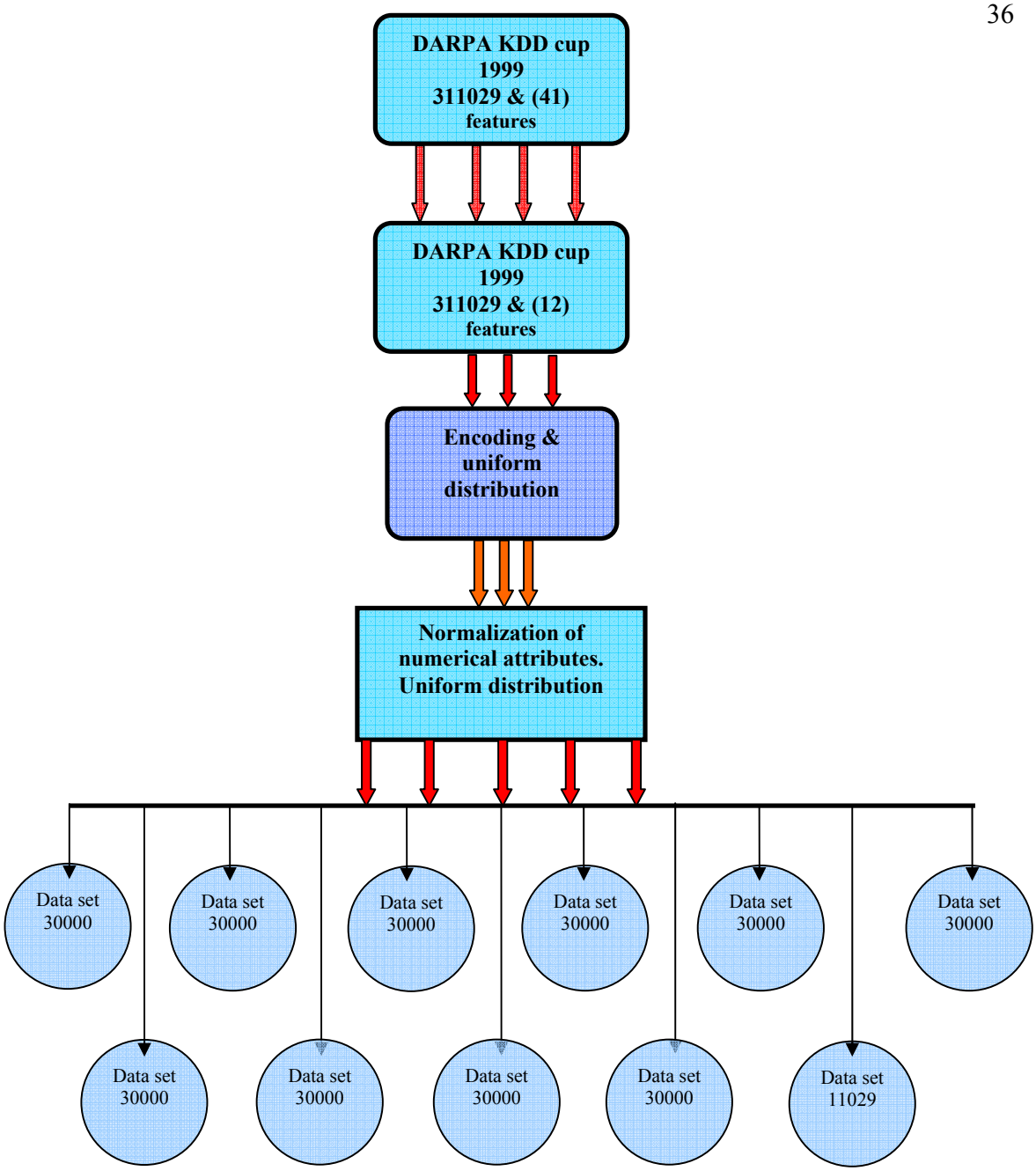


Figure (3.1) the flow chart of the Pre-processing of datasets.

3.6 Fuzzy Membership

The ID problem is viewed in its models (anomaly, and the misuse or signature) as a two-class classification problem: the goal is to classify patterns of the system behavior in two categories (normal and attack) using patterns of known attacks that belongs to the

abnormal class, and patterns of the normal behavior. With fuzzy set, the solution of this classification problem is based on neural networks trained with fuzzified data.

In fuzzy sets, a need to define the linguistic notions and membership functions define the truth-value of such linguistic expression. Table (3.3) shows the difference between crisp sets and fuzzy sets [Gom02].

Table (3.3): Comparison between fuzzy sets and crisp sets

Fuzzy sets	Crisp sets
In fuzzy sets an object can partially be in a set.	**In crisp sets an object is entirely in a set or is not.**
The member ship degree takes values between 0 and 1.	**The member ship degree takes only two values 0 or 1.**
1 means entirely in the set, 0 means entirely not in the set, other values mean partially in the set.	**1 means entirely in the set, 0 means entirely not in the set. Other values are not allowed.**

The membership degree for a fuzzy set of an object defines a function where the universe of discourse (set of values that the object can take) is domain, and the interval [0,1] is the range. That function is called the membership function. Figure (3.2) shows the used membership function; the ***triangular membership*** function (used in this work)[Gom02]:

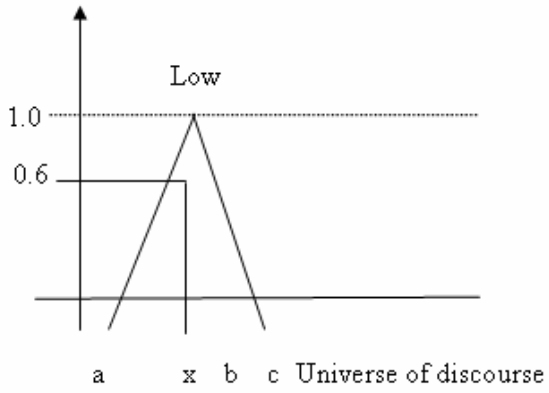


Figure (3.2) triangular membership function for a fuzzy set

In Figure (3.2), the object x has 0.6 degree of member ship to the fuzzy set low, i.e., x does not entirely belong to the set low, but x belong to the fuzzy set low and does not belong to the set at the same time. The equation of the *triangular membership* used in this work is illustrated below [Suz07Gom02]:

$$f(x,a,b,c) = \begin{cases} 0, & x \le a \\ \dfrac{x-a}{b-a}, & a \le x \le b \\ \dfrac{c-x}{c-b}, & b \le x \le c \\ 0, & c \le x \end{cases} \qquad (3.2)$$

For example, assume that a, b, c of each set of the five sets (L, ML, M, MH, H) are:

L: a = 0, b = 0.166, c = 0.333

ML: a = 0.166 b = 0.333 c = 0.5

M: a = 0.333 b = 0.5 c = 0.666

MH: a = 0.5 b = 0.666 c = 0.833

H: a = 0.666 b = 0.833 c = 1

Let x =0,2 using equation (3.2), its membership degree for:

L = 0.7964

ML = 0.2036

M = 0

MH = 0

H = 0

A collection of fuzzy sets, called fuzzy space, define the fuzzy linguistic values or fuzzy classes that an object can belong to. The used fuzzy space for each numerical attribute assigned as illustrated in Figure (3.3).

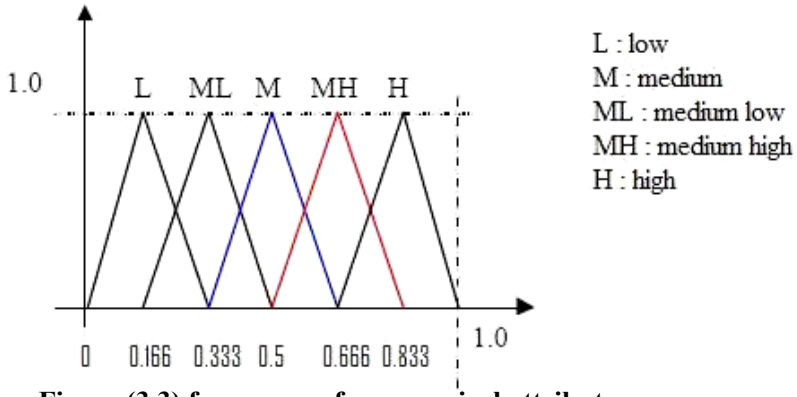


Figure (3.3) fuzzy space for numerical attributes in the KDD-cup 99 data set

For non-numerical attributes like logged-in, the categorical values is used as crisp sets (fuzzy sets that does not overlap each other). Figure (3.4) shows an example of the fuzzy space associated to the logged-in attribute. Therefore, a value of false for this attribute has degree of membership to the crisp set FALSE equal 1.0 and degree of membership to the fuzzy set TRUE equal to Zero [Gom02]:

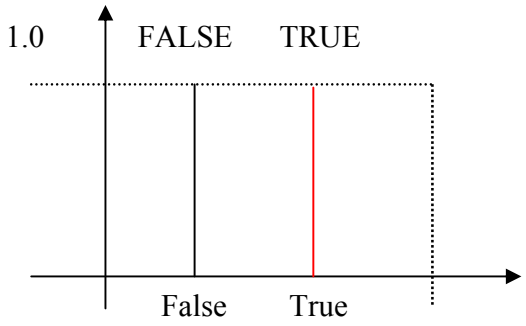


Figure (3.4) fuzzy space for the non-numerical attribute logged-in

With fuzzy spaces, fuzzy logic allows an object to belong to different classes at the same time. This concept is helpful when the difference between classes is not well defined. It is the case in the ID task, where the difference between the normal and abnormal class are not well defined [Gom02] leading to increase false alarms. This was one of the main motivations of the work of fuzzification data, is to reduce false alarms.

