STATUS OF THESIS

Title of thesis: Feature Subset Selection in Intrusion Detection Using Soft Computing Techniques

I, IFTIKHAR AHMAD

hereby allow my thesis to be placed at the Information Resource Center (IRC) of Universiti Teknologi PETRONAS (UTP) with the following conditions:

1. The thesis becomes the property of UTP
2. The IRC of UTP may make copies of the thesis for academic purposes only.
3. This thesis is classified as

☐ Confidential

☑ Non-confidential

If this thesis is confidential, please state the reason:

_____
_____
_____

The contents of the thesis will remain confidential for _____ years.
Remarks on disclosure:

_____
_____
_____

Endorsed by

_____ _____

Signature of Author
Iftikhar Ahmad
Department of Computer Science, FUUAST, G-8
Islamabad, Pakistan.
Date: _____

Signature of Supervisor
Assoc. Prof. Dr. Azween Bin Abdullah
Department of Computer Information Sciences (CIS)
Universiti Teknologi PETRONAS
Date: _____

UNIVERSITI TEKNOLOGI PETRONAS

FEATURE SUBSET SELECTION IN INTRUSION DETECTION USING SOFT COMPUTING TECHNIQUES

By

IFTIKHAR AHMAD

The undersigned certify that they have read, and recommend to the Postgraduate Studies Programme for acceptance this thesis for the fullfilment of the requirements for the degree stated.

Signature: _____

Main Supervisor: Assoc. Prof. Dr. Azween Bin Abdullah

Signature: _____

Head of Department: Dr. Mohd Fadzil Bin Hassan

Date: _____

FEATURE SUBSET SELECTION IN INTRUSION DETECTION USING SOFT COMPUTING TECHNIQUES

by

IFTIKHAR AHMAD

A Thesis

Submitted to the Postgraduate Studies Programme

As a requirement for the degree of

DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER AND INFORMATION SCIENCES

UNIVERSITI TEKNOLOGI PETRONAS

BANDAR SERI ISKANDAR

PERAK

FEBRUARY 2011

DECLARATION OF THESIS

Title of thesis | Feature Subset Selection in Intrusion Detection Using Soft Computing Techniques

I, IFTIKHAR AHMAD

hereby declare that the thesis is based on my original work except for quotations and citations, which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at UTP or other institutions.

Witnessed by

Signature of Author	Signature of Supervisor
Iftikhar Ahmad	Assoc. Prof. Dr. Azween Bin Abdullah
Department of Computer	Department of Computer
Science, FUUAST, G-8	Information Sciences (CIS)
Islamabad, Pakistan.	Universiti Teknologi PETRONAS
Date: _____	Date: _____

To my beloved parents

ACKNOWLEDGEMENT

I am very thankful to Almighty Allah, the most beneficent, the most merciful and the most gracious, for enhancing my courage for the completion of this work successfully. It is matter of great honor and pleasure for me to express my ineffable gratitude and profound indebtedness to *Prof. Dr. Azween Bin Abdullah* for his kind supervision valuable suggestions and intellectual activities, inexhaustible energy to steer forth the students. I also appreciate efforts and cooperation of management of Universiti Teknologi PETRONAS, Malaysia that makes possible for me to conduct the implementation of this research. I am also greatly obliged to my co-supervisor *Prof. Dr. Abdullah Sharaf Alghamdi*, for his keen interest, guidance and moral support. Also expresses my thanks to all my colleagues who helped me whenever needed.

I am also greatly obliged to Dr. Muhammd Hussain, Dr. Mohsin Iftikhar, Dr. A. Rahim, Dr. Anwar, and Dr. Shamim Husain for their kind research guidance and moral support. I feel proud to have such a nice and distinguished ASER group fellows, FUUAST colleagues, and friends, Dr. Mahmat Issa, Irshad Ahmad Sumro, Ayaz Arshad,Imran Baig, Masood Ur Rehman, Hanif Ullah, Gul Faraz, Tazar Hussain, Muhammad Nasir, Syed Amanullah Qadri, Aman Ul Haq and other respected fellows for making the whole period a golden era.

I inexpressibly fall short of diction to express my humble obligation to my parents whose hands always rise in pray for my success and because of their whole moral support; I feel my entity at this stage. Finally, I apologize if I have happened to annoy anybody during studies in this great institution. The errors that remain are mine alone.

ABSTRACT

Intrusions on computer network systems are major security issues these days. Therefore, it is of utmost importance to prevent such intrusions. The prevention of such intrusions is entirely dependent on their detection that is a main part of any security tool such as Intrusion Detection System (IDS), Intrusion Prevention System (IPS), Adaptive Security Alliance (ASA), checkpoints and firewalls. Therefore, accurate detection of network attack is imperative. A variety of intrusion detection approaches are available but the main problem is their performance, which can be enhanced by increasing the detection rates and reducing false positives. Such weaknesses of the existing techniques have motivated the research presented in this thesis.

One of the weaknesses of the existing intrusion detection approaches is the usage of a raw dataset for classification but the classifier may get confused due to redundancy and hence may not classify correctly. To overcome this issue, Principal Component Analysis (PCA) has been employed to transform raw features into principal features space and select the features based on their sensitivity. The sensitivity is determined by the values of eigenvalues. The recent approaches use PCA to project features space to principal feature space and select features corresponding to the highest eigenvalues, but the features corresponding to the highest eigenvalues may not have the optimal sensitivity for the classifier due to ignoring many sensitive features. Instead of using traditional approach of selecting features with the highest eigenvalues such as PCA, this research applied a Genetic Algorithm (GA) to search the principal feature space that offers a subset of features with optimal sensitivity and the highest discriminatory power.

Based on the selected features, the classification is performed. The Support Vector Machine (SVM) and Multilayer Perceptron (MLP) are used for classification purpose due to their proven ability in classification. This research work uses the Knowledge Discovery and Data mining (KDD) cup dataset, which is considered

benchmark for evaluating security detection mechanisms. The performance of this approach was analyzed and compared with existing approaches. The results show that proposed method provides an optimal intrusion detection mechanism that outperforms the existing approaches and has the capability to minimize the number of features and maximize the detection rates.

ABSTRAK

Pencerobohan ke atas sistem rangkaian komputer merupakan isu keselamatan yang utama dewasa ini. Maka, adalah sangat penting untuk menghalang daripada pencerobohan ini. Langkah-langkah pencegahan ini bergantung sepenuhnya kepada sistem pengesanan di mana ia merupakan bahagian terpenting kepada alat keselamatan seperti *Intrusion Detection System* (IDS), *Intrusion Prevention System* (IPS), *Adaptive Security Alliance* (ASA), *checkpoints* dan *firewalls.* Justeru dengan itu, pengesanan yang tepat daripada ancaman rangkaian perlu diberi perhatian. Terdapat pelbagai kaedah pengesanan pencerobohan tetapi masalah utama ialah prestasi, di mana ia perlu ditingkatkan dengan meningkatkan kadar pengesanan dan mengurangkan ketidaktepatan. Kelemahan teknik yang sedia ada ini telah memberi motivasi kepada kajian yang dipersembahkan dalam tesis ini.

Salah satu kelemahan kaedah pengesanan pencerobohan sedia ada adalah penggunaan set data mentah untuk pengkelasan tetapi pengelas mungkin keliru disebabkan oleh penindanan data yang mengakibatkan pengkelasan yang tidak betul. Untuk mengatasi masalah ini, Principle Component Analysis (PCA) telah digunakan untuk mengubah ciri-ciri mentah kepada ruang ciri-ciri utama dan memilih ciri-ciri berdasarkan kepada kepekaan mereka. Kepekaan ditentukan dengan eigenvalues. Kaedah terkini dengan menggunakan PCA untuk menghasilkan ciri-ciri ruang kepada ruang ciri-ciri utama dan memilih ciri-ciri bergantung kepada eigenvalues yang tertinggi, namun ciri-ciri ini mungkin tidak memiliki kepekaan yang optimum untuk pengkelasan disebabkan oleh banyak cirri-ciri yang sensitif. Kajian ini tidak menggunakan kaedah traditional dalam memilih ciri-ciri dengan menggunakan eigenvalues seperti PCA, sebaliknya kajian ini menggunakan satu Generic Algorithm (GA) untuk mencari ruang ciri-ciri utama yang menawarkan satu bahagian ciri-ciri dengan sensitiviti optimum dan kuasa diskriminasi yang tertinggi.

Berdasarkan kepada ciri-ciri pilihan, pengkelasan dilaksanakan. *Support Vector Machine* (SVM) dan *Multiplayer Perceptron* (MLP) telah digunakan untuk tujuan

pengkelasan kerana bukti keupayaan mereka dalam pengkelasan. Kerja kajian ini menggunakan *Knowledge Discovery* and *Data Mining* (KDD) set data cawan, dimana ia dianggap sebagai batu pengukur untuk menilai mekanisma pengesanan keselamatan. Prestasi kaedah ini telah dianalisa dan dibandingkan dengan kaedah-kaedah yang sedia ada. Keputusan menunjukan kaedah yang dibentangkan menghasilkan mekanisma pengesanan pencerobohan yang optima dan mengatasi kaedah-kaedah yang sedia ada dan mempunyai keupayaan untuk mengurangkan jumlah cirri-ciri dan memaksimakan kadar pengesanan.

.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

IDS	Intrusion Detection System
NN	Neural Network
PCA	Principal Component Analysis
GA	Genetic Algorithm
SVM	Support Vector Machine
DOS	Denial of Service Attack
R2L	Remote to Local
U2R	User to Root
KDD	Knowledge Discovery and Data Mining
SC	Soft Computing
ML	Machine Learning
MLP	Multilayer Perceptron
BPROP	Backpropagation
SNN	Supervised Neural Network

CHAPTER 1

INTRODUCTION

This chapter covers the motivation, the prevalent issues in the existing intrusion detection approaches based on which this work is carried out, and the objectives to be accomplished. It also describes relevant research questions, which should be answered, the methodology adopted, and the research workflow followed. Finally, the chapter concludes with the contributions and the organization of the thesis.

1.1 Research Motivations

The rapid development of computer networks and mostly of the Internet has created many stability and security problems such as intrusions on computer and network systems. Further, the dependency of companies and government agencies is increasing on their computer networks and the significance of protecting these systems from attack is serious because a single intrusion can cause a heavy loss or the consistency of network becomes insecure. During recent years, number of intrusions has dramatically increased. Therefore, it is very important to prevent such intrusions. The hindrance of such intrusions is entirely dependent on their detection that is a key part of any security tool such as Intrusion Detection System (IDS), Intrusion Prevention System (IPS), Adaptive Security Alliance (ASA), checkpoints and firewalls. Consequently, interest in network intrusion detection has increased among the researchers (Ahmad et al. 2009), (Ahmad et al. 2008). Several intrusion detection approaches are available but the main problem is their performance, which can be enhanced by increasing the detection rates and reducing false alarms. Such weaknesses of the existing techniques motivated the research presented in this dissertation.

1.2 Problems in Intrusion Detection

The accurate intrusion detection in computer and network systems has always been an elusive aim for system administrator and security researchers. Initially, traditional intrusion detection systems (IDSs) were developed. However, these systems have many limitations like time consuming statistical analysis, regular updating, non-adaptive, accuracy and flexibility. After this, intelligent IDSs (rule based, graphical and hybrid) were introduced but they also suffered many problems such as false positive, false negative, and performance efficiency. Recently, Neural Network (NN) inspired by nervous system has become an interesting tool in the applications of intrusion detection. But it still suffers from many problems; training/learning overhead, detection rate and false alarms (Zargar and Kabiri 2010).

One of the drawbacks of the existing intrusion detection approaches is the usage of a raw dataset for classification but the classifier may get confused due to redundancy and hence may not classify correctly. To handle this problem, Principal Component Analysis (PCA) has been applied to change raw features into principal features space and select the features based on their sensitivity (Liu and Yi 2006). The sensitivity is measured by the values of eigenvalues (Sun et al. 2004). The modern methods use PCA to project features space to principal feature space and pick features corresponding to the highest eigenvalues, but the features corresponding to the highest eigenvalues may not have the finest sensitivity for the classifier due to ignoring many sensitive features. Therefore, the selection of optimized subset of features is another important issue for the intrusion detection system. Other problems include the selection of dataset for training and testing, and the classifier architecture that classifies connections into normal and intrusive.

1.3 Research Objectives

The main goal of the research is to develop an optimized intrusion detection mechanism using soft computing techniques that provide the potential to identify network activity in a robust way. To achieve this goal, a number of specific objectives have been defined as follows:

- Transformation of raw features set of packet into new feature space.
- Selection of optimized subset of features that has higher discriminatory power.
- Selection of the most suitable architecture that identifies network activity into normal and intrusive.
- Development and implementation of the proposed model for intrusion detection.
- Train and test the develop system.
- Implement the developed system in different case studies.

1.4 Research Questions

Based on the abovementioned issues and problems, we can derive research questions as the following:

- Why did we move from conventional to unconventional IDS?
- Why did we use soft computing techniques?
- What are the necessary components (software, services) that are desirable to support the architectural framework?
- What are the specifications of the ideal network intrusion detection system (NIDS) that will ensure best performance?
- How can we map soft computing techniques to NIDS?
- Will the adaptation of this approach into NIDS improve NIDS in some ways or address some issues in NIDS?
- How can we implement /simulate my work?

1.5 Research Methodology

The overall research is divided into several phases, each one is concerned with the specific goals to finally fulfill the main objective. These phases are described as follows:

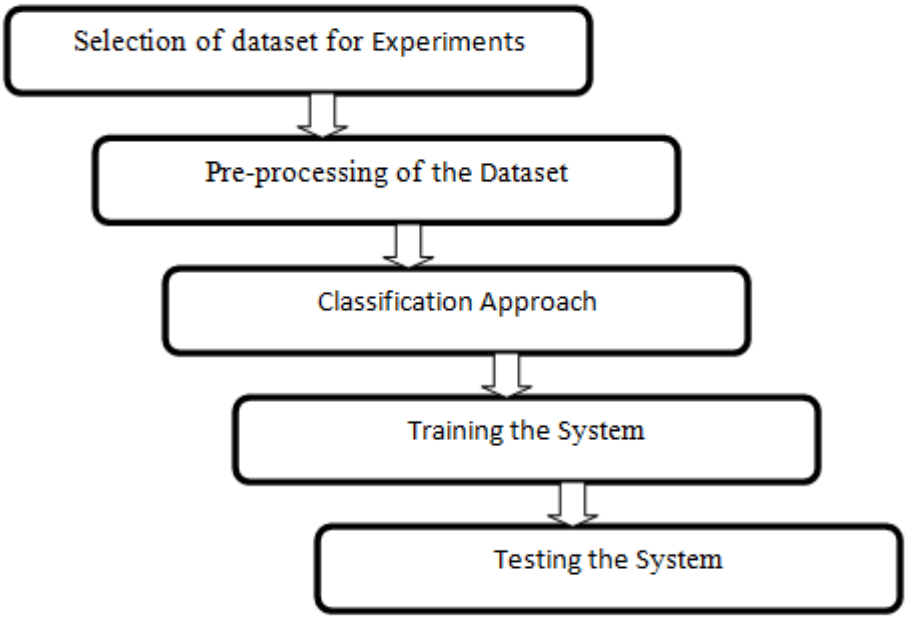


Figure 1.1 Methodology phases

1.5.1 Selection of Dataset for Experiments

The capability of the intrusion detection mechanism depends on the dataset. If the training data is more accurate then performance of trained system will be improved**.** So, the collecting of data for training and testing is a critical dilemma (Liu et al. 2007). Therefore, different issues will be discussed in obtaining or selecting dataset for experimental purpose in this research work. This phase of methodology will discuss which dataset is best and why?

1.5.2 Pre-processing of the Dataset

The selected dataset KDD cup consists of 41 raw features, which fall into different categories such as basic features and derived features (Liu et al. 2007). The basic features describe single network connection.

The *derived features* can be divided into *content-based features* and *traffic based features* (Ahmad et al. in 2008). The *content-based features* are derived using domain

knowledge and the traffic-based features are obtained by studying the sequential patterns between the connection records as well as the correlation between basic features. The second step is the pre-processing of data so that it can be given as input to my designed system. In this phase, this research work will apply PCA for features transformation and GA for the selection of optimal feature set for this proposed approach.

1.5.3 Classification Approach

After features selection, the next phase is determining the approach for classification. This is also another problem. For this purpose, this research work used the Support Vector Machine (SVM) and Multilayer Perceptron (MLP) for classification purpose due to their proven ability in classification (Sun et al. 2004) and (Pervez et al. 2007). Both approaches are applied and tested in different scenarios to compare their performance.

1.5.4 Training the System

The next phase is training the system. During training, both input patterns and desired outputs related to each input pattern are available. Further, the dataset is divided into three parts; (i) cross validation dataset, (ii) test dataset and (iii) training dataset so that better performance of the developed system may be achieved (Ahmad et al. 2007). Aim of the training is to minimize the error output produced by the system in comparison to the desired output. In order to achieve this goal, weights are updated by carrying out certain steps known as training.

1.5.5 Testing the System

After training, the weights of the system are frozen and performance of the system is evaluated. Testing the system involves two steps, which are verification step and generalization step. In verification step, system is tested against the data which is used in training. Aim of the verification step is to test how well trained system learned the

training patterns in the training dataset. In generalization step, testing is conducted with data which is not used in training. Aim of the generalization step is to measure generalization ability of the trained network (Principe et al. 2000). After training, the system only involves computation of the feed forward phase. For this purpose, this method used a production dataset that has input data but no desired data.

1.6 Research Activities

To achieve the goals, the research activities have been organized as follows:

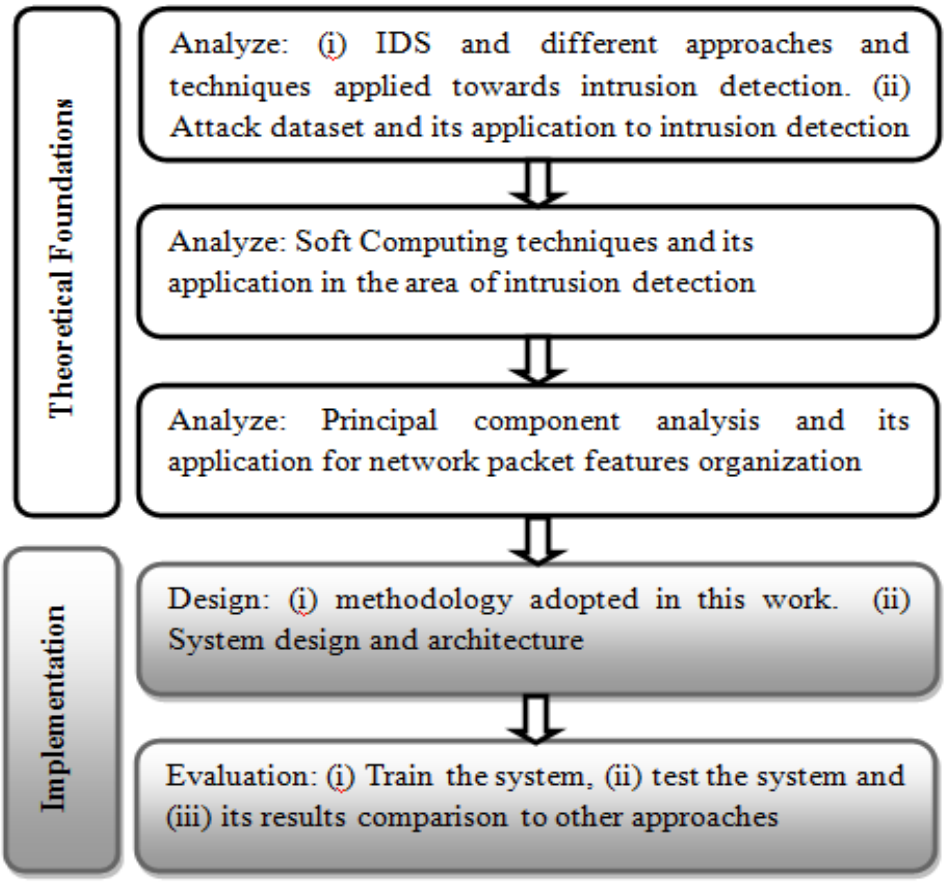


Figure 1.2 Research activities

1.7 Research Contributions

The research has significant positive impact on the performance of network intrusion detection systems, including addressing new open research issues. The main achievement is the development of intrusion detection mechanism that outperforms the existing approaches and has the capability to minimize the number of features and maximize the detection rates.

The major impact of the research span over the following areas:

- Performance optimization such as improving detection rate and reducing false alarms; false positive and false negative
- Minimizing training overheads and number of features for the intrusion detection approaches
- Prototype of the architectural framework
- Contribution to the existing intrusion detection technologies, knowledge and applications
- Help and guidance for the security implementers in the area of intrusion detection

Therefore, in this research work, an optimized intrusion detection mechanism using soft computing techniques; PCA, GA, SVM and MLP is proposed and implemented. This work uses the KDD cup dataset, which is considered a benchmark for evaluating security detection mechanisms. The performance of this approach was analyzed and compared with previous approaches. The outcomes demonstrate that proposed method provides an optimal intrusion detection mechanism that outperforms the existing approaches and has the capability to reduce the number of features and increase the detection rates.

1.8 Organization of the Thesis

After describing the motivation, objectives, research questions, methodology, research workflow and contributions of the work, the remainder of the thesis is organized as follows:

Chapter 2 introduces the basic ideas about intrusion detection system, its components, classification, and foundations and describes the related approaches that are mainly used in designing intrusion detection systems. The chapter thereafter describes the KDD cup dataset that is considered standard in the evaluation of intrusion detection mechanism. Then, the chapter highlights the issues in existing intrusion detection studies and reasoning of using KDD cup dataset. Next, the chapter discusses the background of soft computing techniques, its unique property, and future of soft computing. Then, it explains an overview of applied techniques in my research. The chapter then describes the background of neural networks, Support Vector Machine and Genetic Algorithm that are basic soft computing techniques. The chapter then discusses Principal Component Analysis that is applied for features transformation and organization in this research. Further, the chapter discusses the literature consulted in order to understand and investigate the research problem in the field of network intrusion detection. The chapter then summarizes and evaluates relevant research, and discusses the relationships between different works and describes how it relates to this research. The chapter finally discusses the issues in existing literatures and directs towards methodology that is adopted in this research work.

Chapter 3 explains the set of methods, techniques, and tools used in this research. The chapter, thereafter, demonstrates the workflow process of designing the system and architecture for network intrusion detection. Then, it describes different phases of adopted methodology like selection of dataset, pre-processing of dataset, determine the architecture, training and testing the designed system. Finally, the chapter provides directions towards implementation of the proposed model.

Chapter 4 describes the proposed model with its basic architecture in block diagram, and then details of each part or block of its main architecture. Then, it explains features description of the dataset used for experiments, feature transformation process using PCA and optimal features subset selection using GA. After this, the chapter describes the details of classification architectures with basic algorithms and mathematical foundation of multilayered perceptron model (MLP) and Support Vector Machine (SVM).

Then, the chapter explains system implementation, and the basic parameters used during training and testing. Finally, the chapter concludes with the contributions, and the chapter summary

Chapter 5 presents the experimental results obtained by the developed system. After that, it discusses the performance evaluation of the system by examining the number of false positives and false negatives that they generated during testing. Then, it discusses the results and their comparison with existing published results.

Chapter 6 concludes the work by summarizing the main contributions and findings of the study, the limitations of the study and some possibilities for future research and development.

The appendix ‘A’ provides a list of publications during this research work and appendix ‘B’ describes some terminologies used in this thesis.

CHAPTER 2

BACKGROUND AND LITERATURE REVIEW

2.1 Introduction

This chapter describes basic background of intrusion detection systems, their functional components, classification and characteristics in the following Sequence: (i) Staging of an overview of IDS foundations and analysis techniques to intrusion detection. (ii) Description of intrusion detection approaches such as anomaly, misuse detection and combined or hybrid approach. (iii) Description of attack dataset: KDD cup dataset, a standard dataset for evaluating security detection mechanisms that is used to train and test the proposed system. (iv) An overview of the soft computing, its properties, its applications in a variety of fields and its future usage. Then, the chapter describe technical background of soft computing techniques: Neural Networks (NN), Support Vector Machine (SVM), and Genetic Algorithm (GA) utilized in this research work. Later, the chapter describes the Principal Component Analysis (PCA) and its different steps towards feature transformation into PCA space. Further, the chapter presents the literature review to intrusion detection. Followed by, the presentation of the related study to intrusion detection using SVM, MLP, PCA and GA. Finally, the chapter describes a systematic review of related work and issues in the existing intrusion detection approaches.

2.2 Intrusion Detection Systems

This section describes some basic and fundamental concepts to Intrusion Detection Systems (IDSs). First work in the field of intrusion detection was performed by Anderson in the early 1980s. Anderson defines an intrusion as any unauthorized attempt to access, manipulate, modify, or destroy information, or to render a system

unreliable or unusable (Anderson 1980). Intrusions are caused by accessing the systems from Internet, by attackers, by authorized users of the systems who attempt to gain additional privileges for which they are not authorized, and authorized users who misuse the privileges given to them. IDSs are software or hardware products that monitor the system in question and try to detect any attack against the system. A truly secure system is still a dream, as there are always bugs in application programs, and also communication protocols always have vulnerabilities that can be exploited by attackers. In addition, passwords can be cracked, users can lose their passwords, and entire crypto system can be broken. As a result, security mechanisms (e.g. firewalls), which are deployed to protect the information system, may not be able to prevent all security breaches. IDSs are usually deployed along with the other security mechanisms, such as access control, authentication and firewalls, as a last defence line to improve security of the information system (Amini and Jalili 2005). The main goal of an IDS is to provide high rates of attack detection with very small rates of false alarms (Pervez et al. 2007). There are two important types of errors in intrusion detection:

False positives: False positives are the errors occurring when IDS flags a normal activity as an attack. Simply, false positives are false alarms.

False negatives: False negatives are the errors occurring when IDS fails to detect an ongoing attack.

2.3 IDS Functional Components

This section explains functional components of IDS. An IDS consists of three functional components as shown in Figure 2.1 (Bace and Mell 2001): (1) Information source that provides a stream or flow of event records, (2) Analysis engine that analyzes and classifies intrusions; and (3) Response component that generates reactions based on the output of the analysis engine.

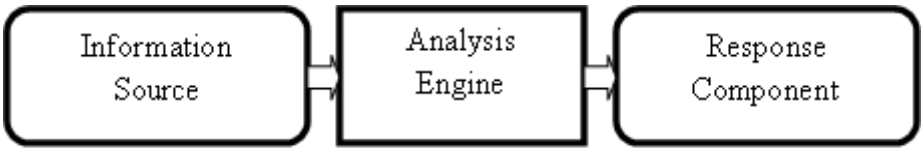


Figure 2.1 Functional components of IDS

2.3.1 Information Sources

The first component of an intrusion detection system is the data source, where input information, which will be analyzed, is collected. Input information can be audit trails, system logs or network packets.

2.3.2 IDS Analysis

Second component of IDS is the analysis engine, which analyse data from information source and classify it into normal or intrusive. The IDS analysis engines are classified into two categories such as misuse detection and anomaly detection.

2.3.3 Response Component

Response component is the third component of an IDS, where reaction to a detected attack is given. According to the response types, IDS can be either active or passive. An IDS is said to be active, if it actively reacts to the attack by taking corrective (closing holes) or proactive actions (logging out possible attackers, closing down services). If an IDS just generates alarms, it is said to be passive. Passive IDS responses provide information to system administrator who takes necessary actions based on that information.

2.4 Classification of Intrusion Detection Systems

This section describes two classes of IDS based on the type of the data source. According to the data sources used, IDSs can be classified into two categories; host-based IDSs and network-based IDSs (Jonsson et al. 2004).

2.4.1 Host-based IDS

Host-based IDSs observe activities within an individual computer system and work on information obtained from these activities (Ahmad et al. 2007). As target environment was mainframe environment, and all users were local to the system, initial researches in the field of IDSs were performed on the host-based IDSs. Host-based IDSs normally use two information sources, operating system audit trails, and system logs. Operating system trails are generally generated at the kernel level; hence they are more detailed and better protected than system logs. However, system logs are simpler and smaller than operating system trails; consequently, they can be understood more easily. Further, some benefits and drawbacks of this class of IDS are listed here.

Benefits

- As host-based IDSs monitor local activities, therefore they can detect attack that can not be detected by network-based IDSs.
- Information sources of the host-based IDSs are generally generated on a plaintext data, therefore they can successfully operate in an environment where network traffic is encrypted.
- Performance of the host-based IDSs is not affected by the topology of the network they operate in. They successfully operate on switched networks.

Drawbacks

- As host-based IDSs should be placed on every monitored host, it is harder to manage and configure host-based IDSs.
- Host-based IDSs run on the host targeted by attacks, and it may be disabled by a successful attack. For instance, certain denial-of-service attacks.
- As host-based IDSs can only see network packets received by its host, detection rate of host-based IDSs is poor in the case attacks are targeted to the entire network.

- Amount of information used by network-based IDSs can be huge; hence host–based IDSs are unable to generate good results.
- IDS may need extra storage on the system on which it is running.
- Host-based IDSs share the computing resources (e.g. CPU, main memory) with the monitored host. Consequently, they cost additional operational overheads and may affect the performance of the hosting computer.

2.4.2 Network-based IDS

As computing environments shifted from mainframe to the networks of workstations, studies on intrusion detection started to focus on attacks targeted to the network. Network attacks cannot be detected by examining operating system trails or system logs, or at least detection of network attacks by examining data sources on the host computer is not an easy task. As a result, network-based IDSs were developed, which capture network packets and search attacks in these network packets. Network based IDSs monitor activities on a network segment or switch, so that they can protect hosts connected to the monitored segment (Amini et al. 2006). Network-based IDSs generally consist of sensors which are placed at various points (such as at LAN and WAN backbones) in the network. Sensors collect network packets and feed them to the network-based IDS that classify them into normal or intrusive class (Cannady 1998). Further, some benefits and drawbacks of network based IDS are listed here.

Benefits

- A huge network can be monitored easily by using a few numbers of sensors, if sensors are placed at the critical parts of the network (as at hubs, routers or probes).
- Network-based IDSs are generally passive devices and do not affect the normal operation of the network.
- Network-based IDSs can be very secure against attacks, and even they can be made invisible to the attackers.

Drawbacks

- If monitored network is large or network traffic is high, it may be difficult to process all network packets.
- Problems arise when network-based IDSs are placed on a switched network. Most of the switches do not provide universal monitoring ports and this fact limits the monitoring ability of network-based IDS.
- Network-based IDSs can not analyze encrypted traffic. This is due to the fact that, the sensors analyze packet headers to determine source and destination addresses and type of data being transmitted, and analyze the packet payload to discover information in the data being transmitted.
- Distorted network packets may cause a network-based IDS to crash.

2.5 Characteristics of Intrusion Detection Systems

This section listed some characteristics of an ideal IDS (Ahmad et al. 2008).

- Intrusion detection systems should be automatic and reliable to monitor its front end and back end running programs.
- System should be fault tolerant in such a way that system crash should not affect its knowledge base where attack pattern are being stored for detecting intruders activities.
- System must be able to monitor itself to ensure subversion resistant.
- These systems should have less computational overhead to avoid degradation in system performance.
- System should have capability to detect deviation from normal behaviour.
- Its defence mechanism should be adaptable to new patterns as use patterns of every system are different.
- As system behaviour changes due to the addition of new applications in the system. Therefore, the IDS should have potential to adapt these changes and be able to detect any intrusion.
- Last but not the least its architecture should be like that intruders could not be successful to modify it for their desired activities.

2.6 Foundations of an Intrusion Detection System

This section describe the basic elements of IDS such as; metrics, models and profiles (Cannady 1998) and (Pervez et al. 2007). These elements are described here briefly.

2.6.1 Metrics

Any statistical intrusion detection methodology needs the use of a set of metrics. These metrics characterize the utilization of a variety of system resources (i.e., CPU usage, number of files accessed, number of login attempts). These metrics are usually one of three different types. The first metric, event counters identify the occurrences of a specific action over a period of time. These metrics may include the number of login attempts, the number of times that a file has been accessed, or a measure of the number of incorrect passwords that are entered. The second metric, time intervals identify the time interval between two related events. Each time interval compares the delay in occurrence of the same or similar event .An example of a time interval metric is the periods of time between a user's logins. The third metric is resource measurement that includes the expenditure of CPU time, number of records written to a database, or the number of files transmitted over the network. Keystroke dynamics is another method of quantifying a user's activities, which offers an effective measure of user identification. The concept involves the development of an electronic signature of a user based on their individual typing characteristics. These characteristics include; (i) typing speed, (ii) intervals in typing, (iii) number of errors, and (iv) the user's typing rhythm. These characteristics may be verified on login and monitored throughout a session. Complete intrusion detection mechanisms have been developed exclusively around the use of keystroke dynamics techniques.

2.6.2 Models

The selected metrics are then used in statistical models, which attempt to identify deviations from an established norm. The models, which have been most frequently used, include the operational model, average and standard deviation model, the multivaried model, the markovian model, and the time series model. The operational

model makes the assumption that an anomaly can be identified through a comparison of an observation with a predefined limit. This model is frequently used in the situations where a specific number of events, (i.e., failed logins), is a direct indication of a possible attack. The average and standard deviation model is based on the traditional statistical determination of the commonness of an observation based on its position relative to a specific confidence range. This model offers the advantage that it "learns" a user's behavior over time instead of requiring prior knowledge of the user's activities. As a result, the model can establish a foundation for the identification of potential anomalies for each user and identify potential problems from users who consistently behave in a manner, which would indicate normally the misuse of system resources. This is particularly useful in identifying what is normal for an individual user without relying on a comparison with other users. The multivaried model is built upon the average and standard deviation model. The difference between these two approaches is that the multivaried model is based on a correlation of two or more metrics. This model permits the identification of potential anomalies where the complexity of the situation requires the comparison of multiple parameters. The markovian model is used with the event counter metric to determine the normalcy of a particular event, based on the events that preceded it. The model characterizes each observation as a specific state and utilizes a state transition matrix to determine if the probability of the event is high (normal) based on the preceding events. This model is particularly useful when the sequence of activities is particularly important. The final model, the Time Series Model, attempts to identify anomalies by reviewing the order and time interval of activities on the network. If the probability of the occurrence of an observation is low, then the event is labeled as abnormal. This model provides the ability to evolve over time based on the activities of the users (Denault et al. 1994) and (Denault et al. 1994).

2.6.3 Profiles

These models are then used in the development of a variety of profiles, which attempt to map the nonintrusive activities of the system. The profiles serve to establish a baseline of a user's behavior, which can then be used for comparisons with the current observations. Profiles usually consist of specific characteristics, such as login

information, (i.e., frequency, origin, duration), program execution information, (i.e., frequency, CPU utilization), database access information, (i.e., tables accessed, data manipulation functions), and file access information (i.e., types of files accessed, created, or destroyed) (Jonsson et al. 2004).

2.7 Analysis Techniques for Intrusion Detection

This section explains analysis techniques those are fundamental components in an intrusion detection system, which examines the captured information into normal or intrusive class (Cannady 1998) and (Pervez et al. 2007). There are many approaches towards intrusion detection but this section describes an overview of five common approaches that have been used in numerous traditional intrusion detection mechanisms.

2.7.1 Statistical Analysis

This approach involves statistical comparison of specific events based on a predetermined set of criteria. The data was collected from the system and the network. This collected data was tested for attack analysis by statistical models.

The models which have been used most frequently, include the operational model, average and standard deviation model, the multivaried model, the markovian model, and the time series model. This was much laborious and time consuming work.

2.7.2 Rule Based System

This approach relies on sets of predefined rules, which are provided by an administrator, automatically created by the system, or both. Each rule is mapped to a specific operation in the system. The rules serve as operational preconditions, which are continuously checked in the audit record by the intrusion detection mechanism. If the required conditions of a rule are satisfied by user activity, the specified operation is executed. This approach was unable to detect novel intrusion. A frequent update of rules is required in this approach (Lunt 1989).

2.7.3 Expert Systems

The use of expert system techniques in intrusion detection mechanisms was a significant milestone in the development of effective intrusion detection system in information security systems. An expert system consists of a set of rules, which encode the knowledge of a human "expert". Unfortunately, expert systems require frequent updates by a system administrator to remain current. The lack of maintenance or update will degrade the security of the entire system by causing the system's users to be misleading into believing that the system is secure, even as one of the key components becomes increasingly futile over time (Mukherjee et al. 1994).

2.7.4 Pattern Recognition

In this approach, a series of penetration scenarios are coded into the system. Pattern recognition possesses a distinct advantage over anomaly and misuse detection methods in that it is capable of identifying attacks, which may occur over an extended period of time, or by multiple attackers working in concert. This approach is effective in reducing the need to review a potentially large amount of audit data.

2.7.5 Network Monitoring

This technique monitors network activity for indications of attacks. The greatest advantage of network monitoring mechanisms is its independence on audit data. Because this method does not require input from any operating system's audit trail, it can use standard network protocols to monitor heterogeneous sets of operating systems and hosts.

2.8 Approaches to Intrusion Detection

The following are the approaches being utilized to accomplish the desirable elements of an intrusion detection system (Fox et al. 1990). This section describe here these approaches briefly.

2.8.1 Anomaly Detection

Anomaly detection is the general category of intrusion detection, which works by identifying activities, which vary from established patterns for users, or groups of users. Anomaly detection typically involves the creation of knowledge bases which contain the profiles of the monitored activities (Khan et al. 2007). This approach has some benefits and drawbacks those are listed below.

Benefits

- As any significant deviation from normal profile will be flagged as anomalous, anomaly detectors can detect unknown attacks.
- Anomaly detectors do not require constant updating of rules or signatures of novel intrusion.
- Anomaly detectors can produce information that can in turn be used to define signatures for misuse detectors.

Drawbacks

- The high false positive rate is the main drawback of the anomaly IDSs. This is due to the fact that, the normal profile of a system cannot be fully learned and/or behavior of users or programs may change over time.
- In order to build normal profile of a system, system in question should be monitored and information should be collected, which in turn will be used to draw normal behaviour of the system. However, if the collected information

contains attacks, intrusive behavior will be a part of the normal profile, and in future, these attacks will go undetected.

- Anomaly detection approaches need extensive data sets to build profile of the system.

2.8.2 Misuse Detection

The second general approach to intrusion detection is misuse detection. This technique involves the comparison of a user's activities with the known behaviors of attackers attempting to penetrate a system. Misuse detection also utilizes a knowledge base of information (Mukherjee et al. 1994). This approach has some benefits and drawbacks those are listed below.

Benefits

- Misuse IDSs can detect intrusion with a certain degree of certainty. Misuse detectors are very effective in detecting attacks without giving high false alarm rates.
- Misuse IDSs can detect all intrusions whose signatures are known.
- Misuse IDSs are easy to implement (state machine, signature analysis) and deploy (no need to form a profile of the system).

Drawbacks

- Detection ability of misuse detectors is limited to signatures that they posses. A new intrusion or even a variation of a known intrusion may be undetected. So misuse IDS require regular updates of signatures in order to remain current.
- The process of developing a new attack signature is time consuming.

2.8.3 Combined Anomaly and Misuse Detection

Research has also been conducted into intrusion detection methodologies, which combine the anomaly detection approach and the misuse detection approach. The combined approach permits a single intrusion detection system to monitor for indications of external and internal attacks (Pervez et al. 2007).