3.6.1 Type of Alarms

The main work of intrusion detection system (IDS) is the discovery of which connection record passes through the network, and after, issuing alarm. There are four types of alarms as illustrated in Table (3.4):

Table (3.4) alarm type

True positive	**If it is normal and detection system is normal.**
True negative	**If it is attack and detection system is attack.**
False negative	**If it is attack and detection system is normal.**
False positive	**If it is normal and detection system is attack.**

Among these alarms, the concentration is on false negative alarm, which is the most dangerous one since it classifies the attack as normal.

3.7 Architecture of the Artificial Neural Networks

This section concerned with the explanation of the structural Artificial Neural Networks (ANN) used in this work, types and components have been clarified in the previous chapter. In fact, in this work, feed-forward neural network is used with two different architectures, one trained with nonfuzzified dataset and other for the fuzzified data. The learning algorithm is learngdm (Matlab, 2007). Learngdm calculates the weight change dW for a given neuron from the neuron's input P and error E, the weight (or bias) W, learning rate LR, and momentum constant MC, according to gradient descent with momentum:

$$dW = mc \times d \times W_{old} + (1\text{-}mc) \times lr \times gW \qquad (3.3)$$

3.7.1 Artificial Neural Networks used with Non-Fuzzified data

In general we know that the feed-forward neural networks consists of three layers, these are input, hidden, and output layers. In this work, the feed-forward neural networks (Backpropagation) are used. Input layer consists of ***twelve*** neurons (input) as equal features

that have been selected from KDD dataset based on previous studies [Che05]. The process of determining the number of hidden layers of each network is by using the concept (Trail and Error). Is to identify a certain number of layers and number of neurons per layer, therefore, different nets are trained and examining the network performance. Then choose the best network architecture. In this work, the hidden layer consists of 20 neurons. In addition to output layer consists of ***five*** neurons, this will be four of the intrusion (Probing, Dos, U2R, and R2L) and is one of the outputs is normal. In addition to the parameters used in this feed-forward Neural Network (NN) such as:

- TrainParam.epochs = 500; Epoch number (***Batch***, Sequential) – improve the result and condition stop.
- TrainParam.lr = 0.000001; learn rate, value to update weight – improve the result.
- TrainParam.goal = 0000; condition stop.
- TrainParam.min_grad=0.0000000001; value change in min_grad–improve the result.
- Threshold if the value is zero or less than zero is equal (-1), otherwise i.e. if the value more than zero is equal (1).

This is the type of neural network discussed briefly in previous chapter: the units each perform a biased weighted sum of their inputs and pass this activation level through a transfer function to produce their output, and the units are arranged in a layered feed-forward topology. An activation function used to transform the activation level of a unit (neuron) into an output signal. The Backpropagation algorithm uses an activation function. In this work, Tan-Sigmoid transfer function (tansig) is used, like the logistic function, except that output lies in the range (-1, +1). Often performs better than the logistic function because of its symmetry. Ideal for customization of multilayer perceptrons, particularly the hidden layers, Figure (3.5) shows the Tan-Sigmoid function. Figure (3.6) show the structure feed-forward neural network of that.

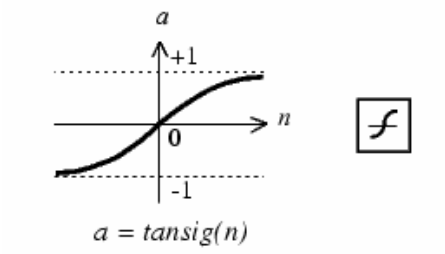


Figure (3.5) Tan-sigmoid transfer function [Mth07].

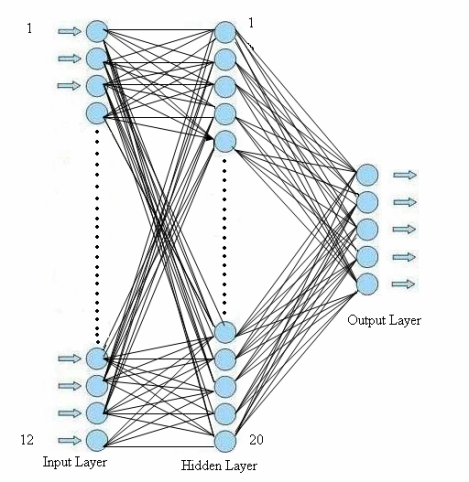


Figure (3.6) structure the neural network with normal data

3.7.2 Artificial Neural Networks Used with Fuzzified Data

In this type of feed-forward neural networks with fuzzified data, the input layer consists from ***sixty*** neurons, because of the membership, each value in the previous network will be represented with five values, the hidden layer consists of ***seven*** neurons. In addition to output layer consists of ***five*** neurons. In addition, the network will use the previous parameters and use the Backpropagation algorithm with the same transfer function. Figure (3.7) below shows the structure of the feed-forward neural network:

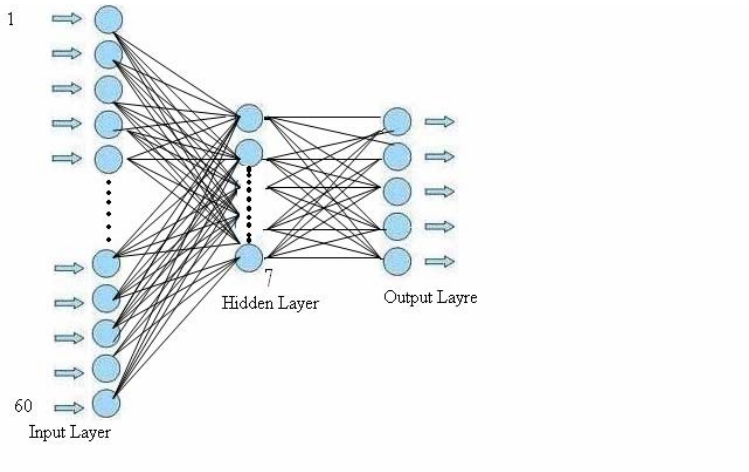


Figure (3.7) structure the neural network with fuzzification data

Chapter Four
Assessment Results

4.1 Introductions

The aim of this work is to develop and implement appropriate neural network architectures to work as intrusion detector using dataset is extracted from KDD dataset at which each connection record contains 41 attributed. Two architectures were suggested, based on dataset consists of 12 features selected from the 41 features (as illustrated in chapter three) which is suggested by Chebrolu [Che05]. The first neural net designed based on 12 inputs (non fuzzified attributes), while the other one is designed based on 60 input (fuzzified with 5 linguistics) to detect intrusions. The two suggested architectures were tested to evaluate their applicability and performance in detecting intrusions and check their behavior if it is anomaly or misuse or hybrid. The classification ability of the two suggested neural networks are studied and compared.

Convergence speed comparison will be ignored since the important factor in pattern classification problems is classification accuracy. To evaluate the performance of the suggested neural network recognition system, the accuracy of the system result should be calculated as follows:

$$\frac{\textit{Number of correctly classified patterns}}{\textit{Total number of patterns}} \qquad (4.1)$$

Also the four alarms will be calculated as follows [Suz07, Cha07, and Ves07]:

Let

TP= # normal connection record that are classified as normal (True positive)

TN= # attack connection record that are classified as attack (True negative)

FP= # normal connection record that are classified as attack (False positive)

FN= # attack connection record that are classified as normal (false negative)

Then

$$\text{True\_Positive\_Rate (sensitivity)} = TP/(TP+FN) \qquad (4.2)$$

$$\text{True-Negative\_Rate (specificity)} = TN/(TN+FP) \qquad (4.3)$$

$$\text{False\_Negative\_Rate} = (1\text{-sensitivity}) = FN/(FN+TP) \qquad (4.4)$$

$$\text{False-Positive\_Rate= (1-specificity)} = FP/(FP+TN) \qquad (4.5)$$

This chapter will illustrate the used dataset and the attributes in each connection record, then explore the experimental results and comparison between various NN classifiers. Analysis and assessment of the results will be illustrated.

4.2 General Data Splitting Method.

In this work, the used dataset is extracted from KDD dataset; it contains about 311029 connection records, the whole data is divided into five parts:

- Normal data part which are (60593) records.
- Probing attack part which are (4166) records.
- DOS attack part which are (229853) records.
- U2R attack part which are (230) records.
- R2L attack part which are (16187) records.

After Pre-Processing of the dataset (illustrated in section 3.5), as mentioned in the previous chapter, the data divide into eleven subsets, six of them to be used for training purposes and the remaining for testing and then change subsets subsequently. Each subset will be divided into three subsets, the first subset is the ***validation*** set (to help in choosing the most suitable NN architecture), the second subset is the ***training*** set, and the third subset is ***testing*** set. During training process, it is very important to avoid one of the major problems that may occur during neural network training ***overfitting***. The behavior of a net with the three subsets illustrated in figure (4.1).