All current intrusion detection systems make four statements about the systems that they are designed to protect:

- Activities taken by system users, either authorized or unauthorized, can be monitored.
- It is possible to identify those actions, which are indications of an attack on a system
- Information obtained from the intrusion detection system can be utilized to enhance the overall security of the network.
- A fourth element, which is desirable from any intrusion detection mechanism, is the ability of the system to make an analysis of an attack in real-time.

2.9 Attack Dataset

This section explains attack dataset that has used in this research work. The defense advanced research projects agency (DARPA) project was prepared and executed by the Massachusetts Institute of Technology (MIT) Lincoln Laboratory, USA (Pervez et al. 2007), (Bankovic et al. 2009). One of the reasons for choosing this dataset is that the dataset is standard. This will make it easy to compare the results of this work with other similar works. Another reason is that it is difficult to get another data set, which contains so rich a variety of attacks as the one used here.

There are 41 features, which fall into different categories such as basic features and derived features. The basic features describe single network connections. The derived features can be divided into content-based features and traffic based features.

The content-based features are derived using domain knowledge and the traffic-based features are obtained by studying the sequential patterns between the connection records as well as the correlation between basic features.

In order to create the feature set, the raw tcpdump data has been pre-processed into connection records. The basic features are directly obtained from the connection records. The derived features fall into two groups; (i) Content-based features and (ii) Traffic based features. Table 2.1 explains 41 features of KDD cup dataset in terms of feature number, name of feature and type of feature. These features form a record that represents an attack or normal activity. Table 2.2 shows connection-based features, Table 2.3 shows content-based features and Table 2.4 shows nine time-based features from the KDD cup dataset

Table 2.1 Features of KDD cup dataset

Feature #	Name of feature	Type of feature
1	DURATION	CONTINUOUS
2	PROTOCOL_TYPE	SYMBOLIC
3	SERVICE	SYMBOLIC
4	FLAG	SYMBOLIC
5	SRC_BYTES	CONTINUOUS
6	DST_BYTES	CONTINUOUS
7	LAND	SYMBOLIC
8	WRONG_FRAGMENT	CONTINUOUS
9	URGENT	CONTINUOUS
10	HOT	CONTINUOUS
11	NUM_FAILED_LOGINS	CONTINUOUS
12	LOGGED_IN	SYMBOLIC
13	NUM_COMPROMISED	CONTINUOUS
14	ROOT_SHELL	CONTINUOUS
15	SU_ATTEMPTED	CONTINUOUS

Feature #	Name of feature	Type of feature
16	NUM_ROOT	CONTINUOUS
17	NUM_FILE_CREATIONS	CONTINUOUS
18	NUM_SHELLS	CONTINUOUS
19	NUM_ACCESS_FILES	CONTINUOUS
20	NUM_OUTBOUND_CMDS	CONTINUOUS
21	IS_HOST_LOGIN	SYMBOLIC
22	IS_GUEST_LOGIN	SYMBOLIC
23	COUNT	CONTINUOUS
24	SRV_COUNT	CONTINUOUS
25	SERROR_RATE	CONTINUOUS
26	SRV_SERROR_RATE	CONTINUOUS
27	RERROR_RATE	CONTINUOUS
28	SRV_RERROR_RATE	CONTINUOUS
29	SAME_SRV_RATE	CONTINUOUS
30	DIFF_SRV_RATE	CONTINUOUS
31	SRV_DIFF_HOST_RATE	CONTINUOUS
32	DST_HOST_COUNT	CONTINUOUS
33	DST_HOST_SRV_COUNT	CONTINUOUS
34	DST_HOST_SAME_SRV_RATE	CONTINUOUS
35	DST_HOST_DIFF_SRV_RATE	CONTINUOUS
36	DST_HOST_SAME_SRC_PORT_RATE	CONTINUOUS
37	DST_HOST_SRV_DIFF_HOST_RATE	CONTINUOUS
38	DST_HOST_SERROR_RATE	CONTINUOUS
39	DST_HOST_SRV_SERROR_RATE	CONTINUOUS
40	DST_HOST_RERROR_RATE	CONTINUOUS
41	DST_HOST_SRV_RERROR_RATE	CONTINUOUS

Table 2.2 Connection-based feature

No.	Feature name	Description
1	DURATION	Length (number of seconds) of the connection
2	PROTOCOL_TYPE	Type of the protocol, e.g. tcp, udp, etc.
3	SERVICE	Network service on the destination
4	SRC_BYTES	Number of data bytes from source to destination
5	DST_BYTES	Number of data bytes from destination to source
6	FLAG	Normal or error status of the connection
7	LAND	1 if connection is from/to the same host/port
8	WRONG_FRAGMENT	Number of `wrong' fragments
9	URGENT	Number of urgent packets

Table 2.3 Content-based features

No.	Feature name	Description
10	HOT	Number of hot indicators
11	NUM_FAILED_LOGINS	Number of failed login attempts
12	LOGGED_IN	1 if successfully logged in; 0 otherwise
13	NUM_COMPROMISED	Number of compromised conditions
14	ROOT_SHELL	1 if root shell is obtained; 0 otherwise
15	SU_ATTEMPTED	1 if su root command attempted; 0 otherwise
16	NUM_ROOT	Number of root accesses
17	NUM_FILE_CREATIONS	Number of file creation
18	NUM_SHELLS	Number of shell prompts
19	NUM_ACCESS_FILES	Number of operations on access control files
20	NUM_OUTBOUND_CMDS	Number of outbound commands
21	IS_HOST_LOGIN	1 if the login belongs to the host else 0
22	IS_GUEST-LOGIN	1 if the login belongs to the guest else 0

Table 2.4 Time-based features

No.	Feature name	Description
23	COUNT	Number of connections to the same host as the current connection in the past two seconds
Note: The following features refer to these same-host connections.		
24	SERROR_RATE	% of connections that have ``SYN'' errors
25	RERROR_RATE	% of connections that have ``REJ'' errors
26	SAME_SRV_RATE	% of connections to the same service
27	DIFF_SRV_RATE	% of connections to different services
28	SRV_COUNT	number of connections to the same service as the current connection in the past two seconds
Note: The following features refer to these same-service connections.		
No.	Feature name	Description
29	SRV_SERROR_RATE	% of connections that have ``SYN'' errors
30	SRV_RERROR_RATE	% of connections that have ``REJ'' errors
31	SRV_DIFF_HOST_RATE	% of connections to different hosts
32	DST_HOST_COUNT	Number of connection to host
33	DST_HOST_SRV_COUNT	Number of services requested to host
34	DST_HOST_SAME_SRV_RATE	% of connection with same service
35	DST_HOST_DIFF_SRV_RATE	% of connection with different services
36	DST_HOST_SAME_SRC_PORT_RATE	% of connection using the same source port
37	DST_HOST_SRV_DIFF_HOST_RATE	% of connection with same service but to different host

No.	Feature name	Description
38	DST_HOST_SERROR_RATE	% of connection that have SNY error
39	DST_HOST_SRV_SERROR_RATE	% of connection with same service that have SYN error
40	DST_HOST_RERROR_RATE	% of connection that have REJ errors
41	DST_HOST_SRV_RERROR_RATE	% of connection with same service that have REJ errors

The description of these types of attacks requires some domain knowledge and cannot be done only based on information available in the packet header. Most of these attacks are R2L and U2R attacks. Traffic based features have been computed automatically. They are effective for the detection of DOS and probe attacks. A list of the computer attacks is described briefly that are considered in this research work.

2.9.1 Denial of Service (DOS) Attacks

DOS is a type of attack that aims to make an organization's services or resources unavailable for an indefinite amount of time by flooding it with useless traffic (Kim et al. 2005). The examples of DOS attacks are given as follows.

- *Ping of Death (pod):* It makes the victim host unavailable by sending it oversized internet control message protocol (ICMP) packets as ping requests.
- *Back:* It is a denial of service attack against apache web servers. The attacker sends requests containing many front slashes. The processing of which is time consuming.
- *Land:* Spoofed synchronization (SYN) packet sent to the victim host resulting in that host repeatedly synchronizing with itself.
- *Smurf:* A broadcast of ping requests with a spoofed sender address which results in the victim being bombarded with a huge number of ping responses.
- *Neptune:* The attacker half opens a number of transmission control protocol (TCP) connections to the victim host making it impossible for the victim host to accept new TCP connections from other hosts.

- *Teardrop:* Confuses the victim host by sending it overlapping internet protocol (IP) fragments: overlapping IP fragments are incorrectly dealt with by some older operating systems.

Each feature of KDD record is separated by semicolon. A sample of DOS attack and its normal pattern is shown in Table 2.5.

Table 2.5 A DOS attack and its normal pattern

Attack pattern	Normal pattern
0;18;21;20;1480;0;0;1;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;1;1;0.00;0.00;0.00;0.00;1.00;0.00;0.00;1;1;1.0;0.00; 1.00; 0.00; 0.00; 0.00; 0.00; 0.00; 1; 0	0;18;21;20;1480;0;0;1;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;2;4;0.00;0.00;0.00;0.00;1.00;0.00;0.50;2;4;1.00;0.00;1.00;0.50;0.00;0.00;0.00;0.00;0;1

2.9.2 Probing Attacks

Probing attack involves discovering the algorithms and parameters of the recommender system itself (Kim et al. 2005). It may be necessary for an intruder to acquire this knowledge through interaction with the system itself. The examples of probing attacks are given as follows.

- *Ipsweep:* It probes the network to discover available services on the network.
- *Portsweep:* It probes a host to find available services on that host.
- *Nmap:* It is a complete and flexible tool for scanning a network either randomly or sequentially.
- *Satan:* It is an administration tool; it gathers information about the network. This information can be used by an attacker.

Each record of KDD consists of features that are separated by semicolon. A sample of probing attack and its normal pattern is shown in Table 2.6

.

Table 2.6 A Probing attack and its normal pattern

Attack pattern	Normal pattern
0;6;19;34;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;1;1;0.00;0.00;1.00;1.00;1.00;0.00;0.00;255;1;0.00;1.00;1.00;0.00;0.00;0.00;1.00;1.00;1;0	0;6;22;20;259;17289;0;0;0;0;0;1;0;0;0;0;0;0;0;0;0;0;2;2;0.00;0.00;0.00;0.00;1.00;0.00;0.00;20;255;1.00;0.00;0.05;0.03;0.00;0.00;0.00;0.00;0;1

2.9.3 Remote to Local (R2L) Attacks

R2L attack involves unauthorized access from a remote machine (Ahmad et al. 2007). The examples of remote to local attacks are given below:

- *Imap:* It causes a buffer overflow by exploiting a bug in the authentication procedure of the imap server on some versions of LINUX. The attacker gets root privileges and can execute an arbitrary sequence of commands.
- *Ftpwrite:* This attack exploits a misconfiguration affecting write privileges of anonymous accounts on a file transfer protocol (FTP) server. This allows any ftp user to add arbitrary files to the FTP server.
- *Phf:* This is an example of badly written common gateway interface (CGI) scripts that is distributed with the apache server. Exploiting this flaw allows the attacker to execute codes with the http privileges.
- *Warezmaster:* This attack is possible in a situation where write permissions are improperly assigned on a FTP server. When this is the case, the attacker can upload copies of illegal software that can then be downloaded by other users.
- *Warezclient:* This attack consists in downloading illegal software previously uploaded during a warezmaster attack.

Every feature of KDD record is separated by semicolon. A sample of R2L attack and its normal pattern is shown in Table 2.7.

Table 2.7 A R2L attack and its normal pattern

Attack pattern	Normal pattern
4;6;29;20;30;93;0;0;0;0;0;1;0;0;0;0;0;0;0;0;0;0;1;1;0.00;0.00;0.00;0.00;1.00;0.00;0.00;255;248;0.97;0.01;0.00;0.00;0.00;0.00;0.01;0.00;1;0	0;6;24;20;1342;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;6;6;0.00;0.00;0.00;0.00;1.00;0.00;0.00;171;63;0.37;0.02;0.37;0.00;0.01;0.00;0.00;0.00;0;1

2.9.4 User to Root (U2R) Attacks

U2R attack involves unauthorized access to local super user privileges by a local unprivileged user (Ahmad et al. 2007). A sample of U2R attack and its normal pattern is shown in Table 2.8. The examples of such attacks are described as follows.

Loadmodule: This attack exploits a flaw in how SUNOS 4.1 dynamically load modules. This flaw makes it possible for any user of the system to get root privileges.

Perl: Exploits a bug in some practical extraction and report language (PERL) implementations on some earlier systems. This bug consists in these PERL implementations improperly handling their root privileges. This leads to a situation where any user can obtain root privileges.

Buffer Overflow: It consists in overflowing input buffers in order to overwrite memory locations containing security relevant information. Table 2.8 shows 38 features of the dataset that are separated by semicolon.

Table 2.8 A U2R attack and its normal pattern

Attack pattern	Normal pattern
0;17;19;20;105;147;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;1;1;0.00;0.00;0.00;0.00;1.00;0.00;0.00;255;253;0.99;0.01;0.00;0.00;0.00;0.00;0.00;0.00;0;1	0;17;19;20;105;147;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;1;1;0.00;0.00;0.00;0.00;1.00;0.00;0.00;255;253;0.99;0.01;0.00;0.00;0.00;0.00;0.00;0.00;0;1

2.9.5 Other Attacks

There are some other types of attacks; guessing passwords, making use of spy programs, and making use of rootkit and multihop attack (Ahmad et al. 2008). A sample of such attacks and their normal patterns are shown in Table 2.9.

Table 2.9 A sample of another attack and its normal pattern

Attack pattern	Normal pattern
1;6;27;20;2599;293;0;0;0;0;0;1;0;0;0;0;0;0;0;0;0;3;3;0.00;0.00;0.00;0.00;1.00;0.00;0.00;255;246;0.96;0.04;0.00;0.00;0.00;0.00;0.04;0.00;1;0	0;17;23;20;44;115;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;1;8;0.00;0.00;0.00;0.00;1.00;0.00;0.38;255;251;0.98;0.01;0.00;0.00;0.00;0.00;0.00;0.00;0;1

2.10 Soft Computing

This section provides brief information to Soft Computing (SC) in general and explains applied SC techniques in specific. SC refers to algorithms that are able to deal with uncertainty and incomplete information and that are still capable of discovering approximately good solutions to complex computational problems, and doing so faster from a computational perspective. These algorithms include neural networks, evolutionary computing (genetic algorithms and genetic programming), Support Vector Machines and fuzzy logic etc. Actually, the role model for soft computing is the human mind. The soft computing is tolerant of imprecision, uncertainty, partial truth, and approximation while hard computing shows precision, certainty, and rigor (Saad 2008) and (Eiben and Smith 2003). The guiding principle of soft computing is exploiting the tolerance for imprecision, uncertainty, partial truth, and approximation to achieve flexibility, robustness and low solution cost (Jirapummin et al. 2002) and (Bäck 1996). Mostly applications of soft computing techniques are those areas where imprecision, or inaccuracy is acceptable, and the solution is tolerable of imprecision.

2.10.1 Unique Property of Soft Computing

The unique property of soft computing is described below (Michalewicz 1996).

- The soft computing learns from experimental data.
- The soft computing techniques derive their power of generalization from approximating to produce outputs from previously unseen inputs by using outputs from previous learned inputs.
- The generalization is done usually in a high dimensional space.

2.10.2 Applications of Soft Computing

The soft computing has been used in different areas. But few applications of soft computing are listed here (Bebis et al. 2000).

- Hand written recognition
- Automotive systems and manufacturing
- Image processing and data compression
- Architecture
- Decision-support systems
- Power systems
- Neuro fuzzy systems
- Fuzzy logic control etc.

2.10.3 Future of Soft Computing

Soft computing is playing an important role in science and engineering, but sooner or later, its influence may extend much farther. It represents a significant paradigm shift in the aims of computing which reflects the fact that the human mind, unlike current computers, possesses a remarkable ability to store and process information which is imprecise, and uncertain (Verikas et al. 2010) and (Jang et al. 1997).

2.11 Overview of Soft Computing Techniques

The primary techniques of soft computing (SC) are fuzzy logic (FL), neural networks (NN), SVM, evolutionary computation (EC), and machine learning (ML) and probabilistic reasoning (PR) (Jang et al. 1997). However, this section describes an overview of applied techniques in this research work such as neural networks, SVMs and GAs.

2.11.1 Neural Networks (NN)

Neural network is an information processing model that is inspired by the biological nervous systems, such as the brain, process information. The main element of this model is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working in combination to solve particular problems (Hammerstrom 1993).

Neural networks (NNs), like people, learn by example. Neural network is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons. This is true of artificial neural networks or neural networks as well (Yu et al. 2005) and (Fausett 2009).

2.11.1.1 *Neurobiological Background*

The nervous system of living organisms is a structure consisting of many elements or processing units working in parallel fashion and in connection with one another. This structure (neural cell of the brain) is known as neuron that is developed in 1836. The structure of biological neuron is shown in Figure 2.2 as ascribed in the literature (Fausett 2009).

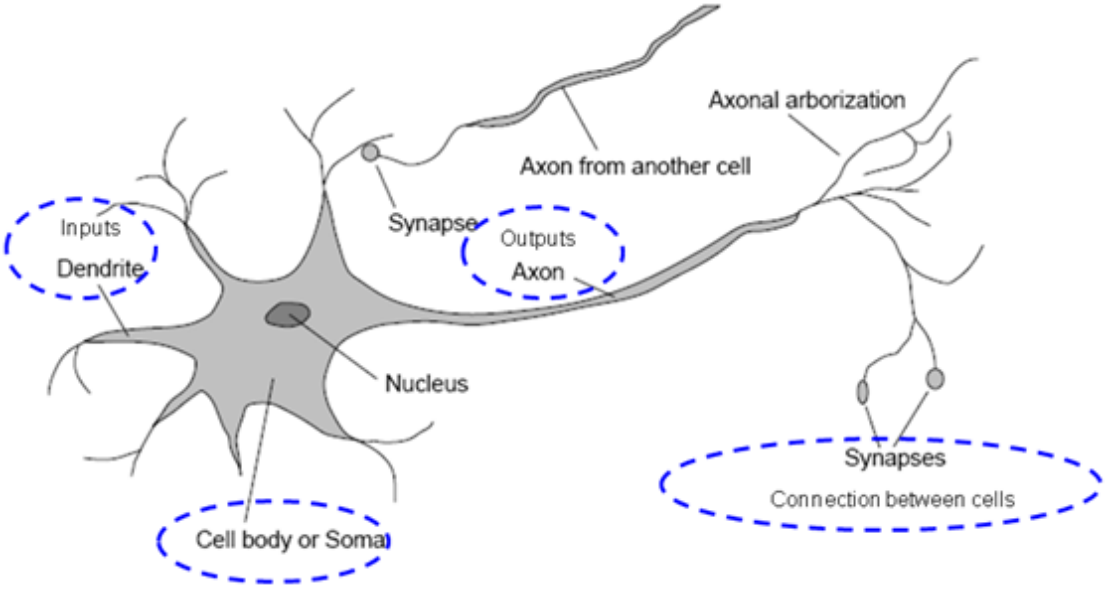


Figure 2.2 Structure of biological neuron

This is a result worth of the Nobel Prize (1906). The neuron has many-inputs and one-output units as shown in Figure 2.2 (Fausett 2009). The output can be excited or not excited, just two possible choices (like a flip-flop). The signals from other neurons are summed together and compared against a threshold to determine if the neuron shall excite ("fire"). The input signals are subject to attenuation in the synapses which are junction parts of the neuron. The concept of synapse was introduced in 1897.The next important step was to find that the synapse resistance to the incoming signal can be changed during a "learning" process (1949). If an input of a neuron is causing the neuron to fire repeatedly and persistently, a metabolic change happens in the synapse of that particular input to reduce its resistance (Fausett 2009).

2.11.1.2 Artificial Neuron

The first artificial neuron was produced in 1943 by the neurophysiologist Warren McCulloch and the logician Walter Pits. But the technology available at that time did not allow them to do too much (Jirapummin et al. 2002). The structure of ANN is shown in Figure 2.3.

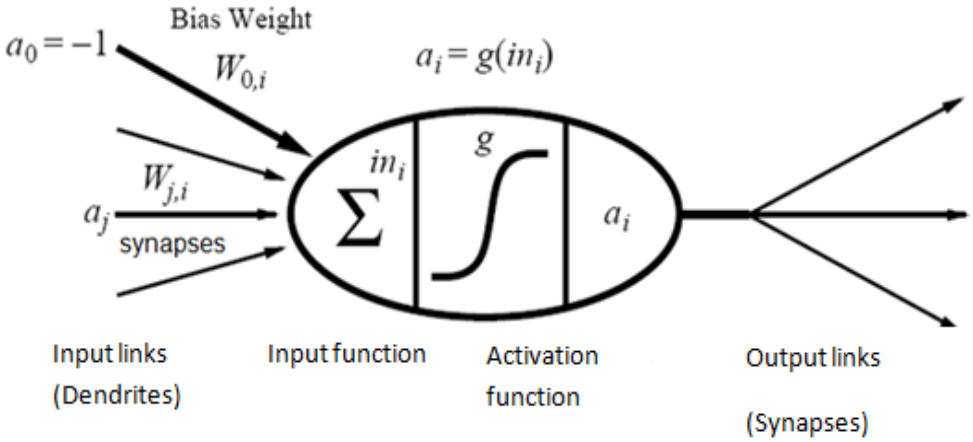


Figure 2.3 Structure of artificial neuron

A simple mathematical expression can be expressed of a neuron model as given in equation (2.1). The unit's output activation is given in Equation 2.1.

$$a_i = g(\sum_{j=0}^{n} W_{j,i}\, a_j) \qquad (2.1)$$

Where a_j is the output activation of unit j and W_{ji} is the weight on the link from unit j to this unit.

An artificial neuron is a processing element with many inputs and one output. The neuron has two types of operation; one is the training phase and the other is using or testing phase. In the training phase, the neuron can be trained to fire (or not), for specific input patterns or exemplars. In the using or testing phase, when a taught input pattern or exemplar is detected at the input, its associated output becomes the current output. If the input does not belong in the taught list of input patterns, the firing rule is used to determine whether to fire or not (Principe et al. 2000).

2.11.1.3 Architecture of Neural Networks

This section explains two types of neural networks. One is feed-forward and other is feedback.

a). Feed-forward Networks

Feed-forward ANNs allow signals to travel one way only; from input to output. There is no feedback (loops) i.e. the output of any layer does not affect that same layer. Feed-forward ANNs tend to be straightforward networks that associate inputs with outputs. They are used extensively in pattern recognition (Yu et al. 2005) and (Ahmad et al. 2007). This type is also referred to as bottom-up or top-down that is shown in Figure 2.4.

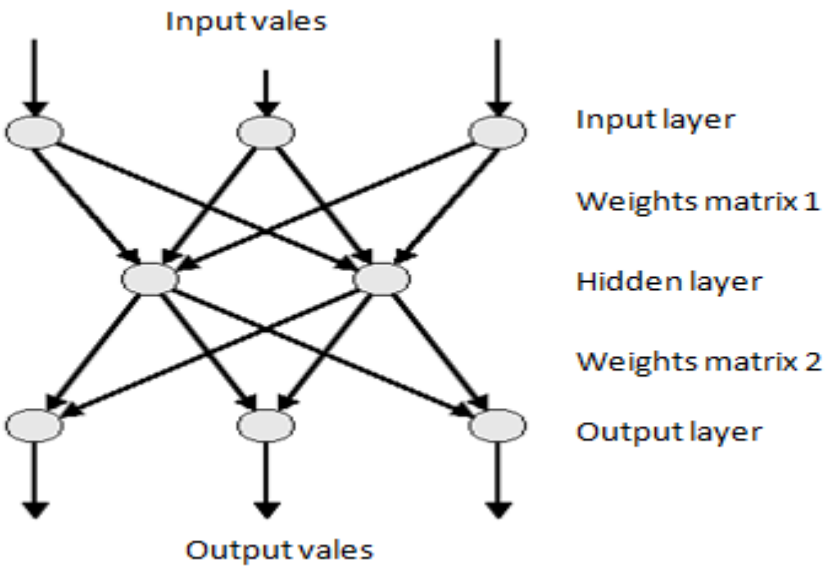


Figure 2.4 Feed -forward ANN

b). Feedback Networks

Feedback networks can have signals traveling in both directions by introducing loops in the network. Feedback networks are dynamic; their state is changing continuously until they reach an equilibrium point. They remain at the equilibrium point until the input changes and a new equilibrium needs to be found. It is also known as interactive

or recurrent (Ahmad et al. 2007) and (Ahmad et al. 2008). This type of neural network is shown in Figure 2.5.

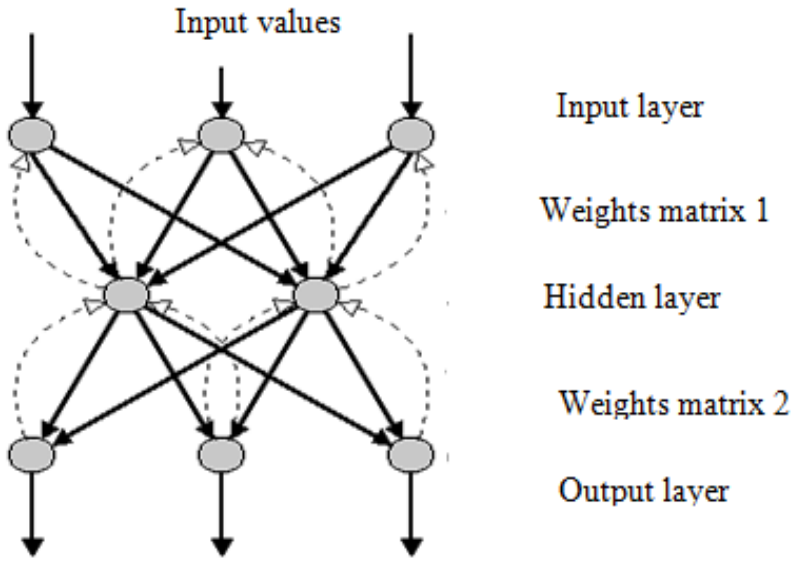


Figure 2.5 Feedback ANN

2.11.1. 4 Characteristics of Neural Networks

The neural network is popular due to the following characteristics (Fausett 2009).

- The NNs demonstrate capabilities, that is, they can map input patterns to their associated output patterns.
- The NNs learn by examples. They can be trained with known examples of a problem before testing.
- The NNs possess the capability to generalize. Thus, they can predict new outcomes from past trends.
- The NNs are robust systems and are fault tolerant. They can recall full pattern from incomplete, partial or noisy patterns.
- The NNs can process information in parallel, at high speed, and in a distributed manner.

2.11.1.5 Learning Methods

There are two main learning methods in neural networks; supervised and unsupervised. This section describes them briefly.

a). Supervised Learning

It includes an external teacher, so that each output unit is told what its desired response to input signals ought to be. During the learning process global information may be required. Models of supervised learning include; (i) error-correction learning, (ii) reinforcement learning, and (iii) stochastic learning. An imperative issue regarding supervised learning is the problem of error convergence, i.e. the minimization of error between the desired and computed unit values. The aim is to determine a set of weights, which minimizes the error. One well-known method, which is common to many learning models, is the least mean square (LMS) convergence (Browne 2000).

b). Unsupervised Learning

It uses no external teacher and is based upon local information only. It is also known as self-organization, in the sense that it self-organizes data presented to the network and detects their evolving collective properties. Paradigms of unsupervised learning are hebbian lerning and competitive learning (Sandhya 2009).

2.11.1.6 Transfer Function

The behavior of an ANN (Artificial Neural Network) depends on both the weights and the input-output function (transfer function) that is specified for the units. This function typically falls into one of three categories (Sandhya 2009).

a) Linear
b) Threshold
c) Sigmoid

a) *Linear units:* The output activity is proportional to the total weighted output.
b) *Threshold units:* The output is set at one of two levels, depending on whether the total input is greater than or less than some threshold value.
c) *Sigmoid units:* The output varies continuously but not linearly as the input changes. Sigmoid units bear a greater similarity to real neurons than do linear or threshold units, but all three must be considered rough approximations.

2.11.1.7 Neural Network Analysis

An artificial neural network consists of a collection of processing elements that are highly interconnected and transform a set of inputs to a set of desired outputs. The result of the transformation is determined by the characteristics of the elements and the weights associated with the interconnections among them. By modifying the connections between the nodes the network is able to tune to the desired outputs (Bankovic et al. 2007). Unlike expert systems, which can provide the user with an absolute answer if the characteristics, which are analyzed precisely, match those, which have been coded in the rule base, a neural network conducts an analysis of the information and provides a probability estimate that the data matches the characteristics, which it has been trained to recognize. While the probability of a match determined by a neural network can be 100 %, the accuracy of its decisions relies totally on the experience the system gains in examining examples of the stated problem. The neural network gains the experience initially by training the system to identify correctly preselected examples of the problem. The response of the neural network is analyzed and the configuration of the system is clarified until the neural network's analysis of the training data reaches an agreeable level. In addition to the initial training period, the neural network also gains experience over time as it conducts analyses on data related to the problem (Hammerstrom 1993) ,(Cannady 2000a) and (Pervez et al. 2007).

2.11.1.8 Neural Network Intrusion Detection Systems

A limited amount of research has been conducted on the application of neural networks to detecting computer intrusions. Artificial neural networks offer the potential to resolve a number of the problems faced by the other current approaches to intrusion detection. Artificial neural networks are alternatives. Neural networks were specifically proposed to identify the typical characteristics of system users and identify statistically significant variations from the user's established behavior (Fu 1992). Artificial neural networks have also been designed for use in the detection of computer viruses. They were proposed as statistical analysis approaches in the detection of viruses and malicious software in computer networks. The neural network architecture, which was selected, was SOM a self-organizing feature map which uses a single layer of neurons to represent knowledge from a particular domain in the form of a geometrically organized feature map. The proposed network was designed to learn the characteristics of normal system activity and identify statistical variations from the norm that may be an indication of a virus (Denault et al. 1994).

2.11.1.9 Advantages of Neural Network-based IDS

The first advantage of a neural network in the detection of instances of misuse would be the flexibility that the network would provide. A neural network would be capable of analyzing the data from the network, even if the data is incomplete or distorted. Similarly, the network would possess the ability to conduct an analysis with data in a non-linear fashion. Further, because some attacks may be conducted against the network in a coordinated attack by multiple attackers, the ability to process data from a number of sources in a non-linear fashion is especially important. The built in speed of neural networks is another benefit of this approach. Because the protection of computing resources requires the timely identification of attacks, the processing speed of the neural network could enable intrusion responses to be conducted before permanent damage occurs to the system. Because the output of a neural network is expressed in the form of a probability, the neural network provides a predictive capability to the detection of instances of misuse. A neural network-based misuse detection system would identify the probability that a particular event, or series of events, was indicative of an attack against the system. As the neural network gains

experience it will enhance its aptitude to determine where these events are likely to occur in the attack process. This information could then be used to generate a series of events that should occur if this is in fact an intrusion attempt. By tracking the subsequent occurrence of these events the system would be capable of improving the analysis of the events and possibly conducting defensive measures before the attack is successful. However, the most important advantage of neural networks in intrusion detection is the ability of the neural network to "learn" the characteristics of intrusion attacks and identify instances that have been observed before by the network. The probability of an attack against the system may be estimated and a potential threat flagged whenever the probability exceeds a specified threshold (Fox et al. 1990), (James 1997) and (Ahmad et al. 2008).

2.11.1.10 Disadvantages of Neural Network-based IDS

There are two primary reasons why neural networks have not been applied to the problem of misuse detection in the past. The first reason relates to the training requirements of the neural network. Because the ability of the artificial neural network to identify indications of an intrusion is completely dependent on the accurate training of the system, the training data and the training methods that are used are critical. The training routine requires a very large amount of data to ensure that the results are statistically accurate. The training of a neural network for intrusion detection purposes may require thousands of individual attacks sequences, and this quantity of sensitive information is difficult to obtain (Pervez et al. 2007) and (Cannady 2000b). However, the most significant disadvantage of applying neural networks to intrusion detection is the "black box" nature of the neural network. Unlike expert systems, which have hard-coded rules for the analysis of events, neural networks adapt their analysis of data in response to the training which is conducted on the network. The connection weights and transfer functions of the various network nodes are usually frozen after the network has achieved an acceptable level of success in the identification of events. While the network analysis is achieving a sufficient probability of success, the basis for this level of accuracy is not often known.

The "Black Box Problem" has overwhelmed neural networks in a number of applications. This is an on-going area of neural network research (Cannady 2000a) and (Fox et al. 1990).

In the last few years, the intrusion detection field has developed significantly and therefore many IDSs have been developed. The initial IDSs were anomaly detection tools but now, most of the commercial IDSs are misuse detection tools. IDSs have become a need, as number of computer and network systems increased seriously. The purpose of this research is to propose and analyze the applicability of soft computing in the field of intrusion detection. The proposed network based intrusion detection system is network-based, because it uses network data to determine whether an intrusion has taken place or not.

2.11.2 Support Vector Machine (SVM)

SVM was introduced in computational learning theory conference (COLT-92) in 1992. Support vector machines (SVMs) are a set of related supervised learning methods used for classification and regression (Cortes and Vapnik 1995).

SVM can be applied to the problem of traffic classification in computer network systems. This technique is suitable for solving classification problems with high dimensional feature space and small training set size. Although the basic technique was conceived for binary classification, several methods for single and multi-class problems have been developed. As a supervised method, it relies on two phases: training and testing. The algorithm acquires knowledge about the classes by examining the training set during the training phase. During the evaluation or testing phase, a classification method examines the evaluation or testing set and associates its members to the classes that are available. During the training phase, the target of the algorithm is the estimation of boundaries between the classes described by the samples in the training sets. To describe the method with a very simple example one can think of a two class problem where a single regular surface perfectly divides the features space in two regions, each one fully representative of the corresponding class (Este et al. 2009). There can be some issues noticed with neural networks. Some of them are having many local minima and also finding how many neurons might be

needed for a task is another issue, which determines whether optimality of that NN is reached. Another thing to note is that even if the neural network solutions used tends to converge, this may not result in a unique solution. SVM performs better in term of not over generalization (Mitchell 1997).

Figure 2.6 shows two different types of data and there are many hyper planes, which can classify it. However, which one is better? Which of the linear separators is optimal? Therefore, the solution of this problem of selecting suitable hyperplane is SVM (Smith and Gales 2002).

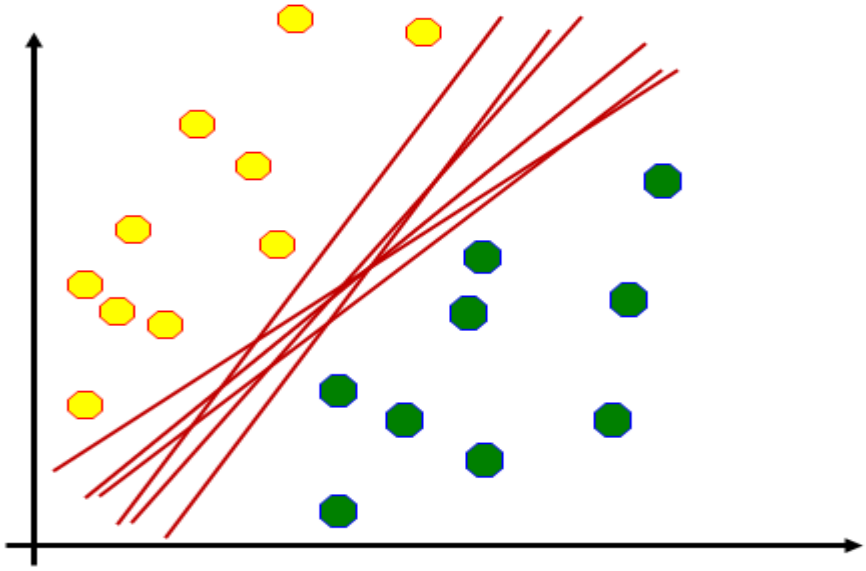


Figure 2.6 Hyper planes for classification of data

From above Figure 2.6, there are many linear classifiers (hyper planes) that separate the data. However, only one of these achieves maximum separation. The reason of using hyper plane is to classify data into two classes. However, hyper plane may be closer to one dataset compared to others and this is not good to happen and thus the concept of large margin classifier or hyper plane is a clear solution. The next Figure 3.6 gives the large margin classifier example, which provides a solution to the above-mentioned problem (Cristianini and Shawe 2000).

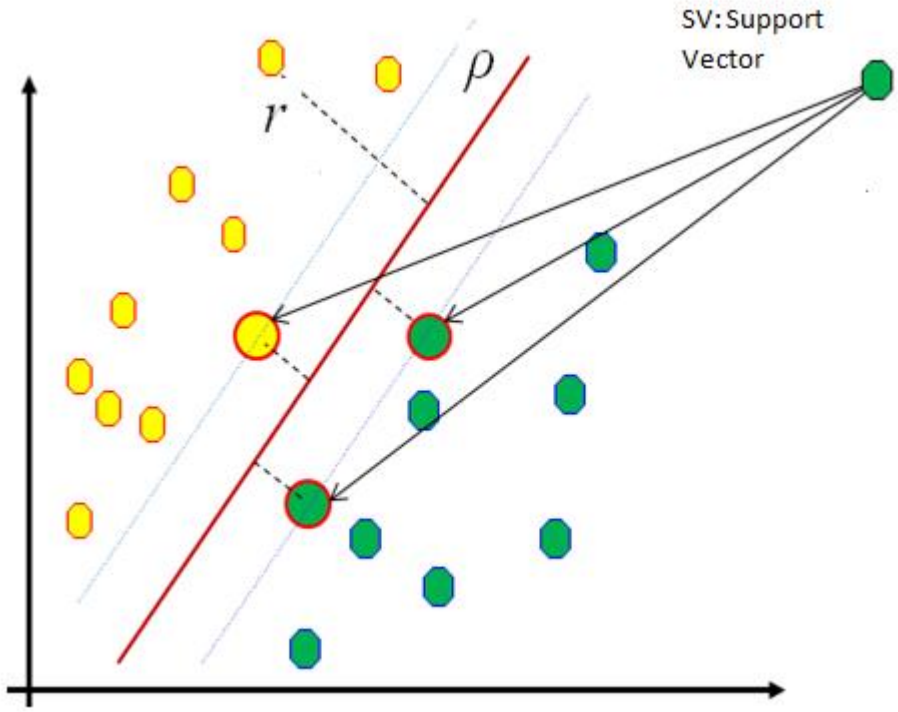


Figure 2.7 Linear SVM

The expression for large margin is given as;

$$m\,argin = \underset{X\epsilon\ D}{\text{argmind}}(\text{X}) = \underset{X\epsilon\ D}{\text{argmin}} \frac{|X.w + b|}{\sqrt{\sum_{i=1}^{d} W_i^2}} \tag{2.2}$$

Figure 2.7 is the large margin linear classifier with the maximum range. In this context, it is an example of a simple linear SVM classifier. There are some good explanations, which include better empirical performance. One advantage is that if there is a small error in the location of the boundary than this gives a least chance of misclassification. The other advantage would be avoiding local minima and better classification. Now, the SVM is expressed mathematically and try to present a linear SVM. The goals of SVM are separating the data with hyper plane and extend this to non-linear boundaries using kernel trick (Mitchell 1997). For calculating SVM, the goal is to classify all the data correctly.