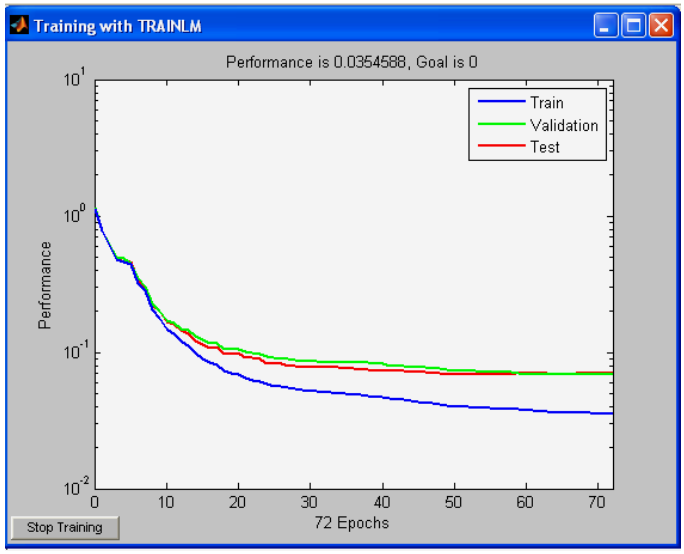


Figure (4.1) shows the divide subset

4.3 Improving Generalization.

One of the problems that occur during neural network training is called overfitting. The error on the training set is driven to a very small value, but when new data is presented to the network the error will be increased since the net could only recognize the dataset that belong to the training subset.

Figure (4.2) shows the response of a neural network that has been trained to approximate a noisy sine function. The underlying sine function is shown by the dotted line, the noisy measurements are given by the `+' symbols, and the neural network response is given by the solid line. Clearly this network has overfitted the data and will not generalize well [Mth07].

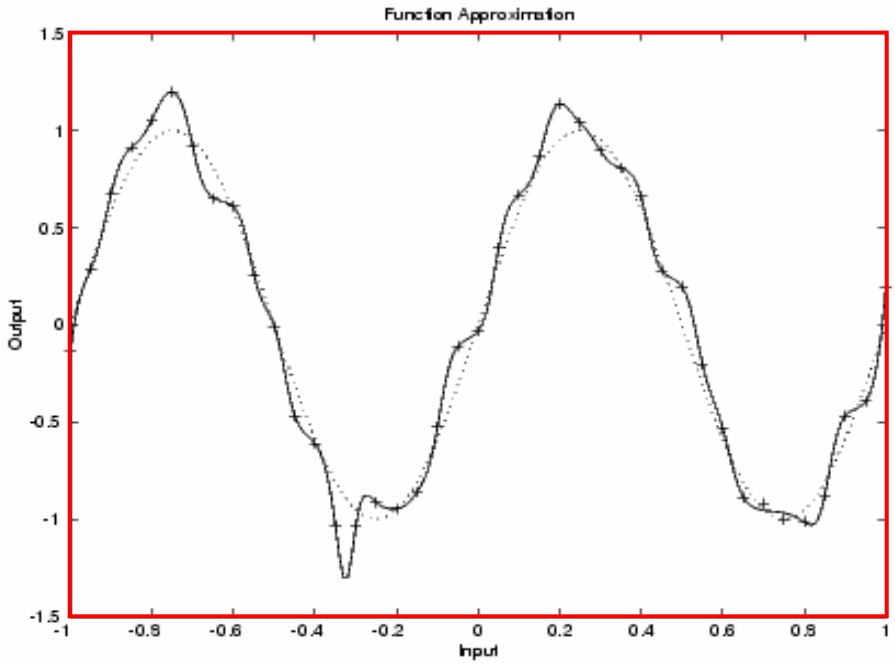


Figure (4.2) Generality

One method for improving network generalization is to use a network that is just large enough to provide an adequate fit. For larger network, the functions become more complex. If you use a small enough network, it will not have enough power to overfit the data [Mth07].

Unfortunately, it is difficult to know beforehand how large a network should be for a specific application. There are two other methods for improving generalization that are implemented in Neural Network: Regularization and Early stopping.

Note that if the number of parameters in the network is much smaller than the total number of points in the training set, then there is little or no chance of overfitting. If you can easily collect more data and increase the size of the training set, then there is no need to worry about the following techniques to prevent overfitting [Mth07].

4.4 Experimental Results: analysis and comparison

The experimental results analysis and comparison (for both developed Neural Network) are needed to answer the following questions:

1. Which NN architecture is the most suitable one for detecting intrusions through specifying the accuracy of each alarm type true negative, true positive, false positive, false negative. Choose the architecture with the highest classification accuracy and minimum false negative rate since it is the most dangerous alarm.
2. Check if the developed intrusion detector is anomaly, misuse, or hybrid.
3. Analyze the behavior of each of the developed neural network; check its ability for classifying each type of attacks (not only distinguish between attacks and normal connection records).

4.4.1 Train and Test Neural Network with Non-Fuzzified Dataset

To choose the suitable network architecture, different network architecture were trained with the same data subset to choose the best Neural Network architecture (i.e. the NN with highest-accuracy), in addition to changing the parameters in order to choose the most suitable set of parameters for the suggested feed-forward Neural Network (NN), these parameters are:

– TrainParam.epochs = 500; Epoch number (Batch) – improve the result and condition stop.
– TrainParam.lr = 0.000001; learn rate, value to update weight – improve the result.
– TrainParam.goal = 0000; condition stop.
– TrainParam.min_grad = 0.0000000001: value change in min_grad – improve the result.

– Threshold if the value is zero or less than zero is equal (-1), otherwise i.e. if the value more than zero is equal (1).

Approximately, the training time takes 20 minutes.

4.4.1.1 Result the Train and Test Neural Network with Non-Fuzzified Dataset

As illustrated in chapter three, that neural networks consisting of input layer, the hidden layer and output layer, the input layer and output layer are fixed. Therefore, the input layer consists of ***twelve*** neurons with the actual data and output layer consists of ***five*** neurons, but the input layer of fuzzified dataset consists of ***sixteen*** neurons with the and output layer consists of ***five*** neurons. But the number of hidden layer and the number neurons are variable. The main point is usually to determine ***the number of hidden layers*** of each network using "Trail and Error" concept to identify a certain number of layers and number of neurons within layer, therefore different network architecture were trained and their performance are examined. The chosen subset contains (30000) records (i.e. contain the subset all different type from the attack in different degree. Figure (4.3) shows the size of the chosen subset records:

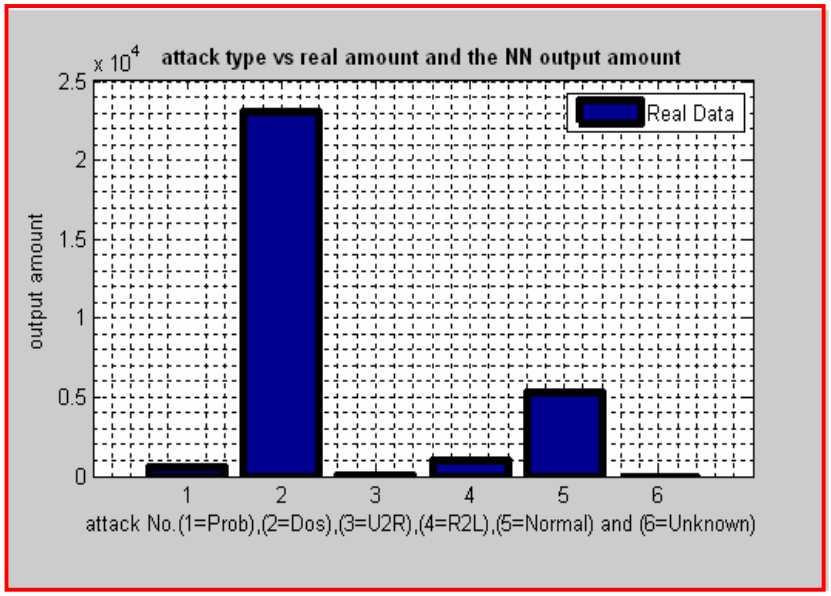


Figure (4.3) ratios of data that have been tested

'Normal' 5725, 'Probing' 358, 'DOS' 22366, 'U2R' 512, 'R2L' 1527, in fact the chosen subset contains the 30488 the increment resulted from increasing number of attack type

U2R since it is only small set with respect to other types to help in increasing accuracy. Among several tested neural nets, the following are the most promising networks:

- ***The first neural network architecture***: One hidden layer composed of ***twenty*** neurons in the hidden layer. The parameters used in this feed-forward Neural Network (NN) are:
  - TrainParam.epochs = 500; Epoch number (Batch) – improve the result and condition stop.
  - TrainParam.lr = 0.0001; learn rate, value to update weight – improve the result.
  - TrainParam.goal = 0; condition stop.
  - TrainParam.min_grad = 0.0000000001: value change in min_grad – improve the result.
  - Threshold if the value is zero then unknown, less than zero set to (-1), if the value more than zero set to (1).