For mathematical calculations, SVM can be represented in the following equations,

$$[a]\ If\ Y_i = +1; \mathrm{wx_i} + \mathrm{b} \geq 1 \qquad (2.3)$$

$$[b]\ If\ Y_i = -1; \mathrm{wx_i} + \mathrm{b} \leq 1 \qquad (2.4)$$

$$[c] for\ all\ i;\ y_i\ (w_i + b) \geq 1 \qquad (2.5)$$

In this equation x is a vector point and w is weight and is also a vector. So to separate the data [a] should always be greater than zero. Among all possible hyper planes, SVM selects the one where the distance of hyper plane is as large as possible. If the training data is good and every test vector is located in radius r from training vector than chosen hyper plane is located at the farthest possible from the data (Lewis 2004). This desired hyper plane which maximizes the margin also bisects the lines between closest points on convex hull of the two datasets. Thus afore mentioned equations (2.3), (2.4) and (2.5) are drawn in Figure 2.8.

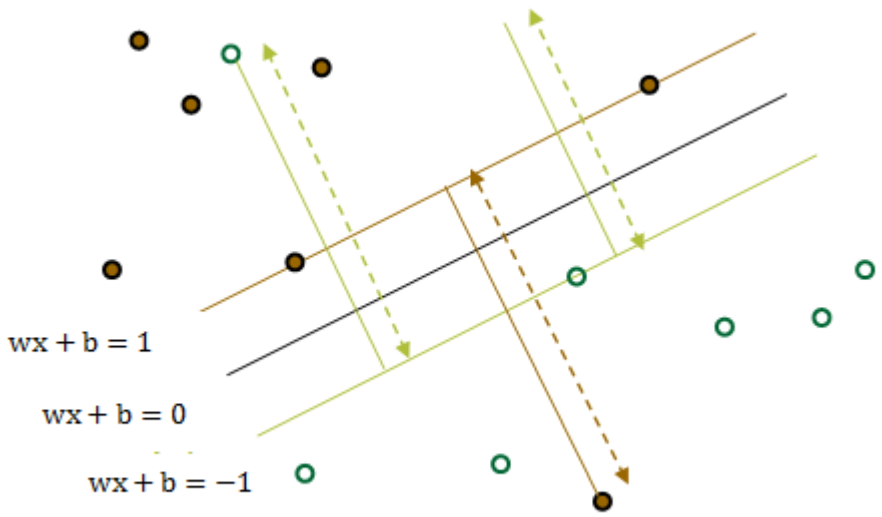


Figure 2.8 Representation of hyper planes

Distance of closest point on hyperplane to origin can be found by maximizing the x as x is on the hyper plane. Similarly, there will be similar scenario for the other side points. Thus solving and subtracting the two distances, the resultant is the summed distance from the separating hyperplane to nearest points.

Maximum Margin = M = 2 / ||w||. Now maximizing the margin is same as minimum (Lewis 2004). Suppose a quadratic optimization problem and there is need to solve for w and b. To solve this problem, the quadratic function has to optimize with linear constraints. The solution involves constructing a dual problem and where a Langlier's multiplier α_i is associated. The target is to find w and b such that Φ (w) =½ |w'||w| is minimized; and for all {(x_i, y_i)}p: y_i (w * x_i + b) ≥ 1.

After solving the result is that w =$\Sigma\alpha_i * x_i$; $b = y_k$- w *x_k for any x_k such that $\alpha k \neq 0$ and the classifying function will have the following form: $f(x) = \Sigma\alpha_i y_i x_i * x + b$.

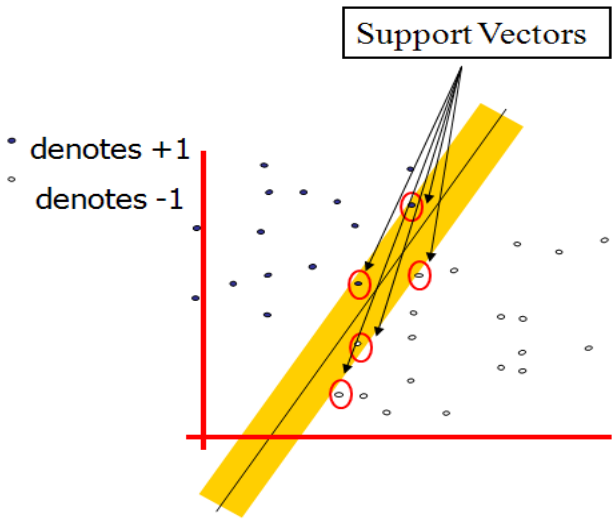


Figure 2.9 Representation of Support Vectors

2.11.2.1 SVM Representation

This section describes the quadratic programming (QP) formulation for SVM classification (Mitchell 1997), (Lewis 2004) and (Burges 1998). The simple representation of SVM can be expressed as,

SV classification:

$$\min_{f,\xi_i} ||f||_k^2 + C \sum_{i=1}^{l} \xi_i \tag{2.6}$$

$$y_i f(X_i) \geq 1 - \xi_{i,} \; for\; all\; i \;\; \xi_i \geq 0 \tag{2.7}$$

SVM classification, Dual formulation:

$$\min_{a_i} \sum_{i=1}^{1} a_i - \frac{1}{2} \sum_{i=1}^{1} \sum_{j=1}^{1} a_i \, a_j y_i y_j K(X_i, X_j) \tag{2.8}$$

Where $0 \leq a_i \leq C, for\; all\; i$; and $\sum_{i=1}^{l} a_i \, y_i = 0$

Variables ξ_i are called slack variables and they measure the error made at point ($\mathbf{x}_i$,y_i).Training SVM becomes quite challenging when the number of training points is large. A number of methods for fast SVM training have been proposed (Mitchell 1997), (Lewis 2004) and (Burges 1998).

2.11.2.2 Soft Margin Classifier

In real world problem, it is not likely to get an exactly separate line dividing the data within the space and there might have a curved decision boundary. There might be a hyperplane, which might exactly separate the data, but this may not be desirable if the data has noise in it. It is better for the smooth boundary to ignore few data points than be curved or go in loops, around the outliers. This is handled in a different way using slack variables those are introduced in existing research work (Mitchell 1997), (Lewis 2004), (Burges 1998). This can be expressed as, $y_i(w'x + b) \geq 1 - S_k$.. This allows a point to be a small distance S_k on the wrong side of the hyper plane without violating the constraint. This might end up having huge slack variables which allow any line to separate the data, thus in such scenarios the Lagrangian variable are

introduced which penalizes the large slacks.

$$\min L = {}^{1}/_{2}\ W'W - \sum \lambda_K\ (\ y_k\ (w'x_k\ + b) +\ s_k - 1 + a \sum s_k) \qquad (2.9)$$

Where reducing α allows more data to lie on the wrong side of hyper plane and would be treated as outliers which give smoother decision boundary (Burges 1998).

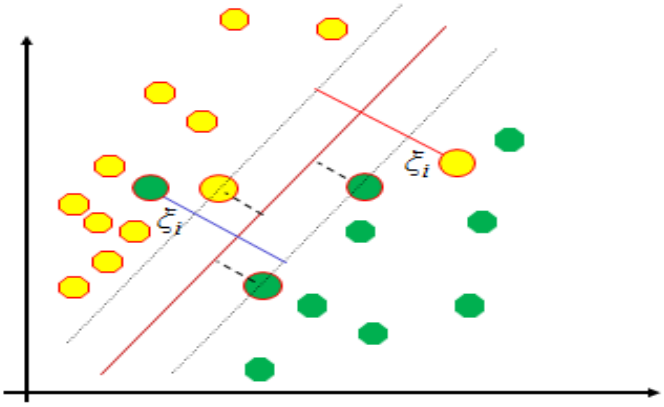


Figure 2.10 Soft margin classification

2.11.2.3 Kernel and Feature Space

a). Kernel: If data is linear, a separating hyper plane may be used to divide the data. However, it is often the case that the data is far from linear and the datasets are inseparable. To allow for this kernels are used to non-linearly map the input data to a high-dimensional space. A very simple illustration of this is shown in Figure 2.11.

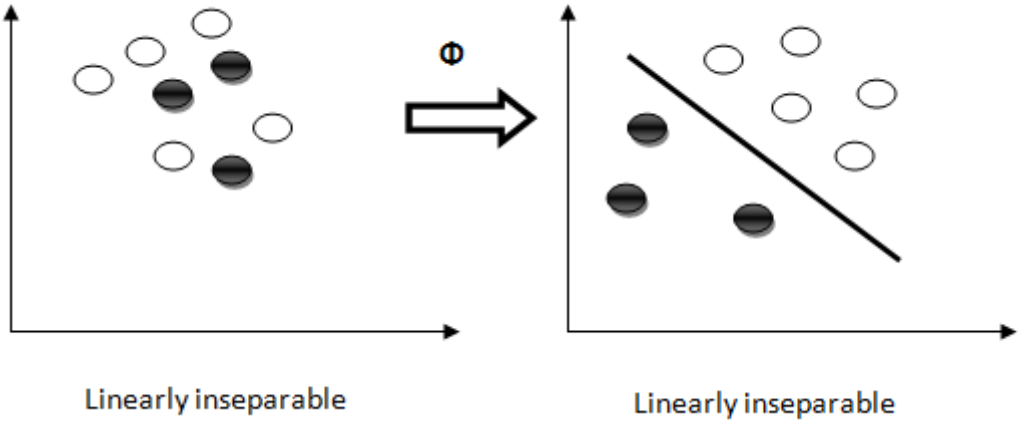


Figure 2.11 Use of kernels

This mapping is defined by the Kernel:

$$K(x, y) = \Phi(x).\Phi(y) \tag{2.10}$$

b). Feature Space: Transforming the data into feature space makes it possible to define a similarity measure on the basis of the dot product. If the feature space is chosen suitably, pattern recognition can be easy.

$$(x_1.x_2) \leftarrow K(x_1, x_2) = (\Phi(x).\Phi(y)) \tag{2.11}$$

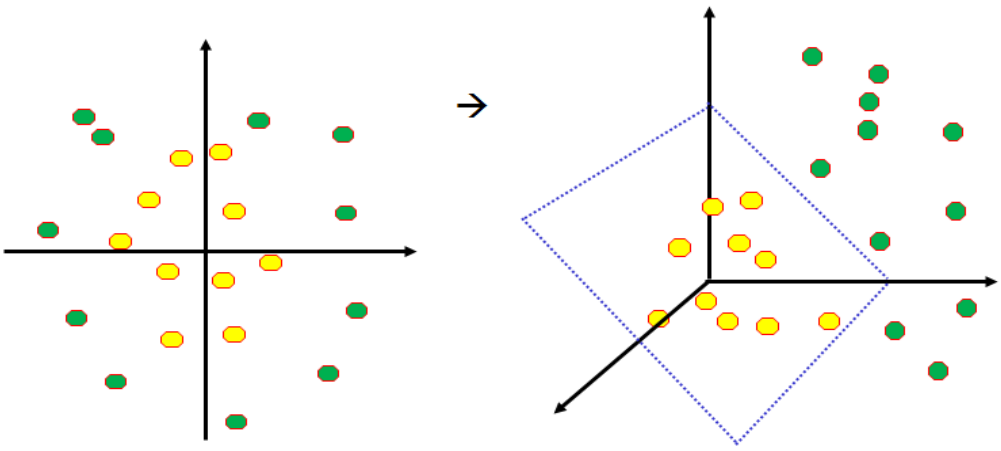


Figure 2.12 Feature space representation

Note the legend is not described, as they are sample plotting to make understand the concepts involved. Now getting back to the kernel trick, when w,b is obtained the

problem is solved for a simple linear scenario in which data is separated by a hyper plane. The Kernel trick allows SVM's to form nonlinear boundaries. The steps involved in kernel trick are given (Burges 1998), (Cristianini et al. 2002).

- The algorithm is expressed using only the inner products of data sets. This is also called as dual problem.
- Original data are passed through non linear maps to form new data with respect to new dimensions by adding a pair wise product of some of the original data dimension to each data vector.
- Rather than an inner product on these new, larger vectors, and store in tables and later do a table lookup, this can be represented by a dot product of the data after doing non linear mapping on them. This function is the kernel function.

a). Dual Problem

First, the problem is converted with optimization to the dual form in which try to eliminate w, and a Lagrangian now is only a function of λ_i. There is a mathematical solution for it. To solve the problem maximize the L_D with respect to λ_i. The dual form simplifies the optimization and the major achievement is the dot product obtained from this method (Burges 1998) and (Cristianini et al. 2002).

b). Inner Product Summarization

This section represents the dot product of the data vectors used. The dot product of nonlinearly mapped data can be expensive. The kernel trick just picks a suitable function that corresponds to dot product of some nonlinear mapping instead (Burges 1998) and (Cristianini et al. 2002). A particular kernel is chosen only by trial and error on the test set, choosing the right kernel based on the problem or application would enhance SVM's performance.

c). Kernel Functions

The idea of the kernel function is to enable operations to be performed in the input space rather than the potentially high dimensional feature space. Hence the inner

product does not need to be evaluated in the feature space. The function performs mapping of the attributes of the input space to the feature space. The kernel function plays a critical role in SVM and its performance. It is based on reproducing Kernel Hilbert Spaces (Nello 2000).

The below mentioned Equation (2.12) shows mapping from input space to the feature space.

$$K(x, x') = (\phi(x), \phi(x')) \tag{2.12}$$

If K is a symmetric positive definite function, which satisfies Mercer's conditions than,

$$K(x, x') = \sum_{m}^{\infty} a_m \, \phi_m(x'), \qquad a_m \geq 0 \tag{2.13}$$

$$\iint K(x, x') g(x) g(x') dx dx' > 0, \quad g \in L_2 \tag{2.14}$$

Then the kernel represents a legitimate inner product in feature space. The training set is not linearly separable in an input space. The training set is linearly separable in the feature space. This is called the "kernel trick" (Cristianini et al. 2002) and (Nello 2000). The different kernel functions are listed below.

Polynomial: A polynomial mapping is a popular method for non-linear modeling. The second kernel is usually preferable as it avoids problems with the hessian becoming Zero.

$$K(x, x') = (x, x')^d \tag{2.15}$$

$$K(x, x') = ((x, x') + 1)^d \tag{2.16}$$

Gaussian Radial Basis Function: Radial basis functions most commonly with a Gaussian form.

$$K(x, x') = \exp\left(-\frac{||x - x'||^2}{2\,\sigma^2}\right) \quad (2.17)$$

Exponential Radial Basis Function: A radial basis function produces a piecewise linear solution which can be attractive when discontinuities are acceptable.

$$K(x, x') = \exp\left(-\frac{||x - x'||}{2\,\sigma^2}\right) \quad (2.18)$$

Multi-Layer Perceptron: The long established MLP, with a single hidden layer, also has a valid kernel representation.

$$K(x, x') = \tanh\left(\rho(x, x')\ell\right) \quad (2.19)$$

2.11.2.3 Applications of SVM

The SVM has been used in the following areas (Bebis et al. 2002).

- Hand written recognition
- Data Classification
- Image processing and data compression Geo- and Environmental Sciences
- Character Recognition
- Intrusion Detection
- Bioinformatics
- Face Recognition
- Decision Tree Predictive Modeling
- E-learning etc.

2.11.3 Genetic Algorithm (GA)

Genetic algorithms are a part of evolutionary computing, which is a rapidly growing area of artificial intelligence. Genetic algorithms (GA) are search algorithms based on the principles of natural selection and genetics. The bases of GA approach are given by Holland and it has been deployed to solve wide range of problems (James 1997).

In a GA, a population of strings (called chromosomes or the genotype of the genome), which encode candidate solutions (called individuals, creatures, or phenotypes) to an optimization problem, evolves toward better solutions. Traditionally, solutions are represented in binary as strings of 0s and 1s, but other encodings are also possible. The evolution usually starts from a population of randomly generated individuals and happens in generations. In each generation, the fitness of every individual in the population is evaluated, multiple individuals are stochastically selected from the current population (based on their fitness), and modified (recombined and possibly randomly mutated) to form a new population. The new population is then used in the next iteration of the algorithm (James 1997).

Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population. If the algorithm has terminated due to a maximum number of generations, a satisfactory solution may or may not have been achieved.

A typical GA requires a genetic representation of the solution domain and a fitness function to evaluate the solution domain. A standard representation of the solution is as an array of bits. Arrays of other types and structures can be used in essentially the same way. The main property that makes these genetic representations convenient is that their parts are easily aligned due to their fixed size, which facilitates simple crossover operations. Variable length representations may also be used, but crossover implementation is more complex in this case. Tree-like representations are explored in genetic programming and graph-form representations are explored in evolutionary programming.

The fitness function is defined over the genetic representation and measures the quality of the represented solution. The fitness function is always problem dependent. For instance, in the knapsack problem one wants to maximize the total value of objects that can be put in a knapsack of some fixed capacity. A representation of a solution might be an array of bits, where each bit represents a different object, and the value of the bit (0 or 1) represents whether or not the object is in the knapsack. Not every such representation is valid, as the size of objects may exceed the capacity of the knapsack. The fitness of the solution is the sum of values of all objects in the

knapsack if the representation is valid, or 0 otherwise. In some problems, it is hard or even impossible to define the fitness expression; in these cases, interactive GAs are used. Once the genetic representation and the fitness function are defined, then GA proceeds to initialize a population of solutions randomly, then improve it through repetitive application of mutation, crossover, inversion and selection operators (Bankovic et al. 2009).

2.11.3.1 *Initialization*

Initially many individual solutions are generated randomly to form an initial population. The population size depends on the nature of the problem, but typically contains several hundreds or thousands of possible solutions. Traditionally, the population is generated randomly, covering the entire range of possible solutions (the search space). Occasionally, the solutions may be "seeded" in areas where optimal solutions are likely to be found (Bankovic et al. 2009).

2.*11.3.2 Selection*

During each successive generation, a proportion of the existing population is selected to breed a new generation. Individual solutions are selected through a fitness-based process, where fitter solutions (as measured by a fitness function) are typically more likely to be selected. Certain selection methods rate the fitness of each solution and preferentially select the best solutions. Other methods rate only a random sample of the population, as this process may be very time-consuming. Most functions are stochastic and designed so that a small proportion of less fit solutions are selected. This helps keep the diversity of the population large, preventing premature convergence on poor solutions. Popular and well-studied selection methods include roulette wheel selection and tournament selection (Bankovic et al. 2009).

a). Roulette Wheel Selection

Parents are selected according to their fitness. The better the chromosomes are, the more chances to be selected they have. Chromosome with bigger fitness will be

selected more times. Imagine a roulette wheel where all chromosomes in the population are placed, everyone has its place accordingly to its fitness function as in Figure 2.13.

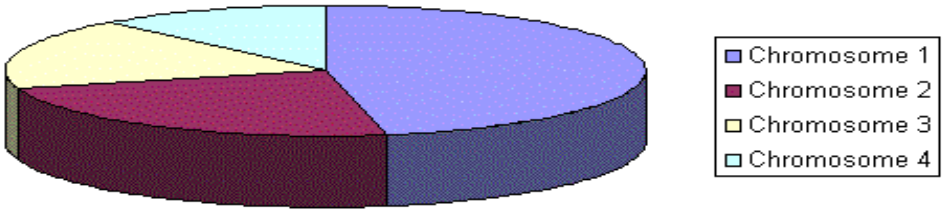


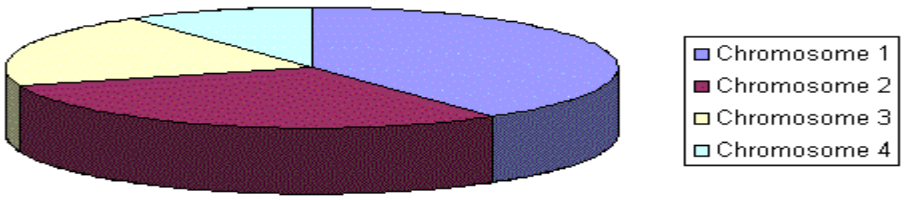
Figure 2.13 Roulette wheel

b). Rank Selection

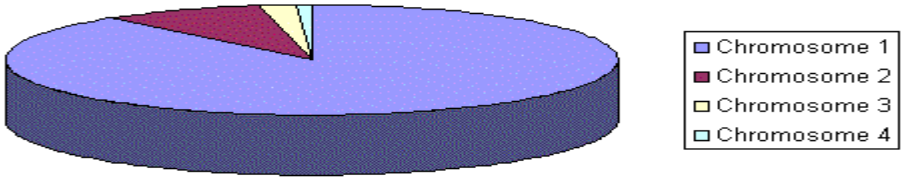
The previous selection will have problems when the fitness differs very much. For example, if the best chromosome fitness is 90% of all the roulette wheel then the other chromosomes will have very few chances to be selected (Bankovic et al. 2007).