The degree of accuracy of this net was (95.9303 %), the figure (4.4) shows the train of the neural network:

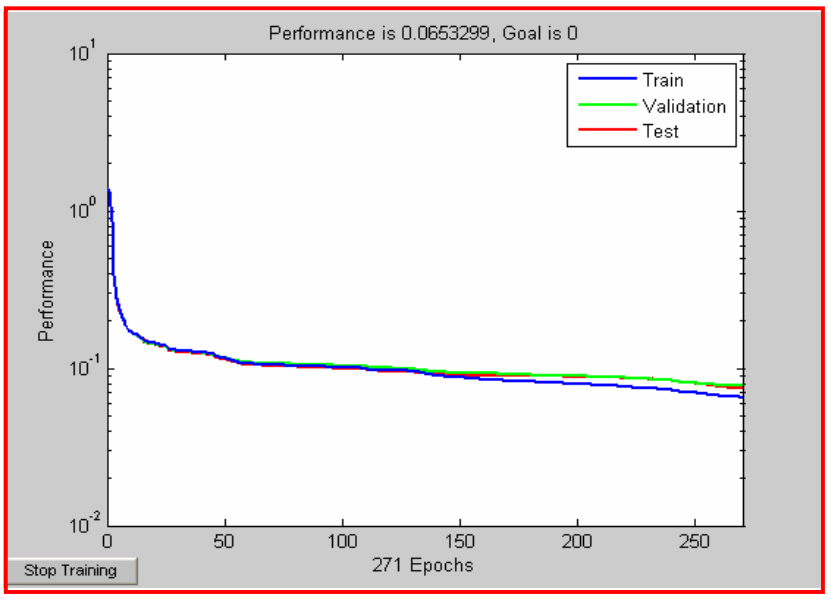


Figure (4.4) illustrates the training process

- ***The second structure neural network*:** two hidden layers, the first layer contains ***twenty*** neurons, the second layer contains ***five*** neurons. The parameters used in this Feed-forward Neural Network (NN) are:
  - TrainParam.epochs = 500; Epoch number (Batch) – improve the result and condition stop.
  - TrainParam.lr = 0.001; learn rate, value to update weight – improve the result.
  - TrainParam.goal = 0; condition stop.
  - TrainParam.min_grad = 0.0000000001: value change in min_grad – improve the result.
  - Threshold if the value is zero then unknown, less than zero is set to (-1), if the value more than zero is set to (1).

  In this neural network the accuracy obtained is (94.2778 %). Figure (4.5) shows the train of the neural network:

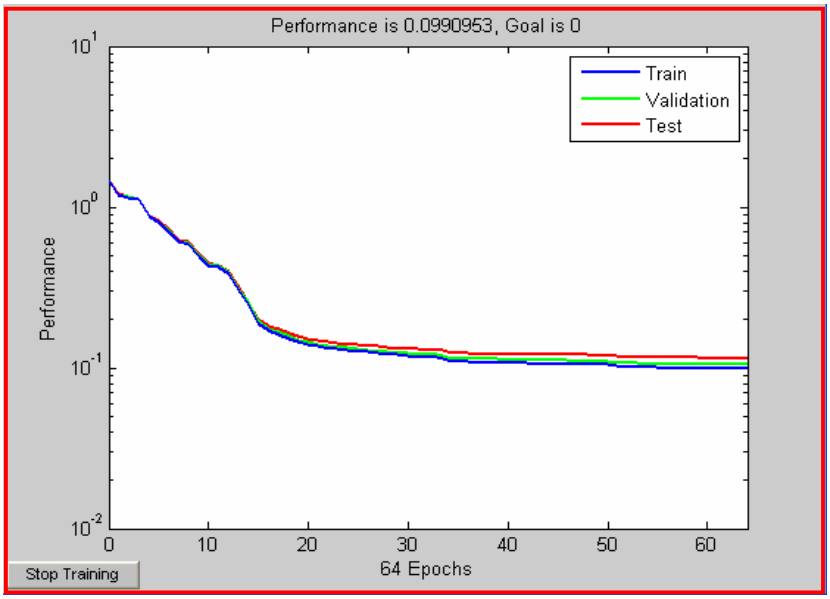


Figure (4.5) illustrates the training process

- ***The third structure neural network*:** the third neural network consists of two hidden layers, the first layer contains of ***twenty-two*** neurons and the second

layer contains ***five*** neurons. In addition the parameters used in this Feed-forward Neural Network (NN) are:

- TrainParam.epochs = 500; Epoch number (Batch)) – improve the result and condition stop.
- TrainParam.lr = 0.00001; learn rate, value to update weight – improve the result.
- TrainParam.goal = 0; condition stop.
- TrainParam.min_grad = 0.0000000001: value change in min_grad – improve the result.
- Threshold if the value is zero or less than zero is equal (-1), otherwise i.e. if the value more than zero is equal (1).

It is this neural network; the degree of accuracy was (95.1856 %). Figure (4.6) below shows the train of the neural network.

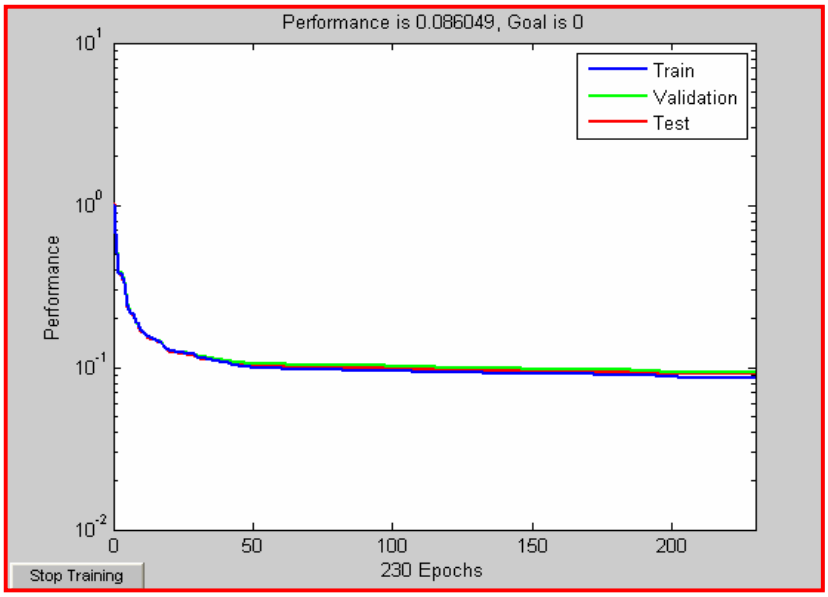


Figure (4.6) illustrates the training process

Now select the best neural networks based on the highest accuracy, therefore select the first neural network architecture, at which the degree of accuracy was network ***(95.9303 %)***.

After choosing the best NN, train it with different datasets and keep the weights of the best training accuracy. During the testing phase, calculate the total accuracy, true positive, false positive, true negative, false negative. To illustrate the number of the unknown records and type of each unknown record, in addition to illustrating the true positive and false positive rates.

Test1: Take the first subset and show the result of the testing part of that set, at which the total accuracy is calculated using equation (4.1).

Total Accuracy = 96.5033%

Table (4.1) illustrates the detection statistics of the testing results. Real field represents number of the connection records in the testing dataset of each corresponding type, ANN, shows the total records that are classified as the corresponding type, Match_type represents the connection records that are correctly classified, Miss_type shows the number of connection record that are miss classified, finally the Accuracy is calculated buy using equation (4.1).

Table (4.1) Detection Rate

Type	Real	ANN	Match type	Miss type	Accuracy
Normal	5725	4970	4925	45	86.0262
Probing	358	366	329	37	91.8994
DOS	22366	22305	22294	11	99.6781
U2R	24	33	23	10	95.8333
R2L	1527	2010	1380	630	90.3733
Unknown	0	316	0	316	Nan

Table (4.2) illustrates the number of missed classified connection records of each type with its percent out of total connection record per each type. It is clearly seen that the highest percentage miss classification is in normal connection type.

Table (4.2) Analysis of Missed Records

Type	Normal	Probing	DOS	U2R	R2L
Number	800 (0.139)	29 (0.081)	72 (0.003)	1 (0.041)	147 (0.096)

While table (4.3) shows the analysis (actual type) of each record classified as unknown.

Table (4.3) Analysis of records classified as Unknown

Type	Normal	Probing	DOS	U2R	R2L
Number	120	24	64	1	107

The most important factor is to measure the alarm rates since the issue is to find out the false alarms (especially false negative the most dangerous alarm). Table (4.4 a and b) show the number of alarms per each alarm_type, and the rates of alarm types.

- TP=filed of Match_type for Normal in table (4.1).
- TN= $\sum_{all-attck\_types} Match\_field$
- FN= filed of Miss_type for Normal in table (4.1).
- FP=Real field of normal - Match_type of normal - #of Normal in unknown table (4.3)

Table 4.4 (a) Alarm rates

Type	TP	TN	FP	FN
Number	4925	24026	680	45

The rates of the alarms are calculated using equations (4.2-4.5)

Table 4.4 (b) Alarm rates

Alarm Type	Accuracy
True positive	99.0946
True negative	97.2476
False negative	0.9054
False positive	2.7524

Finally, the analysis of false negative (FN) alarms sources (i.e. type of attacks that classified as normal) is illustrated in table (4.5). It shows that most FN alarms resulted mainly from R2L attack.

Table (4.5) False Negative

Type	Probing	DOS	U2R	R2L
Number	3	6	0	36

Figure (4.7) below shows the real subset enters the test and the true output of the neural network and accuracy of the real (actual) and output (desired):

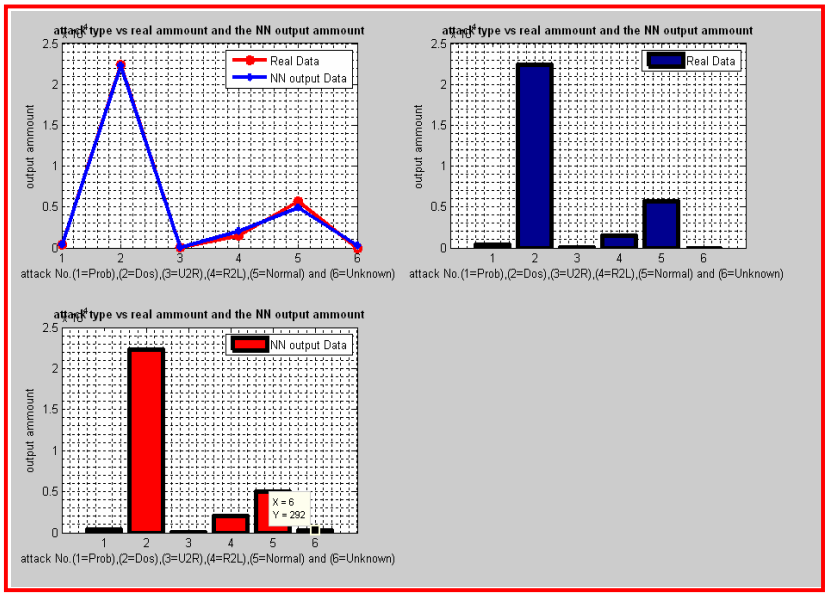


Figure (4.7) the test neural network with one subset

Test2: In this step test the suggested neural network classification accuracy and behavior using the whole dataset (311029) connection records to check if the net works on anomaly, misuse, or hybrid bases since defiantly the whole dataset will contain new connection records with novel attribute values. The total accuracy is calculated in equation (4.1).

Total Accuracy = 94.8314%

Table (4.6) illustrates the detection statistics of the testing results, its fields is as illustrated in "Test1". The highest classification rate is for DOS, the Lowest is for Normal, the worst classification accuracy at first was for U2R but we did improve it by increasing its amount of records in the training phase.

Table (4.6) Detection Rate

Name	Real	ANN	Match type	Miss type	Accuracy
Normal	60593	49691	48700	991	80.3723
Probing	4166	4049	3579	470	85.9097
DOS	229853	228437	228257	180	99.3056
U2R	230	390	208	182	90.4348
R2L	16187	24086	14209	9877	87.7803
Unknown	0	4376	0	4376	Nan

Table (4.7) illustrates the number of Missed records for each type, when the net behaves as anomaly detector the highest miss classification is in probing and R2L.

Table (4.7) Analysis of Missed Records

Type	Normal	Probing	DOS	U2R	R2L
Number	11893 (0.196)	587 (0.140)	1596 (0.006)	22 (0.095)	1978 (0.122)

Table (4.8) shows the analysis (actual type) of each record classified as unknown.

Table (4.8) Analysis of records classified as Unknown

Type	Normal	Probing	DOS	U2R	R2L
Number	1417	448	1081	21	1409

Table (4.4 a and b) shows the number of alarms per each alarm_type, and the rates of alarm types. The statistics are calculated at the same way shown in “Test1”.

Table 4.9 (a) Alarm rate

Type	TP	TN	FP	FN
Number	48700	246253	10476	991

Table 4.9 (b) Alarm Rate

Alarm Type	Accuracy
True positive	98.0057
True negative	95.9194
False negative	1.9943
False positive	4.0806

The analysis of false negative (FN) alarms sources (i.e. type of attacks that classified as normal) is illustrated in table (4.10). It shows that FN alarms resulted from R2L and DOS.

Table (4.10) False Negative

Type	Probing	DOS	U2R	R2L
Number	78	431	0	482

Figure (4.8) explains the real subset (actual) of the test and the true output (desired) of the neural network and a comparison between the actual and desired output:

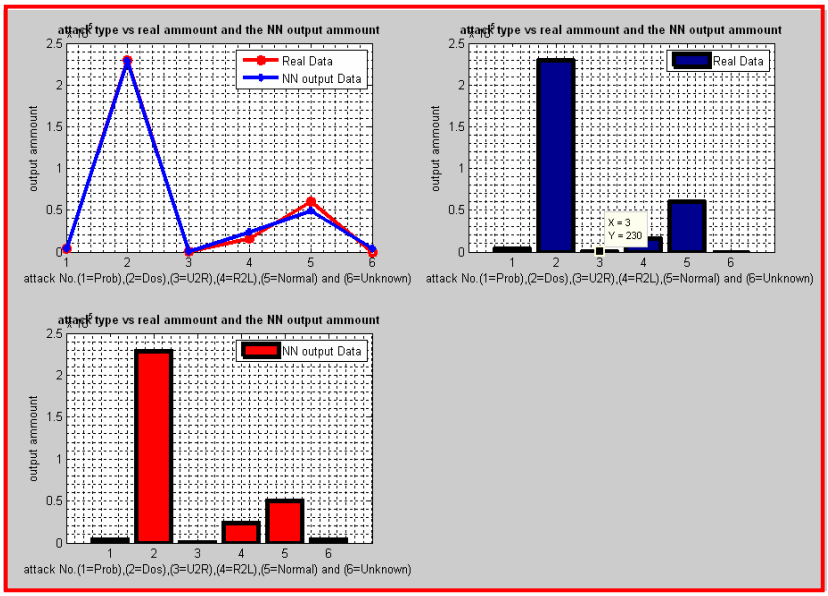


Figure (4.8) the test neural network with all subset

4.4.2 Result the Train and Test Neural Network with Fuzzified Data

To select suitable Neural network architecture, train different neural networks with the same dataset and select the neural with the highest classification accuracy, choose a subset, the subset contains the (30000) record (i.e. contains the subset of all different types of connection records) train the nets with the validation data part of the subset. Figure (4.9) shows the size of the subsets record:

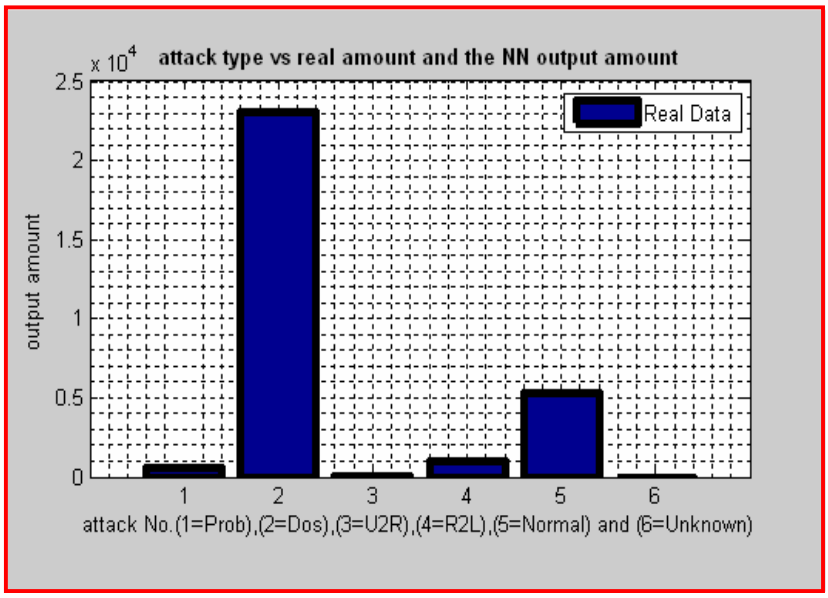


Figure (4.9) ratios of data that have been tested

'Normal' 5341, 'Probing' 601, 'DOS' 23061, 'U2R' 10, 'R2L' 987, the chosen subset contains the 30000 additional connection records of the attack type U2R to improve accuracy since U2R always cause classification problem due to its limited amount of connection records of its type. The following are some promising architectures among large number of tested architectures:

- ***The first neural network architecture***: In this neural networks, one hidden layer composed of ***seven*** neurons (in the hidden layer) is used. The parameters used are:
  - TrainParam.epochs = 500; Epoch number (Batch)) – improve the result and condition stop.
  - TrainParam.lr = 0.00001; learn rate, value to update weight – improve the result.
  - TrainParam.goal = 0; condition stop.
  - TrainParam.min_grad = 0.00000000001;: value change in min_grad – improve the result.
  - Threshold: if the value is zero then unknown, less than zero then set to (-1), if the value greater than zero set to (1).

The suggested neural network obtains a degree of accuracy (**97.4890** %). Figure (4.10) shows the behavior of training the neural network.

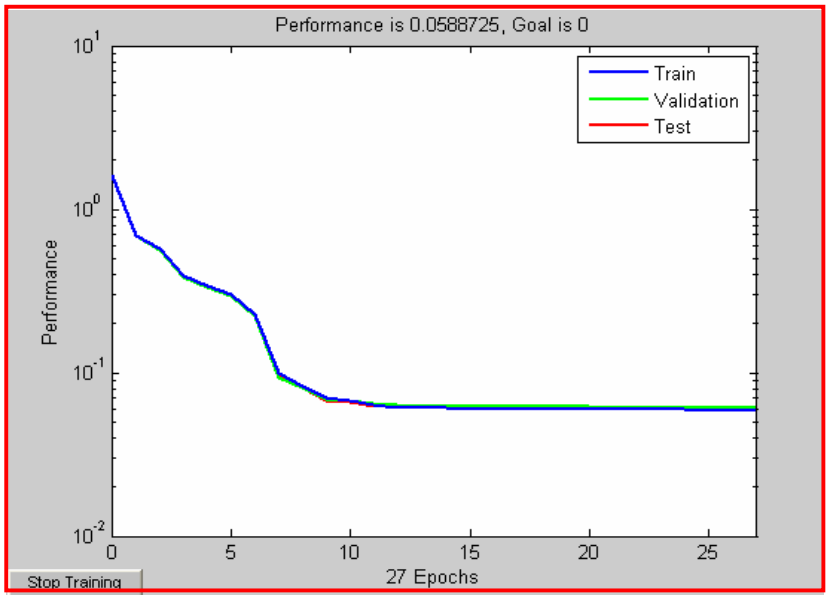


Figure (4.10) illustrates the training process

- ***The second neural network architecture*** use one hidden layer composed of ***four*** neurons (in the hidden layer). The parameters used in this Feed-forward neural network (NN) are:
  - TrainParam.epochs = 500; Epoch number (Sequential) – improve the result and condition stop.
  - TrainParam.lr = 0.000001; learn rate, value to update weight – improve the result.
  - TrainParam.goal = 0; condition stop.
  - TrainParam.min_grad = 0.00000000001: value change in min_grad – improve the result.
  - Threshold: if the value is zero then unknown, less than zero then set to (-1), if the value greater than zero set to (1).