Rank selection first ranks the population and then every chromosome receives fitness from this ranking. The worst will have fitness 1, second worst 2 etc. and the best will have fitness N (number of chromosomes in population).

After this, all the chromosomes have a chance to be selected. However, this method can lead to slower convergence, because the best chromosomes do not differ so much from other ones.



Situation before ranking (graph of fitness)



Situation after ranking (graph of order numbers)

Figure 2.14 Graph before and after roulette wheel selection

c). Steady State Selection

This is not particular method of selecting parents. Main idea of this selection is that big part of chromosomes should survive to next generation (Bankovic et al. 2009).

GA then works in the following way. In every generation a few (good - with high fitness) chromosomes are selected for creating a new offspring. Then some (bad - with low fitness) chromosomes are removed and the new offspring is placed in their place. The rest of population survives to new generation.

2.11.3.3 Reproduction

The next step is to generate a second generation population of solutions from those selected through genetic operators: crossover (also called recombination), and/or mutation. For each new solution to be produced, a pair of "parent" solutions is selected for breeding from the pool selected previously. By producing a "child" solution using the above methods of crossover and mutation, a new solution is created which typically shares many of the characteristics of its "parents". New parents are selected for each new child, and the process continues until a new population of solutions of appropriate size is generated. Although reproduction methods that are based on the use of two parents are more "biology inspired", some research suggests that more than two "parents" are better to be used to reproduce a good quality chromosome. These processes ultimately result in the next generation population of chromosomes that is different from the initial generation. Generally the average

fitness will increase by this procedure for the population, since only the best organisms from the first generation are selected for breeding, along with a small proportion of less fit solutions (Bankovic et al. 2009).

a). Crossover

Crossover is a genetic operator that combines (mates) two chromosomes (parents) to produce a new chromosome (offspring). The idea behind crossover is that the new chromosome may be better than both of the parents if it takes the best characteristics from each of the parents (Bankovic et al. 2009). Crossover occurs during evolution according to the Crossover Probability. This probability should usually be set fairly high (0.9 is a good first choice). There are three basic crossover operators; one-point crossover, two-point crossover and uniform crossover.

i. *One Point* - Randomly selects a crossover point within a chromosome then interchanges the two parent chromosomes at this point to produce two new offspring. Consider the following two parents that have been selected for crossover. The" |" symbol indicates the randomly chosen crossover point.

Parent 1: 11001|010

Parent 2: 00100|111

After interchanging the parent chromosomes at the crossover point, the following offspring are produced:

Offspring1: 11001|111

Offspring2: 00100|010

ii. *Two Point* - Randomly selects two crossover points within a chromosome then interchanges the two parent chromosomes between these points to produce two new offspring. Consider the following two parents that have been selected for crossover. The "|" symbols indicate the randomly chosen crossover points.

Parent 1: 110|010|10

Parent 2: 001|001|11

After interchanging the parent chromosomes at the crossover point, the following offspring are produced:

Offspring1: 110|001|10

Offspring2: 001|010|11

iii. *Uniform* - Decides (with the probability defined by the mixing ratio) which parent will contribute each of the gene values in the offspring chromosomes. This allows the parent chromosomes to be mixed at the gene level rather than the segment level (as with one and two point crossover). For some problems, this additional flexibility outweighs the disadvantage of destroying building blocks.

Consider the following two parents, which have been selected for crossover:

Parent 1: 11001010

Parent 2: 00100111

If the mixing ratio is 0.5, approximately half of the genes in the offspring will come from parent 1 and the other half will come from parent 2. Below is a possible set of offspring after uniform crossover:

Offspring1: $1_1 0_2 1_2 0_1 0_2 0_1 1_1 1_2$

Offspring1: $0_2 1_1 0_1 0_2 1_1 1_2 1_2 0_1$

In this research method, one point cross over is used that is simple and performs the best as compared to others. The crossover probability used in all experiments is 0.9.

b). Mutation

Mutation is a genetic operator that alters one or more gene values in a chromosome from its initial state. This can result in entirely new gene values being added to the gene pool. With these new gene values, the GA may be able to arrive at a better solution than was previously possible. Mutation is an important part of the genetic search as it helps to prevent the population from stagnating at any local optima. Mutation occurs during evolution according to the probability defined. This probability should usually be set fairly low. If it is set too high, the search will turn into a primitive random search (Bankovic et al. 2007).

2.11.3.4 Termination

This generational process is repeated until a termination condition has been reached (Bankovic et al. 2007). Common terminating conditions are:

- A solution is found that satisfies minimum criteria
- Fixed number of generations reached
- Allocated budget (computation time/money) reached
- The highest ranking solution's fitness is reaching at such a point that successive iterations no longer produce better results
- Manual inspection
- Combinations of the above
- Means Square Error (MSE)
- Root Means Square (RMSR)

2.11.3.5 Applications of GA

A list of GA application is given.

- Feature selection
- Optimization
- Bioinformatics

- Computational science
- Engineering
- Economics
- Chemistry
- Manufacturing
- Mathematics
- Physics and other fields

2.12 Principal Component Analysis (PCA)

PCA was invented in 1901 by Karl Pearson (Pearson 1901). PCA is a useful statistical technique that has found application in fields such as face recognition and image compression, and is a common technique for finding patterns in data of high dimension. The entire subject of statistics is based on around the idea that you have this big set of data, and you want to analyze that set terms of the relationships between the individual points in that set (Smith 2002).

The goal of PCA is to reduce the dimensionality of the data while retaining as much as possible of the variation present in the original dataset. It is a way of identifying patterns in data, and expressing the data in such a way as to highlight their similarities and differences. However, this method uses PCA for transformation of input vectors to the new search space. The selection of number of principal components is done by GA.

The methodology applied in this work consists of different steps that are described here. The goal is to transform a given data set X of dimension M to an alternative data set Y of smaller dimension L. Equivalently, the goal is to find the matrix Y, where Y is the Karhunen–Loève transform (KLT) of matrix X:

$$Y = KLT\{X\} \tag{2.20}$$

Step 1: Organize the data set

Suppose the data comprises a set of observations of M variables, and the goal is to reduce the data so that each observation can be described with only L variables, L < M. Suppose further, that the data are arranged as a set of N data vectors X1, X2,.....XN with each Xn representing a single grouped observation of the M variables.

- Write X1.....XN as column vectors, each of which has M rows.
- Place the column vectors into a single matrix X of dimensions M × N.

Step 2: Calculate the empirical mean

- Find the empirical mean along each dimension m = 1, ..., M.
- Place the calculated mean values into an empirical mean vector u of dimensions M × 1.

$$u[m] = \frac{1}{N} \sum_{n=1}^{N} X[m, n] \tag{2.21}$$

Step 3: Calculate the deviation from the mean

Mean subtraction is an integral part of the solution towards finding a principal component basis that minimizes the mean square error of approximating the data (Miranda 2008). Further may be proceeded by centring the data as follows:

- Subtract the empirical mean vector u from each column of the data matrix X.
- Store mean-subtracted data in the M × N matrix B.

$$B = X - uh \tag{2.22}$$

Where h is a 1x N row vector of all 1s:

$$h\,[n] = 1 \quad form = 1, \ldots\ldots\ldots, N \tag{2.23}$$

Step 4: Find the covariance matrix

Find the *M* × *M* empirical covariance matrix C.

$$C = E[B \otimes B] = E[B.B^*] = \frac{1}{N}\sum B.B^* \quad (2.24)$$

Where

$\otimes$ is the expected value operator, E is the outer product operator, and * is the conjugate transpose operator. If B consists entirely of real numbers, which is the case in many applications, the "conjugate transpose" is the same as the regular transpose. The covariance matrix in PCA is a sum of outer products between its sample vectors, indeed it could be represented as B.B*.

Step 5: Find the eigenvectors and eigenvalues of the covariance matrix

Compute the matrix V of eigenvectors that diagonalizes the covariance matrix C:

$$V^{-1}CV = D \quad (2.25)$$

Where D is the diagonal matrix of eigenvalues of C. This step will typically involve the use of a computer-based algorithm for computing eigenvectors and eigenvalues.

Matrix D will take the form of an M × M diagonal matrix, where

$$D[p,q] = \lambda_m \quad for\ p = q = m \quad (2.26)$$

$$D[p,q] = 0 \quad (2.27)$$

is the mth eigenvalue of the covariance matrix C, and $for\ p \neq q$

Matrix V, also of dimension M × M, contains M column vectors, each of length M, which represent the M eigenvectors of the covariance matrix C. The eigenvalues and eigenvectors are ordered and paired. The mth eigenvalue corresponds to the mth eigenvector.

Step 6: Rearrange the eigenvectors and eigenvalues

- Sort the columns of the eigenvector matrix V and eigenvalue matrix D in order of decreasing eigenvalue.
- Make sure to maintain the correct pairings between the columns in each matrix.

Step 7: Select a subset of the eigenvectors as basis vectors

Save the first L columns of V as the M × L matrix W:

$$W[p,q] = V[p,q] \tag{2.28}$$

$$for \quad p = 1, \ldots\ldots\ldots, \quad M \ , \qquad q = 1, \ldots\ldots, \quad L$$

Where $\mathbf{1} \leq \boldsymbol{L} \leq \boldsymbol{M}$Use the vector g as a guide in choosing an appropriate value for L. The goal is to choose a value of L as small as possible while achieving a reasonably high value of g on a percentage basis. For example, you may want to choose L so that the cumulative energy g is above a certain threshold, like 90 percent. In this case, choose the smallest value of L such that,

$$(g\,[m = L\,])\,/(\sum_{q=1}^{M} D[q,q] \geq 90\ \%) \tag{2.29}$$

2.13 Literature Review

Intrusion detection initiates from traditional audit systems. In early age of computing environments, large mainframe systems produced sequential records of system events which could then be observed manually for purposes such as accounting and security. In the 1970s, the U.S. Department of Defence (DOD) made security goals for such audit mechanisms, among which were allowing the discovery of attempts to bypass protection mechanisms.

Later, James P. Anderson in 1980 introduced the concept of intrusion detection. Further, he also provided the foundation for future intrusion detection system design and development (Anderson 1980) . His work was the start of host-based intrusion

detection and IDS in general. Denning proposed an intrusion detection model which became a landmark in the research in this area (Denning 1987). The model, which she proposed, forms the fundamental core of most of the intrusion detection methodologies in use these days. Denning conducted a study to create user profiles by analyzing audit trails of the government mainframe computers. The first prototype for intrusion detection, the Intrusion Detection Expert System (IDES) was formed with the help of Denning. IDES analyzes audit trails from government systems and tracks user activity. IDES provided a foundation to the intrusion detection development. Further, she explained that how anomalous activity could be used as an indicator of potential security incidents.

An artificial neural network consists of a collection of processing elements that are highly interconnected and transform a set of inputs to a set of desired outputs. The result of the transformation is determined by the characteristics of the elements and the weights associated with the interconnections among them. By modifying the connections between the nodes the network is able to adapt to the desired outputs. Further, they described that a neural network is an implementation of an algorithm inspired by research into the brain. In fact, one branch of neuroscience uses computers to model cognitive functions. But the neural networks discussed here have little to do with biology. Rather, they show technology in which computers learn directly from data, thereby assisting in classification, function estimation, data compression, and similar tasks. Neural networks are valuable because these are adaptive and have generalization ability (Hammerstrom 1993). There are several methods of responding to a network intrusion, but they all require the precise and well-timed identification of the attack (Cannady 2000b). The existing approaches to misuse detection involve the use of rule-based expert systems to identify indications of known attacks. However, these techniques are less successful in identifying attacks, which vary from expected patterns. He presented an approach to the process of misuse detection that utilizes the analytical strengths of neural networks, and he provided the results from his preliminary analysis of this approach. He made his own dataset using some software packages like the RealSecure™, the Internet Scanner™ products from Internet Security Systems (ISS), Inc, and the Satan scanner. His experimental dataset consists of 10,000 packets in which approximately 3000 were simulated attacks. The nine elements (Protocol ID, Source Port, Destination Port,

Source Address, Destination Address, ICMP Type, ICMP Code, Raw Data Length and Raw Data) were selected because they were typically present in network data packets and they provide a complete description of the information transmitted by the packet. He used Multiple Layered Perceptron (MLP) architecture for training and testing his proposed approach. He used parameters root mean square error (RMSE), Training and Testing data correlation to test their result's sensitivity. This presented work has many drawbacks but it was a good initiative towards the application of neural networks in the area intrusion detection.

Jing and their colleagues described a mechanism in intrusion detection by using artificial neural network (Jing-Xin et al. 2004). They described that the traditional intrusion detection systems mainly consist of two kinds, one is misuse IDS, and the other is anomalous IDS. Misuse IDS works by rule matching, suffering from the updating, the searching and the matching of the rule sets. Anomalous IDS works by statistically computing, suffering from the establishment of the exact statistical model and the selection of the threshold. All of those lower the usability of the traditional intrusion detection systems. To address these problems, several new methods have been proposed, such as data mining, artificial neural networks and artificial immune systems, etc. They mainly discussed the application of the artificial neural networks in the field of IDS research. They have designed and implemented a network intrusion detection system based on the artificial neural networks; and the testing results of the prototype system revealed the validity of the method and the advantages over other methods suggested. In contrast with the traditional methods, the main advantages of the artificial neural networks include the fast / rapid information processing, the stronger ability of tolerance and the ability of self-learning. All of these help to overcome the problems of the traditional IDS (Jing et al. 2004).

Lilia and their colleagues presented a network intrusion detection method to identify and classify illegitimate information in TCP/IP packet payload based on the Snort signature set that represents possible attacks to a network (Lilia et al. 2004). Further, they used a neural network named Hamming Net for their experiments. They selected this network on the base of its capability to classify network events in real-time, and to learn faster than other neural network models, such as, multilayer perceptron with backpropagation and Kohonen maps.TCP/IP packet payloads were

used as input pattern to the Hamming Net and Snort signature as exemplar patterns. The challenges faced to model the input and exemplar data and the strategies adopted to capture and scan relevant data in TCP/IP packets and in Snort signatures were described in their work. Their system showed 70% accuracy in the classification of attacks (Lilia et al. 2004).

Este and their colleagues described an approach to traffic classification based on SVM (Este et al. 2009). They applied one of the approaches to solving multi-class problems with SVMs to the task of statistical traffic classification, and described a simple optimization algorithm that allows the classifier to perform correctly with as little training as a few hundred samples. The accuracy of the proposed classifier is then evaluated over three sets of traffic traces, coming from different topological points in the Internet. Their presented results confirmed that SVM-based classifiers could be very effective at discriminating traffic generated by different applications, even with reduced training set sizes. Further, they used different data sets for instance the lawrence berkeley national laboratory (LBNL) data set, the cooperative association for internet data analysis (CAIDA) dataset and self-simulated data set. Their results showed 90 % accuracy. The system has some drawbacks; it could not handle out-of-order packets, packet loss, and fragmentation in a robust way (Este et al. 2009).

SVM for traffic classification also described in another work by (Li and Guan 2007) . They used a technique to train a classifier to recognize seven different classes of applications. In this approach, flows are divided in common classes such as bulk, multimedia, etc. The authors pointed out that changing the features influence the accuracy of classification results. For regular traffic samples with biased prior probability, they achieved an accuracy of approximately 99.4%. For un-biased samples, with uniform prior probability, their method yielded approximately 96.9% accuracy. The way of selecting features was not much more efficient in their work. They did not use standard data set and compared their results with recent approaches.

A detection mechanism was proposed for traffic flooding attacks by Yu and their colleagues (Yu et al. 2008). They used simple network management protocol (SNMP) management information base (MIB) statistical data gathered from SNMP agents,

instead of raw packet data from network links. The involved SNMP MIB variables were selected by an effective feature selection mechanism and gathered effectively by the MIB update time prediction mechanism. Then, they used a machine learning approach based on SVM for attack classification. Using MIB and SVM, they achieved fast detection with high accuracy, the minimization of the system burden, and extendibility for system deployment. The proposed mechanism is constructed in a hierarchical structure, which first differentiates attack traffic from normal traffic and then classifies the types of attack. They used various types of flooding; transmission control protocol synchronization (TCP-SYN) flooding, user data protocol (UDP) flooding, and internet control management protocol (ICMP) flooding. Using MIB datasets collected from real experiments involving a distributed denial of service (DDOS) attack, they validated the possibility of their approaches. It is shown that network attacks were detected with high efficiency (97.07 %), and classified with low false alarms (Yu et al. 2008).

Several researchers have applied data mining techniques in the design of network intrusion detection system (NIDS) (Khan et al. 2007). One of the promising techniques is SVM, which has concrete mathematical foundations that provided satisfying results. SVM separates data into multiple classes (at least two) by a hyperplane, and simultaneously minimizes the empirical classification error and maximizes the geometric margin. Thus, it is also known as maximum margin classifiers.

Osareh and Shadgar applied NN and SVM techniques on the standard KDD cup 99 dataset which has been utilized in the evaluation of security detection mechanism as a benchmark dataset in several different research works. They selected four different categories of attack such as DOS, probing, R2L and U2R. They proved through simulations that the accuracy of NN is higher than that of SVM, but false alarm and detection rate of SVM is better. They used full features of dataset that decreases the performance of the NN and SVM architecture that will also affect on training and testing overheads. Further, their proposed system demonstrated up to 83.5 % accuracy in detection (Osareh and Shadgar 2008).

Even though SVMs have shown good results in data classification, but they are not favorable for huge dataset because the training complexity is dependent on the amount of data in the training set. Larger amount of data would lead to higher training complexity. However, many data mining applications involve millions or even billions of pieces of data records. For instance, in the KDD cup 1999 dataset, there are more than 4 million and 3 million records in the training set and test set, respectively. The SVM technique is powerless to operate at such a large dataset due to system failures caused by insufficient memory, or may take too long to finish the training. Since this work used the KDD cup 1999 dataset, to reduce the amount of data, a combined (PCA+GA) method was applied to preprocess the dataset before SVM training. This approach improved the performance of the system.

The PCA is an important technique in data compression and feature extraction (Oja 1992) and it has been also applied to the field of intrusion detection (Kuchimanchi et al. 2004) , (Labib and Vemuri 2004), (Shyu et al. 2003). A neural network PCA (NNPCA) and nonlinear component analysis (NLCA) were proposed to reduce the dimensionality of network traffic; their approaches focused on retaining the information of the compressed data compared with that of the original data. PCA was used to detect selected denial-of-service and network Probe attacks; the authors analyzed the loading values of the various feature vector components with respect to the principal components (Labib and Vemuri 2004). Based on principal and minor components, a method called principle component classifier (PCC) studied the use of robust PCA in outlier detection; this method was able to distinguish the nature of the anomalies whether they were different form the normal instances in terms of extreme values or different correlation structures; the PCC achieved about 98% detection rate with 1% false positive ratio. However, all the mentioned PCA methods are based on conventional statistical analysis utilizing batch mode computation, which are not suitable for adaptive learning and online computing (Shyu et al. 2003).