The degree of accuracy of this neural network is (**96.6533 %**). Figure (4.11) below shows neural network training:

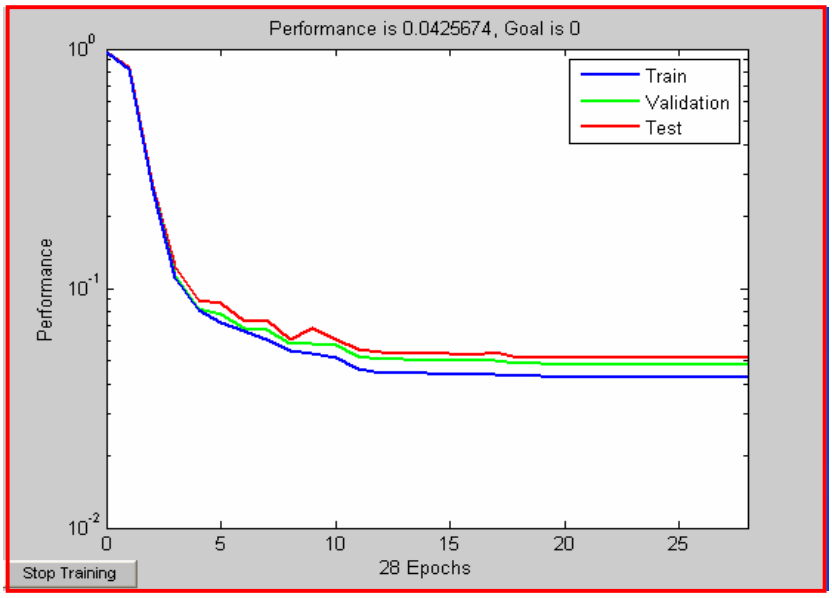


Figure (4.11) illustrates the training process

- ***The third neural network architecture*** will use one hidden layer composed of ***five*** neurons in the hidden layer. The parameters used in this Feed-forward neural network are:
  - TrainParam.epochs = 500; Epoch number (Batch)) – improve the result and condition stop.
  - TrainParam.lr = 0.000001; learn rate, value to update weight – improve the result.
  - TrainParam.goal = 0; condition stop.
  - TrainParam.min_grad = 0.00000000001: value change in min_grad – improve the result.
  - Threshold: if the value is zero then unknown, less than zero then set to (-1), if the value greater than zero set to (1).

  In this neural network, the degree of accuracy is (**95.3333 %**). Figure (4.12) shows the training behavior of the neural network.

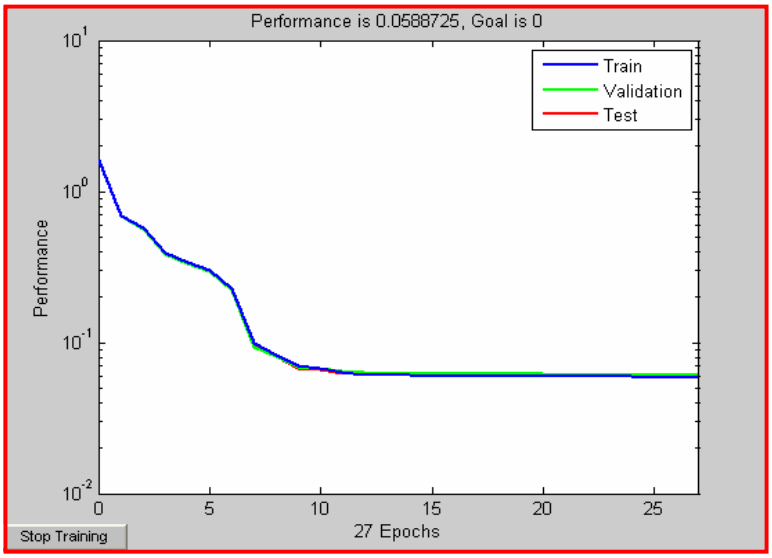


Figure (4.12) illustrates the training process

Now select the best neural network architecture from testing accuracy point of view. Therefore select the first structural, at which the obtained degree of the train in this network is (***97.4890 %***).

In the testing phase, at the beginning take the three parts of a subset to check accuracy of the ANN, in this phase calculate the total accuracy of the test, true positive, false positive, true negative, false negative. Analyze and illustrate the unknown records and the original type of each unknown connection record, in addition to the defining the Alarm rates (using the equations illustrated in the previous section).

Test3: Take the first subset and show the result of the testing part of that set, at which the total accuracy is calculated using equation (4.1).

Total Accuracy= 97.2700%

The detection statistics of the testing results are shown in table (4.11). Real field represents number of the connection records in the testing dataset of each corresponding type, ANN, shows the total records that are classified as the corresponding type, Match_type represents the connection records that are correctly classified, Miss_type shows the number of connection record that are miss classified, finally the Accuracy is calculated buy using equation (4.1).

Table 4.11 Detection Rate

Name	Real	ANN	Match type	Miss type	Accuracy
Normal	5341	4711	4682	29	87.6615
Probing	601	595	595	0	99.0017
DOS	23061	22962	22954	8	99.5360
U2R	10	10	10	0	100
R2L	987	1560	940	620	95.2381
Unknown	0	162	0	162	Nan

The analysis of the miss-classified connection records is shown in table (4.12) along with its percentage out of the total connection of the corresponding type, U2R miss classification is vanished and the highest miss classification rate is in normal connections.

Table (4.12) Analysis of miss classified connection record

Type	Normal	Probing	DOS	U2R	R2L
Number	659 (0.123)	6 (0.009)	107 (0.004)	0	47 (0.047)

Table (4.13) shows the analysis of each connection record classified as unknown.

Table (4.13) Analysis of records classified as Unknown

Type	Normal	Probing	DOS	U2R	R2L
Number	38	6	107	0	11

Table (4.14 a and b) shows the number of alarms per each alarm_type, and the rates of alarm types. The statistics are calculated at the same way shown in "Test1". Generally, the false alarms are reduced compared with results of "Test1" i.e. the data fuzzification improves the detection ability.

Table 4.14 (a) Alarm Rates

Type	TP	TN	FP	FN
Number	4682	24499	621	29

The rates of the alarms are calculated using equations (4.2-4.5)

Table 4.14 (b) Alarm Rates

Alarm Type	Accuracy
True positive	99.3844
True negative	97.5279
False negative	0.6156
False positive	2.4721

Finally, the analysis of false-negative (FN) alarms to specify its sources (i.e. type of attacks that classified as normal) is illustrated in table (4.15). It shows that all FN alarms resulted mainly from R2L attack.

Table (4.15) False Negative

Type	Probing	DOS	U2R	R2L
Number	0	0	0	29

The figure (4.13) below explains the real subset enters the test and the true out put of the neural network and accuracy between the real and output:

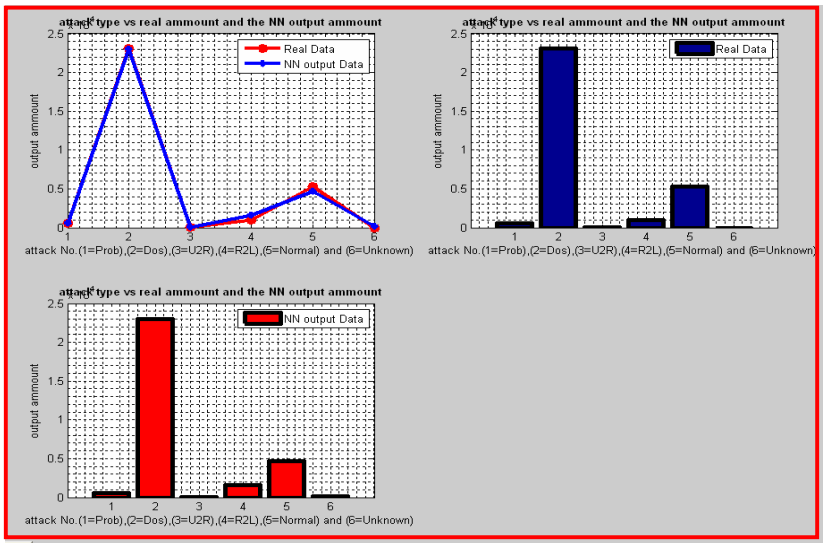


Figure (4.13) the test neural network with one subset

Test4: In this step test the suggested neural network classification accuracy and behavior using the whole dataset (311029) connection record to check if the net works on anomaly,

misuse, or hybrid bases since defiantly the whole dataset will contain new connection records with novel attribute values. The total accuracy is calculated in equation (4.1).

Total Accuracy = **97.0038%**

Table (4.16) illustrates the detection statistics of the testing results, its fields is as illustrated in "Test3". The highest classification rate is for DOS, the Lowest is for Normal, the worst classification accuracy at first was for U2R but we did improve it by increasing its amount of records in the training phase.

Table (4.16) Detection Rate

Name	Real	ANN	Match type	Miss type	Accuracy
Normal	**60593**	**53361**	**52932**	**429**	**87.3566**
Probing	**4166**	**4089**	**4088**	**1**	**98.1277**
DOS	**229853**	**229124**	**229015**	**109**	**99.6354**
U2R	**230**	**231**	**228**	**3**	**99.1304**
R2L	**16187**	**22688**	**15447**	**7241**	**95.4284**
Unknown	**0**	**1536**	**0**	**1536**	**Nan**

Table (4.17) illustrates the number of Missed records for each type, when the net behaves as anomaly detector the highest miss classification is in Normal.

Table (4.17) Analysis of Miss classified records

Type	Normal	Probing	DOS	U2R	R2L
Number	**7661 (0.126)**	**78 (0.018)**	**838 (0.003)**	**2 (0.008)**	**740 (0.045)**

Table (4.18) shows the analysis (actual type) of each record classified as unknown

Table (4.18) Analysis of records classified as Unknown

Type	Normal	Probing	DOS	U2R	R2L
Number	**586**	**77**	**662**	**2**	**209**

Table (4.19 a and b) shows the number of alarms per each alarm_type, and the rates of alarm types. The statistics are calculated at the same way shown in "Test1".

Table 4.19 (a) Alarm Rate

Type	TP	TN	FP	FN
Number	52932	248778	7075	429

Table 4.19 (b) Alarm Rate

Alarm type	Accuracy
True positive	99.1960
True negative	97.2347
False negative	0.8040
False positive	2.7653

Finally, the analysis of false negative (FN) alarms sources (i.e. type of attacks that classified as normal) is illustrated in table (4.20). It shows that most FN alarms resulted mainly from R2L attack.

Table (4.20) false Negative

Type	Probing	DOS	U2R	R2L
Number	0	0	0	429

Figure (4.14) below explains the real subset class (actual) the test and the true output of the neural network (desired) and a comparison between the actual and desired output:

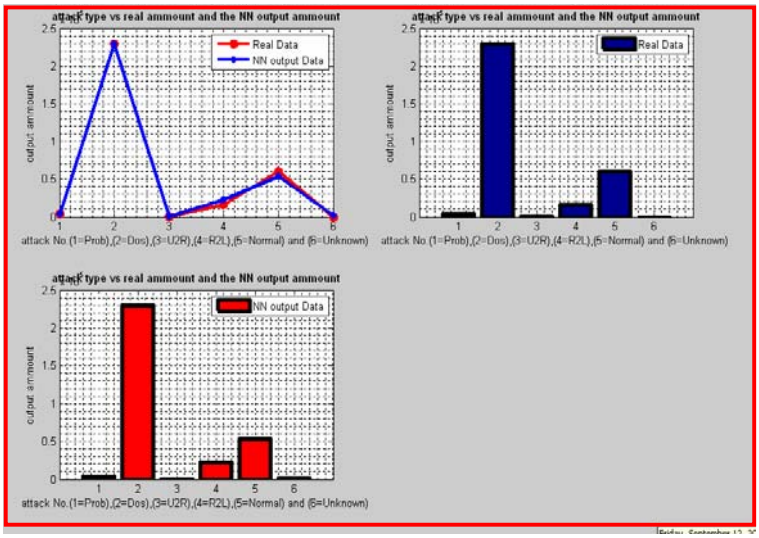


Figure (4.14) the test neural network with all subset

Chapter Five
Conclusion and future works

5.1 Introduction

In the previous chapter, two classifiers were suggested. These classifiers were trained with various types of feature sets and their experimental results were analyzed. This chapter includes conclusions obtained from this work followed by some recommendations for future work that could contribute to the development of more accurate classifiers for intrusion detection.

5.2 Conclusions

The main contribution of the present work is to achieve a classification model with high intrusion detection accuracy and mainly with low false negative. This was done through the design of a classification process for the problem using neural network and neural network with Fuzzification data for the detection of various types of attacks.

From the given results, the following substantial remarks were obtained:

1. The neural networks were able to produce good detectors that give a good estimation of the amount of difference from the normal. This shows that it is possible to apply the suggested classifiers to detect anomalies on real network traffic data.
2. Two classifiers were suggested (Neural Network, Neural Network with Fuzzification data) with best classification accuracy **95.9303%** and **97.4890%** respectively. The suggested neural network with Fuzzification data is more efficient and meets the contribution of this work.
3. Both anomaly detection and misuse detection are supported by the system. This was obvious from the ability of the designed NN (with fuzzified data) of giving approximately the same recognition ability (**97.0038%**) when tested with whole data set (311029) connection record) not only the trained set (30000 connection record) at which the testing result was (**97.2700%)**. The training data set contains only 25 attack type out of 39, while during testing phase; the suggested IDs

recognize more than 25 attack type. This shows that the designed net gives the ability to respond to anomalies and not only to signatures of known attacks.

4. The result of training and testing the neural network with fuzzification data highly reduces the false negative alarms (to 0.804 %) compared with the NN trained with non fuzzified data set (the false negative rate was (**1.9943** %) and that the false negative alarms only caused by R2L attack for fuzzified data set, while for non fuzzified data set caused by Prob, DOS, R2L.

5.3 Future Work

1. Built an OCON (one class one neural) for each attack and study its ability to detected attacks and reduce the false alarms.
2. In this work, we used 12 features to classify connection records, as future work try to study the most suitable feature for each type of normal and attack, for example, in the normal use the feature number (3,11,21,40), in the probing use (1,3,12,18,20,21,23,26), in the Dos use (1,8,10,11,16,17,20,21,23,28,29,31), in the U2R use (11,14,17,28,29,32,36,38), and R2l use (1,3,11,12,18,20,22,25,38). This could be done with OCON structure
3. Another improvement of the specific intrusion system developed is the use of real-time intrusion detection, because such a system can catch a range of intrusion like viruses, Trojan horses, and masquerading before these attacks have time to do extensive damage to a system.
4. Design a classifier depending on the idea of data mining and compare its behavior with this work.

References

1. **[Abr05]** Ajith Abraham1, Ravi Jain2, Sugata Sanya3, and Sang Yong Han,"**D-SCIDS; Distributed Soft Computing Intrusion Detection System**", IWDC, LNCS 3326, Verlag Berlin Heidelberg, 2005, pp. 87-88.
2. **[And80]** Anderson, J.P., "**Computer security threat monitoring and surveillance"**, Technical Report, James P. Anderson Company, Fort Washington, PA, April 1980.
3. **[Axe99]** Stefan Axelsson, "**Research in Intrusion –Detection System: A Survey**", NUTEK project10435, Sweden, 1999, pp.1-25.
4. **[Bac01]** Rebecca Bace & Peter Mell, "**Intrusion Detection Systems**", Released by National Institute of Standards and Technology (NIST), 2001 last accessed in 29-7-2007,pp,5-55. http://www.21cfrpart11.com/files/library/goverment/intrusion_detection_systems_0201 _draft_pdf
5. **[Bou04]** Yacine Bouzida, Frederic Cuppens, Nora Cuppens-Boulahia & Sylvain Gombault, "**Efficient Intrusion Detection Using Principal Component Analysis**", 2004 http://www.rennes.enstbretagne.fr/~fcuppens/article/sar04.pdf
6. [**Cha07**] Ray-I Chang, Liang-Bin Lai , Wen-De Su, Jen-Chieh Wang, Jen-Shiang Kouh "**Intrusion Detection by Back propagation Neural Networks with sample-Query and Attribute-Query**" , International Journal of Computational Intelligence Research, Vol.3, No.1, 2007 , pp,6-10.
7. **[Che07]** Yuehui Chen, Ajith Abraham, Bo Yang," **Hybrid Flexible Neural-Tree-based Intrusion Detection Systems**", International journal of intelligent systems, VOL. 22, 2007, pp337–352. www.softcomputing.net/ijis.pdf
8. **[Che05]** Chebrolu S, Abraham A, Thomas JP." **Feature detection and ensemble design of intrusion detection systems**". Comput Secur; 24, 2005, pp.295–307.
9. **[Chi01]** Adhity Chittur, "**Model Generation for an Intrusion Detection System Using Genetic Algorithm**", NY, 2001.

www1.cs.columbia.edu/ids/publications/gates-thesis01.pdf

10. **[Cro03]** Tim Crothers**, "Implementing Intrusion Detection Systems",** John Wiley & Sons Inc, 2003.
11. **[Das01]** Dipankar Dasgupta & Fabio A. Gonzalez, " **Evolving Complex Fuzzy Classifier Rules Using a Linear Tree Genetic Representation**", Proceeding of the Genetic and Evolutionary Computation Conference, GEC00, 2001
12. **[Den87]** Dorothy E. Denning, "**An Intrusion-Detection Model**", IEEE Transactions on Software Engineering, Vol.SE-13.No.2, February 1987, pp.222-232.
13. **[Den82]** Dorothy E. Denning, "**Cryptography and Data Security"**. Addison Wesley, Reading, Massachusetts, 1982.
14. **[Eri00]** Eric A. Fisch, Gregory B. White, "**Secure Computers and Networks"**, CRC Press, 2000.
15. **[Fau94]** L.Fausett, "**Fundamental of Neural Network**", Printice-Hall International, Inc, 1994.
16. **[Gom03]** Jonatan Gomez, Dipankar Dasgupta, Olfa Nasraoui, & Fabio Gonzalez, "**Complete Expression Trees for Evolving Fuzzy Classifier System with Algorithm and Application to Network Intrusion Detection**", Proceeding of the Genetic and Evolutionary Computing Conference, GECOO,2003.
17. **[Gom02]** Jonatan Gomez & Dipankar Dasgupta, "**Evolving Fuzzy Classifiers for Intrusion Detection**", Proceedings of the 2002 IEEE Workshop on Information Assurance United State Military Academy, West Point, NY June 2001
18. **[KDD99] KDD-cup data set. Link:** http://kdd.ics.uci.edu/data bases /kddcup99/kddcup.html.
19. **[Kem02]** R. A. Kemmerer and G. Vigna, "**Intrusion detection: a brief history and overview**", Computer, vol. 35, no. 4, 2002, pp. 27–30.
    Www. ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1012428.
20. **[Kim02]** Jung won Kim, "**Integrating Artificial Immune Algorithms for Intrusion Detection**", MSc Dissertation, University of London, 2002, pp,1-5.