Liu and Yi had worked on unsupervised learning method based on PCA self-organizing map (PCASOM) for network sessions clustering, and a simplified winner-takes-all SOM was used to generate data clusters with a mean vector and principal basis vectors (Liu and Yi 2006).

Liu and their colleagues described a hierarchical ID model based on the PCANN, which has been used for adaptive computing for both misuse detection and anomaly detection (Liu et al. 2007). The design of PCANN based classifier is detailed and a particular selection of features was made by principal components Analysis. First, they selected 22 features then these features were provided to their system. So, there is a possibility to lose much important features that are more sensitive for the classifier (Liu et al. 2007). Although there are many well-known drawbacks of the PCA neural networks, e.g., the inability to provide a nonlinear mapping, the convergence speed of stochastic neural PCA learning algorithms, etc, their simulations perform well for the specific domain of intrusion detection. They introduced two levels in their proposed model. The top level of model is constructed with a norm profile, and it can distinguish 'bad' connections from 'good' ones at the first stage; all the lower levels are signature-based misuse detectors which can give a specified detection; furthermore, their proposed model trained a new classifier using clustered abnormal connections with data flags, and this enabled them to construct an integrated IDS. They performed different experiments to demonstrate the performance of the proposed model on DARPA 1998 evaluation data sets. Their comparative results showed an improvement in detection performance.

Lakhina and their colleagues described that attacks on the network infrastructure are major threats against network and information security. Most of the existing intrusion detection approaches use all 41 features in the network to measure and look for intrusive pattern some of these features are redundant and irrelevant. The drawback of this approach is time-consuming detection process and degrading the performance of intrusion detection system. They presented hybrid algorithm PCANNA (Principal Component Analysis neural network algorithm) to reduce the number of computer resources, both memory and CPU time required to detect attack. They used PCA to reduce the feature and trained neural network to identify attacks. Test and comparison were made on KDD dataset. They demonstrate that their proposed model showed improvement up to 80.4% data reduction, approximately 40% reduction in training time and 70% reduction in testing time. Their proposed method not only reduces the number of the input features and time but also increases the classification accuracy (Lakhina et al. 2010).

The aforementioned work related to PCA on intrusion detection has emphasized the issues of feature extraction and classification; however, relatively less attention has been given to the critical issue of feature selection. The main trend in feature extraction has been representing the data in a lower dimensional space, for example, using PCA. Without using an effective scheme to select an appropriate set of features in this space, however, these methods rely mostly on powerful classification algorithms to deal with redundant and irrelevant features. Therefore, this method providing a new way of feature subset selection in the area of intrusion detection.

Bankovic and their colleagues presented a serial combination of two genetic algorithm-based intrusion detection systems. They proposed many solutions for intrusion detection based on machine learning techniques, but most of them introduced major computational overhead, which made them time consuming and thus increased their period of adapting to the environmental changes. In the first step of their solution they deployed feature extraction techniques using PCA in order to reduce the amount of data that the system needed to process. Hence, their system was simple and reduced significant computational overhead, but at the same time is accurate, adaptive and fast due to genetic algorithms. Furthermore, on account of inherent parallelism, their solution offered a possibility of implementation using reconfigurable hardware with the implementation cost much lower than the that of the traditional systems. They used two types of classifier; linear and rule based. The model was tested on KDD99 benchmark dataset and showed 92.1 % detection rate (Bankovic et al. 2009).

Bankovoc and their colleagues proposed a misuse detection system based on genetic algorithm (GA) approach. They used the KDD99Cup dataset for evolving and testing new rules for intrusion detection. Further, they deployed PCA to extract the most important features of the data (Bankovic et al. 2007). In that way, they were able to keep the high level of detection rates of attacks while speeding up the processing of the data. However, there is one drawback that is a chance to miss some important features that are more sensitive for the classifier. Genetic algorithm (GA) approach is one of the future approaches in computer security, especially in intrusion detection systems (IDS) (Folino et al. 2005).

GA operates on a population of potential solutions applying the principle of survival of the fittest to produce better and better approximations to the solution of the problem that GA is trying to solve. At each generation, a new set of approximations is created by the process of selecting individuals according to their level of fitness value in the problem domain and breeding them together using the operators borrowed from the genetic process performed in nature, i.e. crossover and mutation. This process leads to the evolution of populations of individuals that are better adapted to their environment than the individuals that they were created from, just as it happens in natural adaptation (Bankovic et al. 2009).

Kim and their colleagues proposed fusions of GA and SVM for efficient optimization of both features and parameters for detection models (Kim et al. 2005). Their method provided optimal anomaly detection model which was capable to minimize amounts of features and maximize the detection rates. In experiments, they showed that the proposed method was efficient way of selecting important features as well as optimizing the parameters for detection model and it provided more stable detection rates. One of the drawbacks of using GA for features selection is that the raw features are not in well organized form so there are chances to miss some key features that are important for the classifier (Kim et al. 2005).

Rayan and their colleagues performed one of the first works to intrusion detection by NN (Ryan et al. 1998). They trained and tested an offline neural network intrusion detection mechanism (NNIDS) on a system of ten users. They used 2-Layer MLP architecture for their system and backpropagation for training purpose. The data source for training and testing was operating system logs in UNIX environment. The result parameters to evaluate the performance of the system were false positive and false negative. They implemented their system in the PlaNet neural network simulator (Ryan et al. 1998). Cannady made another work in the same field. He also used the 2-Layer MLP architecture for his system and backpropagation for training purpose. The data source for training and testing was network packets collected by real secure. Nine of the packet characteristics of network data were selected and presented to the MLP network which has four fully connection layers .He used root means square error (RMSE) parameter for training and testing data for performance measuring.

Ghosh and their colleagues presented a host based IDS that focused on building program profiles and used these program profiles to identify normal software behavior and malicious software behavior. The system was trained and tested on SUN platform and used basic security module (BSM) as source of data. Input data were extracted from BSM and a distance metric, which constituted input vectors of the NN. The IDS presented was a single hidden layer MLP. The number of input nodes was equal to the number of exemplar strings. Lucky bucket algorithm is used to capture the temporal locality of anomalous events. Performance analysis was done with DARPA database. Ghosh and Schwartzbard in 1999 also used Elman Networks for intrusion detection (Ghosh and Schwartzbard 1999) . Rhodes et al. 2000 described another work in intrusion detection. They proposed the use of self-organizing neural networks to recognize anomalies in network data stream. Unlike from other approaches which use self organizing maps to process entire state of a network or computer system to detect anomalies, proposed system breaks down the system by using collection of more specialized maps. A monitor stack was constructed and each neural network became a kind of specialist to recognize normal behavior of a protocol and raise an alarm when a deviation from normal profile occurs. The test intrusions were buffer overflow attempt (Rhodes et al. 2000).

Lippmann and Cunningham of MIT Lincoln Laboratory conducted a misuse detection model with neural networks, by searching attack specific keywords in the network traffic. They used a MLP network to detect Unix-host attacks, and attacks to obtain root-privilege on a server. The data that they presented to the neural network consisted of attack-specific keyword counts in network traffic. Two neural networks were used in the system, one for providing an attack probability and one for classifying attacks. A two-layer perceptron was designed with k input nodes, 2k hidden nodes and 2 outputs (Lippmann and Cunningham 2000).

In another study by Zhang and their colleagues, statistical analysis was used in conjunction with MLP networks (Zhang et al. 2001). System is a distributed hierarchical application in the sense that system consists of hierarchy of Intrusion Detection Agents (IDAs) at multiple tiers where each tier corresponds to different network scope. IDAs are IDS components that monitor the activities of a host or a network. An IDA, which consists of components such as the probe, the event pre-

processor, the statistical processor, the neural network classifier and the post processor. Probe collects network traffic and abstracts it into statistical variables. Event pre-processor collects data from probes and other agents and formats it for the statistical analyzer. Statistical model compares the data to the previously compiled reference model, which describes the normal state of the system. A stimulus vector is formed and forwarded to the NN. Neural network analyzes the vector and decides whether it is anomalous or normal. Post processor generates reports for the agents at higher tiers or it may display the results through a user interface. Backpropagation, perceptron, perceptron-backpropagation hybrid, fuzzy ART MAP, radial-basis function networks with 2-8 hidden nodes were tested. The experimental test bed consisting of 11 workstations and 1 server was built by using operations network (OPNET) network simulation software. UDP flooding attack was simulated within the test bed (Zhang et al. 2001).

Lee and Heinbuch worked on experimental IDS with a hierarchy of neural networks. Each of the neural networks in the hierarchy focused on different portions of nominal TCP behavior. Portions of these observed TCP behaviors are connection establishment, connection termination and port usage. System was trained to detect three kinds of attack, which are SYN flood, fast SYN port scan, and stealth SYN port scan (Lee and Heinbuch 2001).

Jirapummin and their colleagues presented an alternative methodology for both visualizing intrusions by using self organizing map (SOM) and classifying intrusions by using resilient propagation (Jirapummin et al. 2002). Neptune attack (SYN flooding), portsweep and satan attacks (port scanning) were selected from KDD cup 1999 data set. For resilient backpropagation (RPROP), 3 layer NN is utilized with 70 nodes in first hidden layer, 12 neurons in second hidden layer and 4 neurons in the output layer. The transfer functions for the first hidden layer, second hidden layer and the output layer of RPROP were tan-sigmoidal, log-sigmoidal and log-sigmoidal respectively (Jirapummin et al. 2002).

Bivens and their colleagues proposed a neural network model for a network-based intrusion detection system. Their anomaly detection system used MLP network for detection. System uses tcpdump data (Bivens et al. 2002). Another study was made

by Shyu and their colleagues. They used KDD cup 1999 as a data source for training and testing of their system. The neural network used by them was PCC (Shyu et al. 2003).

Yu and their colleagues worked on FTP brute force attacks (Yu et al. 2005). They used samples that were collected from local network traffic. They used Hybrid backpropagation/chaotic neural network (BP/CNN) as neural network architecture. A receiver operator characteristics (ROC) curve is used to evaluate the system performance by them.

Amini and Jalili 2006 worked on intrusion detection by using adaptive resonance theory1 (ART1) and adaptive resonance theory2 (ART2). They compare both NN and showed that ART-1 is better in performance wise but ART-2 is better in response wise. They also used standard data set KDD cup 1999 (Amini and Jalili 2005). In another work Amini et al. , they worked on IP, TCP, UDP, and ICMP packets in the local area network (LAN) environment (Amini et al. 2006).

Ahmed and their colleagues worked in the field of intrusion detection. They used full featured Kddcup 99 data set for their system. They used RBPROP NN for training and testing of the network (Ahmad et al. 2007). Another work is also presented by Ahmed et al. in which different backpropagation algorithms were benchmarked. They used MLP architecture in their system (Ahmad et al. in 2008).

Statistical approach, rule based approach, expert system approach, pattern recognition approach, graph-based approach, hybrid approach and artificial neural network approach toward intrusion detection are evaluated using analytic hierarchy process (AHP) (Pervez et al. 2007) and (Sandhya S 2009). The evaluation process takes into account two different types of criteria i.e. main criteria and sub-criteria. The strength of main criteria is based on its efficiency, adaptability, less updating, suitability and maturity, while the sub-criteria consists of economical, time saving, detection rate, minimum false positive, minimum false negative and having the capability to handle varied intrusion and also coordinated intrusion. According to the study (Ahmad et al. in 2008), it has been concluded that among all the approaches, the artificial neural network approach is most suitable to tackle the current issues of intrusions detection systems such as regular updating, detection rate, false positive,

false negative, suitability and adaptability. The comparative analysis is shown Figure 2.15.

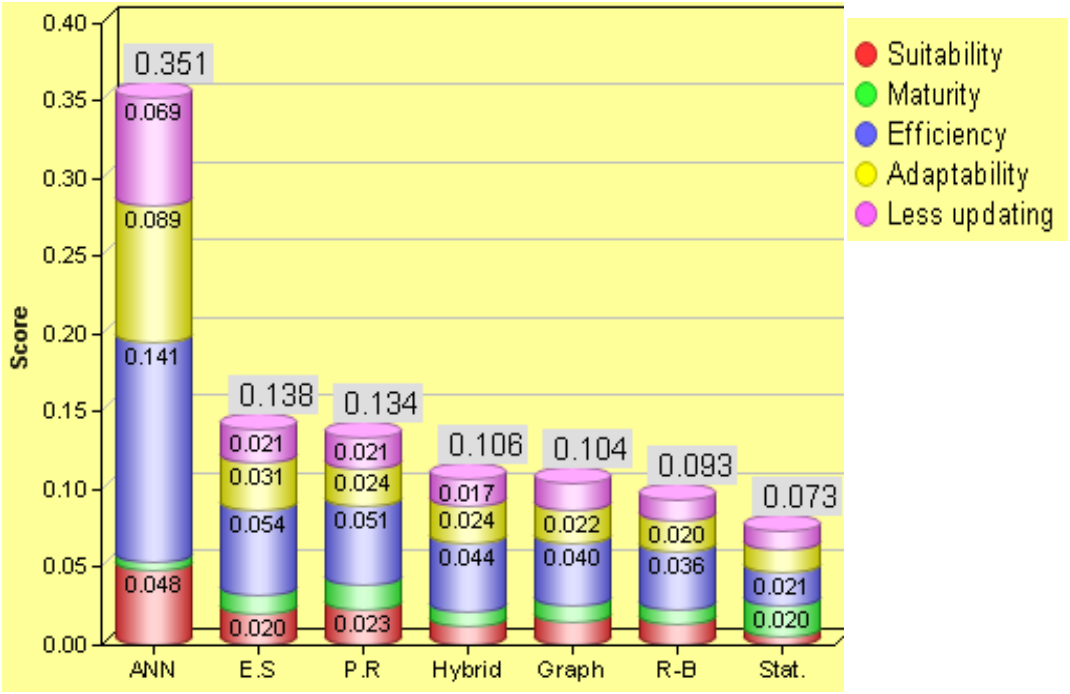


Figure 2.15 Comparative analysis of intrusion detection approaches

Different neural networks for intrusion detection mechanism such as self-organizing map (SOM), adaptive resonance theory (ART), online backpropagation (OBPROP), resilient backpropagation (RPROP) and SVM are evaluated using Multi-criteria Decision Making (MCDM) technique (Dutta et al. 2006) and (Yatim and Utomo 2006). The evaluation based on two types of criteria i.e. the main criteria and sub criteria. The main criteria consists of adaptable, minimum training, performance, maturity and aptitude, while on the other side, the sub criteria consist of detection rate, minimum false positive, minimum false negative, cost, time, handling co-ordinated and varied intrusion.

The hybrid approach using artificial neural networks is a more suitable tactic among other approaches to tackle present issues of intrusion detection systems such as regular updating, detection rate, false positive, false negative, and flexibility. The comparison among them is shown in Figure 2.16.

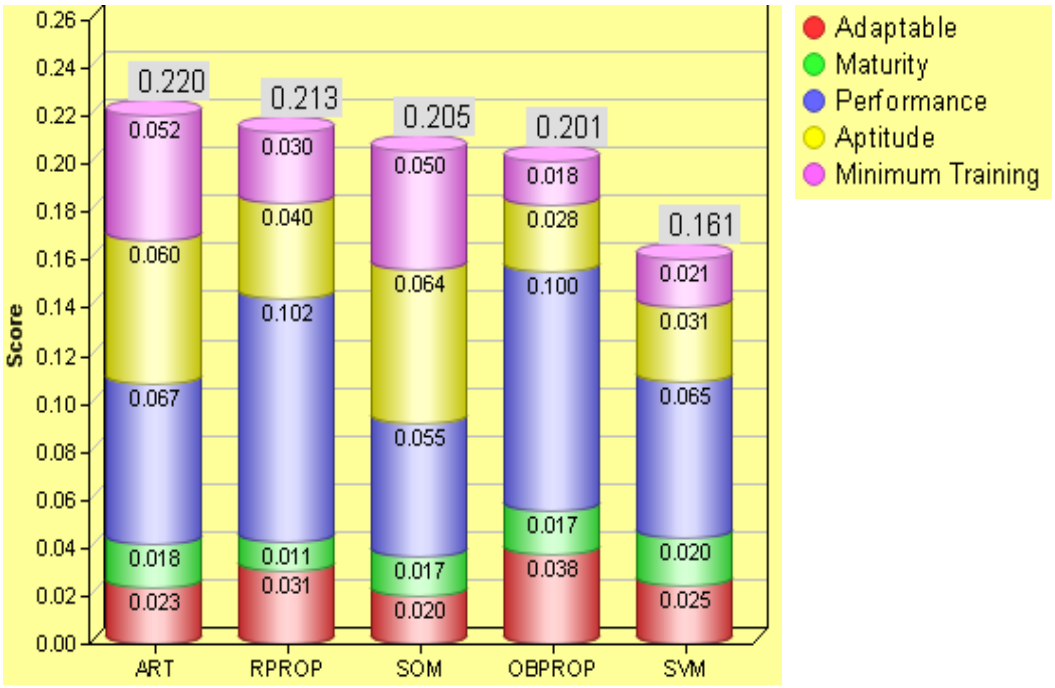


Figure 2.16 Comparative analysis of NN intrusion detection approaches

Three supervised neural networks training algorithms are investigated for intrusion detection mechanism like batch backpropagation (BPROP), online backpropagation (OBPROP) and resilient backpropagation (RPROP) using Java object oriented neural environment (JOONE) and multi-criteria analysis (MCA) technique (Yatim and Utomo 2006) and (Dutta et al. 2006) . The investigation based on two types of criteria; main criteria and sub criteria. The main criteria consist of minimum mean squared error (MSE), less training overhead, performance, memory usage and usability. The criterion "performance" is divided into sub-criteria namely detection rate, minimum false +ve and minimum false -ve. Further, it had concluded that RPROP approach is more suitable approach among other approaches to tackle present issues to intrusion detection systems such as detection rate, false positive, false negative, MSE and memory usage. The comparison among three investigated networks is shown in the Figure 2.17.

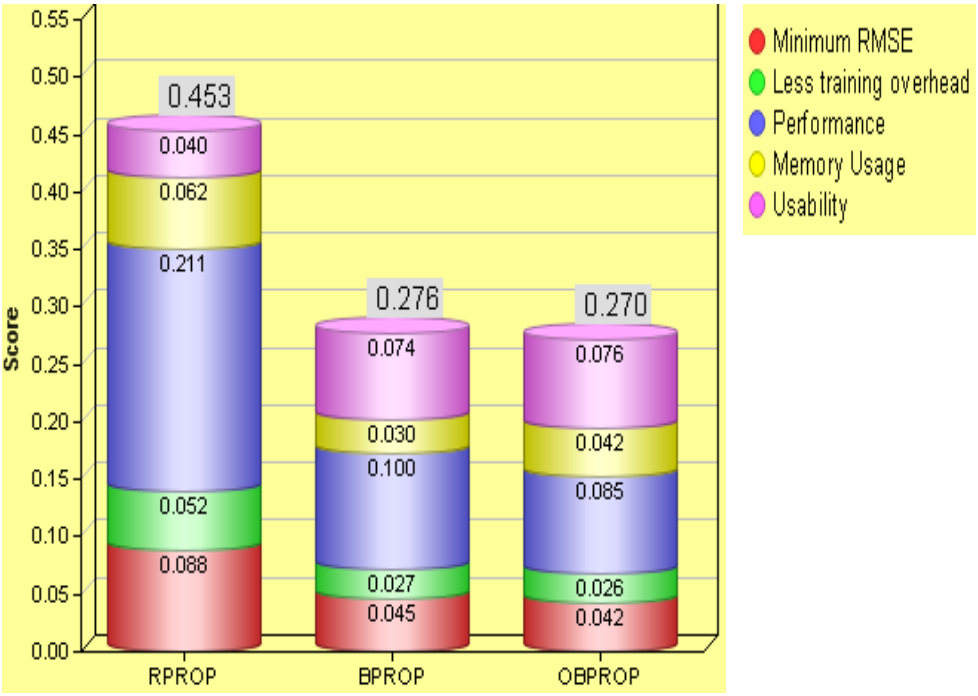


Figure 2.17 Comparative analysis of supervised neural networks

The supervised neural network (SNN) uses supervised learning algorithms such as batch backpropagation (BPROP), online backpropagation (OBPROP), and resilient backpropagation (RBPROP) (Yatim and Utomo 2006) and (Dutta et al. 2006). These SNN algorithms have only one drawback that is unable to detect novel attacks or patterns. On the other hand, unsupervised neural network (UNN) such as self organizing maps (SOMS), and adaptive resonance theory (ART) show poor performance such as detection rate, false positive and false negative but these are more efficient in flexibility and adaptivity (Min and Wang 2009) , (Amini and Jalili 2005). So, a design can be presented for neural network intrusion detection system that merges the advantages of both networks such as SNN and UNN. The working of the designed architecture consists of three phases and is shown in Figure 2.18. The brief detail of each phase is described here.