21. **[Kum95]** Sandeep Kumar, "**Classification and Detection of Computer Intrusions**", A PHD Dissertation to the Faculty of Purdue University, August 1995, pp.49-51.
22. **[Mal02]** Saadat Malik," **Network Security Principles and Practices**", Cisco Press, 2002.
23. **[Man02]** Manikopoulos, C., Papavassiliou, and S.," **Network intrusion and fault detection**": A statistical anomaly approach. Commun. Mag. IEEE 40 (10), 76–82., 2002, pp,1-20.
24. **[Mth07] Matlab online support,** www.mathworks.com access helpdesk techdoc Matlab.shtml.
25. **[Muk04]** Srinivas Mukkamalaa, Andrew H. Sunga, Ajith Abraham." **Intrusion detection using an ensemble of intelligent paradigms**", Elsevier Ltd, 2004, pp.168-175.
26. **[Myk94]** Mykerjee B., Heberlein L. T., & Levitt K. N., " **Network Intrusion Detection**", IEEE Network, Vol.8, No.3, 1994, pp. 26-14.
27. **[Rum86]** D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Leaning Internal. Representations by Error Propagation in Rumelhart, D. E. and McClelland, J. L., "**Parallel Distributed Processing**": Explorations in the Microstructure of Cognition. MIT Press, Cambridge Massachusette, 1986.
28. **[Por92]** Porras, P. A., "**STAT: A State Transition Analysis Tools for Intrusion Detection**", MSc Thesis, Department of Computer Science, University of California Santa Barbara, 1992, pp.15-20.
29. **[Pri97]** Katherine E. Price. "**Host-based misuse detection and conventional operating system 'audit data collection** ", A thesis Submitted to the Faculty of Purdue University, December 1997, pp. 6- 8.
30. **[Kaz04]** Przemyslaw Kazienko & Piotr Dorosz**,"Intrusion Detection Systems (IDS) part2- Classification; method; techniques",** 2004, http://www.windowsecurity.com/articles/IDS-part2-classification-methods-techniques.html**.**

31. **[Sto00]** Salvatore J. Stolfo, Wei Fan, Wenke Lee, Andreas Prodromidis, & Philip K. Chan, "**Cost-based modeling for Fraud and Intrusion Detection: Results from the JAM Project**", 2000, pp.1-15 http://www1.cs.columbia.edu/jam/recent-project-papers.html .
32. **[Sun02]** Aurobindo Sundaram, "**An Introduction to Detection** ", ACM'c First Electronic Publiction 2002.
33. [**Suz07**] Al-naqshabandi Susan M., "**Simulation system for computer network intrusion detection**", a PhD thesis, Al-Nahrain University, Baghdad, Iraq, 2007, pp.61-66.
34. **[Tao07]** Tao Song, "**Formal Reasoning about Intrusion Detection Systems**", a dissertation submitted in partial satisfaction of the requirements for the degree of doctor of philosophy in computer science in the office of graduate studies of the university of California, 2007.
35. [**Ves07**] Veselina G. Jecheva, Evgeniya P. Nikolova," **Learning Problem and BCJR Decoding Algorithm in Anomaly-based Intrusion Detection Systems"** Journal of software, VOL. 2, NO. 6, December 2007, pp, 48-50.
36. **[Yao02]** J.T Yao, S.L. Zhao, L.V. Saxton,"**A Study on Fuzzy Intrusion Detection**", 2002. Link: www2.cs.uregina.ca/~jtyao/Papers/**detection**.pdf.
37. **[Zad65]** L.A. Zadeh, Fuzzy Sets, "**Information and Control**", 8(31965), pp.338-353, 1965.
38. **[Zha05]** Chunlin Zhang, Ju Jiang, Mohamed Kamel." **Intrusion detection using hierarchical neural networks**". Pattern Recognition Letters 26, 2005, 779–791
39. **[Zur96]** J.M.Zurada, "**Introduction to Artificial Neural Systems"**, JAICO publishing House, 1996.

الخلاصة

تعتبر أنظمة كشف الاختراق احد أهم الحقول البحثية في مجال أمنيه الشبكات والتي يمكِن أن تُستَعملُ لتدعيم أمن أنظمةِ الشبكات. نظام كشف الاختراق (IDS) بإمكانه اكتشاف الأنماط المشكوك بها إثناء مرورها عبر الشبكة من خلال مراقبة فعاليات المستخدم, وبعدها تحليل هذه الفعاليات. إن المشكلةَ الرئيسيةَ هي صعوبةُ التَمييز بين السلوكِ الطبيعي (Normal) والسلوكِ الشاذّ(abnormal) في شبكاتِ الحاسوبِ. قد تتولد إثناء عمليةِ الكشفُ (إنذارات كاذبة) تنتج مِن: ***استعمال أنظمة كشف الاختراق مستند على (Anomaly Intrusion Detection Systems)، وكذلك نتيجة التداخل الكبير بين الخواص الواصفة لكل اتصال (overlap of attributes of the connection records).***

وان استخدام (Fuzzy set) قد يقلل من هذه الإنذارات الكاذبة, حيث إن درجة العلاقة المستخدمة ستقوم بعزل هذا التداخل مابين السلوك الطبيعي والسلوك الشاذ ألا وهو المتطفل أو الدخيل في شبكات الحاسوب لذا نحتاج ضبضبه للبيانات (fuzzification) قبل عملية التصنيف.

إن الهدف من هذا العمل هي المساهمة في إيجاد طرق حل تساعدنا في اكتشاف الدخيل (Intrusion) من خلال بناء نظام اكتشاف ذكي باستخدام الشبكات العصبونية ذات التغذية الأمامية بمعدل اكتشاف عالي وبنفس الوقت بمعدل إنذارات كاذبة اقل (False negative) والذي يعتبر من اخطر أنواع الإنذارات كونها تميز السلوك الشاذ على انه سلوك طبيعي , وكذلك التقليل من إنذارات (False Positive) أي إن النظام يولد إنذار كون السلوك شاذ برغم انه في الحقيقة سلوك طبيعي. سيتم في هذا البحث بناء نظامين الأول هو شبكات عصبونية ذات تغذية أمامية مع بيانات غير مضببه والنظام الأخر هو شبكات عصبونية ذات تغذية أمامية مع بيانات مضببه (fuzzified data) .

سيتم تقييم النظام من خلال استخدام بيانات قياسية (KDD1999) مقترحة من قبل (MIT) والمستخدمة في (Lincoln Labs), هذه البيانات مقسمه إلى خمسه أصناف رئيسية وهي (Normal data, Probing attack, Dos attack, U2R attack, R2L attack). وسيتم تدريب الشبكة العصبونية المصممه باستخدام 12 خاصية من خواص كل اتصال بدل من 41 وبهذا سوف نقلل من الوقت اللازم لتدريب ألشبكة المصممه, يستخدم أسلوب المحاولة والخطاء (trail and error) لاختيار التصميم الأفضل للشبكة العصبونية لتصبح أكثر قدره على اكتشاف الدخلاء.

إن النظام المصمم ليس فقط له ألقدره على التمييز مابين السلوك الطبيعي والشاذ ولكن أيضا تصنيف نوع السلوك الشاذ (الأصناف التي ذكرت مسبق). كانت درجة الدقة في نظام الاكتشاف

باستخدام الشبكات العصبونية ذات التغذية الأمامية مع البيات الحقيقة هي (95.9303%). بينما وصلت درجة دقة النظام باستخدام الشبكات العصبونية ذات التغذية الأماميـ ـة مع البيانات مضببه (fuzzified data) إلى (97.4890%).

بسم الله الرحمن الرحيم

جامعة أل البيت

كلية الأمير الحسين بن عبد الله لتكنولوجيا المعلومات

رسالة ماجستير في علم الحاسوب

كشف الاختراق باستخدام الشبكات العصبونية ذات التغذية الأمامية

Intrusion Detection Using Feed-forward Neural Networks

إعداد الطالب

خطاب معجل علي ألعبيدي

0620901007

اسم المشرف

الدكتورة فينوس وزير السماوي

الفصل الأول

2009/2008