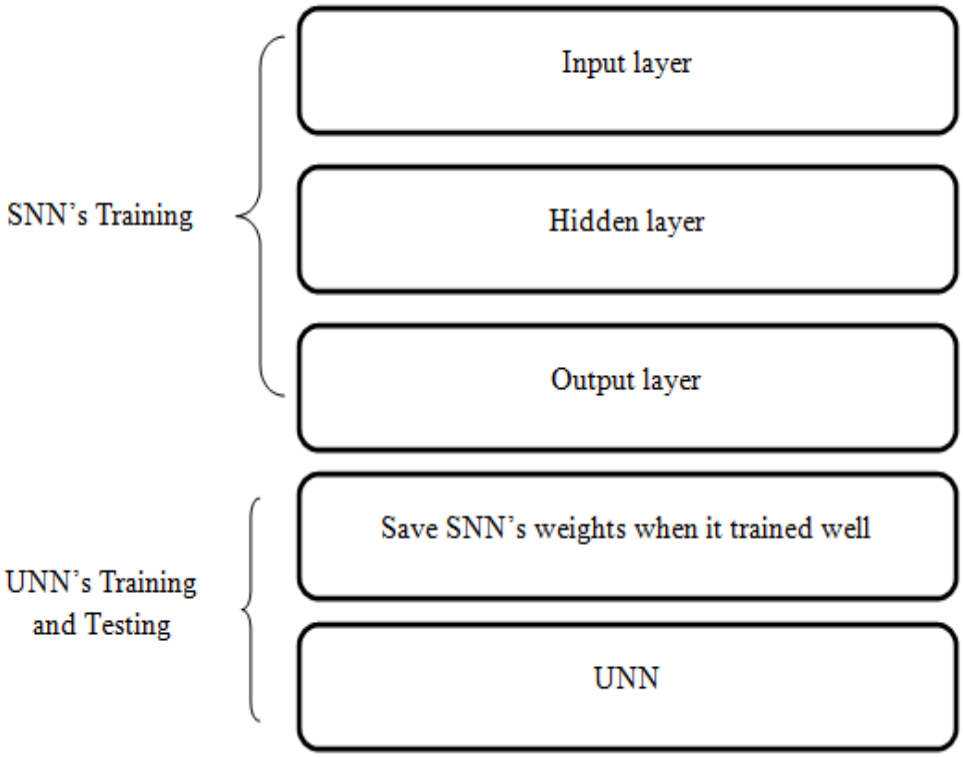


Figure 2.18 Architecture of hybrid learning for NIDS

a) Training SNN

First of all designed SNN is trained on the standard dataset like DARPA. The training process consists of three steps. (1) The Feedforward of input training pattern. (2) The calculation and backpropagation of associated error. (3) The adjustment of the weights.

b) Saving Weights

When SNN is trained well then it freezes its weights. These frozen weights are saved in a file. These saved weights are given to UNN for its initialization and further training and testing process.

c) Training and Testing UNN

Every NN needs weights initialization to start its training process. The optimal assignment of weights to each neuron of neural network is a big issue. A lot of time is

required to reach the optimal weight value through training overhead. Therefore, saved weights used as a starting point for UNN. This is further trained in unsupervised manner to improve performance. However, good results could not obtain due to the problem of different architecture in nature. Because the layered structure of supervised and unsupervised neural networks are different. Therefore, several different types of issues arises; training overhead, saving weights, their proper initialization as inputs for unsupervised neural network architecture.

2.14 A Systematic Review of Related Work

The afore-mentioned work is summarized in a systematic way in Table 2.10. This review consists of main author, year of publication, data source used for training and testing, intrusion analysis structure and results parameters used in earlier research.

Table 2.10 Systematic review of related work

Author	Year	Data Source	Structure	Results
DOD	1970	System audit data	Observed manually	Monitor protection mechanism
J.P Anderson	1980	Introduce the concept of intrusion detection		
Denning	1987	User profile audit trail of main frame system	Proposed an intrusion detection expert system	Anomalous activity
Oja	1992	compression and feature extraction	PCA	Applied to ID
Hammerstrom et al.	1993	Neural Networks (NN) in ID		
Ryan et al.	1998	Operating System Logs	2-Layer MLP	7% FP 4% FN
Cannady	1998	Network packets collected by real secure network	2-Layer MLP	RMSE of 0.0582 for Training Data RMSE of 0.069 for Test Data.

Author	Year	Data Source	Structure	Results
Cannady	2000	His own generated traffic dataset	MLP	RMSE and data correlation
Ghosh et al.	1999	Sun's Basic security module (BSM)	2-Layer MLP	Anomaly Detection: 2.2% FP 22.7% FN Misuse Detection: 18.7% FP &, 9.1 FN
Ghosh et al.	1999	Sun's BSM	Elman Networks	No FP 22.7% FN
Rhodes et al.	2000	Buffer overflow	SOM	D.R (57%) BIND server & rotshb exploit
Lippmann et al.	2000	Network Packets	2-Layer MLP	One False Alarm per Day 20% false Negative D.R 80%
Zhang et al	2001	Network Packets Generated by OPNET UDP flooding attack only	Backpropagation, Perceptron, Perceptron-Backpropagaiton Hybrid, Fuzzy ART MAP, Radial Basis Function Networks	BPROP & HPBPROP performed better than Perceptron, Fuzzy ART MAP, Radial Basis Function networks RMSE<0.05 Statistical Analysis
Lee & Heinbuch	2001	TCP packets [port usage, & connection]	Hierarchy of Neural Networks	SYN flood & Port scanning
Jirapummin et al.	2002	KDD Cup 1999 [TCP SYN & Port Scanning]	3-Layer RPROP with SOM	D.R 90% 5% FP 10% FN
Bivens et al.	2002	DARPA 1999 [DOS,DDOS &Port attacks]	SOM for Clustering MLP for Detection	76% FP No FN
Shyu et al.	2003	KDD cup 1999	PCC	DR→95% FA→5%
Yu et al.	2004	FTP brute force attacks samples from LAN	Hybrid BP/CNN	ROC
Jinget al.	2004	Proposed NNIDS		Overcome traditional IDS issues
Silva et al.	2004	TCP/IP packet payload	Hamming net	Accuracy 70 %
Kuchimanchi	2004	Feature reduction		Applied to ID
Labib	2004	Feature reduction	PCA	Applied to ID

Author	Year	Data Source	Structure	Results
Shyu	2004	Feature reduction	PCA	Applied to ID
Kuchimanchi	2004	KDD cup	NNPCA	
K. Labib	2004	DOS	PCA	
Folino G et al	2005	KDD cup	GA, RULE BASED	
Kim et al	2005	KDD cup	GA+SVM	
Amini et al.	2005	KDD cup99	ART-1 and ART-2	Compare both NN ART-1 better Performance ART-2 better in response wise
Liu et al.	2006	PCASOM		
Amini et al.	2006	LAN→ IP TCP UDP ICMP	SOM + ART-1 ART-2	Compare three NN and found SOM Optimum
Li et al.	2007	Flow classes	SVM	Accuracy 96%
Khan et al.,	2007		SVM	
Liu et al	2007	KDD 22 FEATURES	PCA NN	
Bankovic	2007	KDD	GA, RULE BASED	
Bankovic	2007	KDD	PCA, GA	
Ahmad et al.	2007	KDDCUP99 MIT Lab. USA Full Features Used	RBPROP	D.R compared to ART-1, ART-2 and SOM & RBROP is found optimum

Author	Year	Data Source	Structure	Results
Yu et al.	2008	SNMP MIB TCP,UDP flooding	SVM	Accuracy 97 %
Osareh, and Bita,	2008	KDD	SVM	Accuracy 83 %
Ahmad et al.	2008	KDDCUP99 MIT dataset of Lincoln Laboratory	MLP Online BPROP Batch PROP RPROP	RPROP found best as compared to online and batch
Bankovic et al	2009		PCA,GA, RULE BASED AND LINEAR	Accuracy 92 %
Alice Este et al	2009	CAIDA	SVM	Accuracy 90 %
Ahmad et al.	2009	Probing KDD	4- Layer MLP	Detection rate 98 %
Ahmad et al.	2009	DOS Dataset a subset of KDD cup99 Full Features	MLP 4- Layered and output layer with two processing element/neuron	Detection rate 96.16 %
Lakhina et. al.	2010	KDD cup	PCANNA	Detection rate 80.4 %
Ahmad et al.	2010	U2R Dataset a subset of KDDCUP99 38 Features	GFFNN 3- Layered and output layer with two processing element/neuron	Detection rate 97.7 %
Ahmad et al.	2010	R2L Dataset a subset of KDDCUP99 38 Features	FFNN 3- Layered and output layer with single processing element/neuron	Detection rate 90 %

The above work described in Table 2.10 shows that data can be obtained by one of the following three methods; by using real traffic , by using sanitized traffic and by using simulated traffic but generally IDS are tested on a standard dataset KDD cup of MIT lab. USA. Different researchers used different architectures (NNs, SVM, PCA, Hybrid and Rule-based) to implement their proposed systems in the field of intrusion detection. Predominantly parameters for testing their results are false positives, false

negatives, detection rates and ROC. They used different tools: MATLAB, PlaNet, OPNET, JOONE, URANO, NeuralWorks simulators to implement and test their models for intrusion detection and some of them developed their own systems in a personalized way.

2.15 Issues in Existing Intrusion Detection Approaches

Undoubtedly, soft computing techniques play down a variety of drawbacks in traditional IDSs such as time consuming statistical analysis, regular updating, non adaptive, efficiency, accuracy and flexibility. But they also suffer from several problems in the research of intrusion detection. For instance, training and computational overheads, complex classifier's architecture, accuracy, false alarms, dataset availability, tuning overheads, raw feature set and pre-processing issues. One of the drawbacks of the past intrusion detection methods is the usage of a raw feature set for classification but the classifier may get confused due to redundancy and hence may not classify correctly. Some of the existing approaches of intrusion detection have focused on the issues of feature extraction and classification. However, comparatively less concentration has been given to the critical matter of feature selection. The foremost trend in feature extraction has been representing the data in to another feature space (the PCA space) using PCA. In this method of selecting features on the basis of highest eigenvectors is not appropriate because the features corresponding to the highest eigenvalues may not have the optimal sensitivity for the classifier due to ignoring many sensitive features. As a result, there are many chances to lose some important features that have higher discriminatory power for the classifier. Therefore, there must be an effective scheme to select an appropriate set of features in the PCA space. This leads the classifier to work in an efficient way and increases the overall performance of the intrusion analysis engine. Because, the redundant and irrelevant features increases overheads as well as confuses the classifier.

Therefore, in this thesis, an argument is made that feature selection is an important problem in intrusion detection and Genetic Algorithms (GAs) provide a simple, general, and powerful framework for selecting good subsets of features that

improve detection rates, reduces training and computational overheads, simplify architectural framework of intrusion analysis engine, reduces false alarms and memory usage and speed up the testing process in offline and real time mode. After feature sub set selection, the classification is performed based on adopted methodology. The selection of classifier and determine its architecture is another problem.The collecting of dataset for training/testing is another problem. This can be achieved via three ways; (i) real traffic, (ii) sanitized traffic and (iii) simulated traffic. However some anomalies hinder the utilization of these approaches. Real traffic approach is very unbearable while sanitized approach is risky. The creation of a simulation is also a difficult task and costly. Further, in order to model various networks, different types of traffic is needed respectively. In order to skip issues arising out of all three approaches, KDD cup dataset is used for training/testing in the experimental work. Therefore presently a research is required that will develop optimized intrusion detection mechanism using soft computing techniques that will provide the potential to identify network activity in a robust. In addition, this will reduce overheads as well as increases performance. Thus, in this context, this research work is based on the adopted methodology that results optimal subset of features for intrusion detection mechanism in the subsequent chapters.

2.16 Summary

This chapter details the background knowledge of the intrusion detection systems (IDS), related functional components, classifications and characteristics. Defines foundations, techniques, approaches (anomaly, misuse detection and combined or hybrid approach) to the intrusion detection system. Attack data sets such as KDD cup dataset used in the work. Overview of soft computing, applied techniques (neural networks, SVM and GA). Also explains PCA and its different steps towards feature transformation into PCA space. Further, this chapter describes the related work in intrusion detection using neural networks, SVM, PCA and GA. A tabular presentation of the above mentioned approaches in intrusion detection. The comparative study of existing approaches and related issues. The references of the other related works are also explained.

CHAPTER 3
METHODOLOGY

3.1 Introduction

Based on the identification of the issues via literature review presented in Chapter 3, this chapter describes the methodology incorporated into this research work. Further, the chapter explains different phases of applied methodology; The adopted methodology is divided into five sections; (1) Selection of dataset for training and testing, (2) Pre-processing of dataset, (3) Classification approach, (4) Training the system and (5) Testing the system. These sections of methodology are shown in Figure 3.1. Finally, the chapter concludes with the contributions and directs towards system design and architecture.

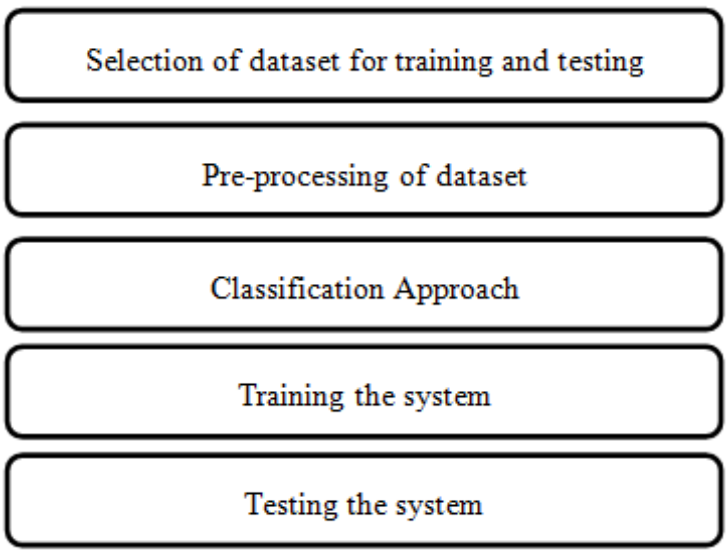


Figure 3.1 Methodology phases

3.2. Selection of Dataset for Training and Testing

The expertise of the intrusion detection mechanism depends on the dataset. Therefore, the performance depends on accuracy of dataset and vice versa. If the training data is optimally accurate with rich contents then efficiency of trained system is improved. Hence, the collection of data for training and testing is a critical dilemma. There are three ways to collect data for experiments in the field of intrusion detection; (i) real traffic, (ii) sanitized traffic, and (iii) simulated traffic. Here, this section describes the pros and cons of these three ways of creating dataset for experimental purposes.

3.2.1 Real Traffic

The dataset is created using real traffic by attacking an organization's servers. In this case, the packets are real but it is unbearable to attack an organization. In addition to that, privacy of the users in the organization may be violated such as private e-mails, passwords and user identities may be released. Hence, this method leads to security and privacy issues.

Advantage

- The dataset consists of real traffic of network packets.

Disadvantage

- Privacy and security issues are raised.

3.2.2 Sanitized Traffic

The problem of security and privacy can be minimized using sanitized traffic. In this method, sensitive information is removed from the data stream and then attack data are inserted into the sanitized traffic.

Advantage

- The dataset consists of real packets without sensitive information and it can be freely distributed for evaluation and experimental purposes.

Disadvantages

- There is a possibility to lose some important features of a packet during the sanitization process.
- It is possible to release sensitive data because it is practically impossible to verify huge amount of data during sanitization process.

3.2.3 Simulated Traffic

The third and the most common way to obtain data, is to create a testbed network and generate background traffic on this network. In the testbed environment, background traffic is generated either by using complex traffic generators modeling actual network statistics or by using simpler commercial traffic generators creating small number of packets at a high rate.

Advantages

- The dataset can be freely distributed, as it does not contain any sensitive information.
- It is guaranteed that generated traffic does not contain any unknown attacks as the background traffic is created by simulators.

Disadvantages

- This is very costly and difficult to create a simulation.
- In order to model various networks, different types of traffic is needed so it increases complexity and cost.

However, aforementioned anomalies hinder the utilization of these approaches. In order to skip issues arising out of all three approaches, this research work uses the defense advanced research projects agency (DARPA) dataset known as knowledge discovery and data mining (KDD) cup for training and testing in my experimental work.

The DARPA project was prepared and executed by the Massachusetts Institute of Technology (MIT) Lincoln Laboratory, USA. This research work uses the KDD cup dataset due to the following reasons:

- One of the reasons for choosing this dataset is that the dataset is standard, which is considered as a benchmark for evaluating security detection mechanisms.
- I evaluated my classification approaches for intrusion detection by analyzing the strengths and weakness of each compartment of the dataset.
- This dataset makes it easy to compare the results of my work with other similar works.
- Another reason is that it is difficult to get another dataset which contains so rich and variety of attacks as KDD cup includes.
- The analysis of intrusion detection approaches in evaluating KDD cup may guide DARPA to future research.

3.3 Pre-processing of Dataset

The next step is preprocessing of selected dataset. Each record of KDD cup dataset consists of three types of features; connection based, content based and time based. There are nine (09) connection based features; thirteen (13) content based features and nineteen (19) are time based features. So, the total number of features are fourty one (41) in each record of KDD cup dataset. Each record represents a network packet that has 41 features. Each packet contains thirty eight (38) numeric features and three (03) symbolic features. First of all, three symbolic features are discarded out of fourty one because these three features do not affect on the applied classification approaches in this research work. A sample of features of the network activity, ‘pre’ and ‘post’ discarding of symbolic values is shown in Table 3.1.

Table 3.1 Feature set of a raw dataset

F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13
0	udp	prvt	SF	105	146	0	0	0	0	0	0	0
F14	F15	F16	F17	F18	F19	F20	F21	F22	F23	F24	F25	F26
0	0	0	0	0	0	0	0	0	1	1	0.0	0.0
F27	F28	F29	F30	F31	F32	F33	F34	F35	F36	F37	F38	F39
0.0	0.0	1.0	0.0	0.0	255	254	1.00	0.01	0.0	0.0	0.0	0.0
F40	F41											

Table 3.2 Feature set after discarding symbolic features

F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13
0	105	146	0	0	0	0	0	0	0	0	0	0
F14	F15	F16	F17	F18	F19	F20	F21	F22	F23	F24	F25	F26
0	0	0	0	0	0	1	1	0.0	0.0	0.0	0.0	1.0
F27	F28	F29	F30	F31	F32	F33	F34	F35	F36	F37	F38	
0.0	0.00	255	254	1.00	0.01	0.00	0.00	0.0	0.0	0.0	0.0	

The remaining thirty eight (38) features are further processed using Principal Component Analysis (PCA) and Genetic Algorithm (GA). The preprocessing of dataset is of great importance as it results in the increase the efficiency of intrusion detection mechanism in case of training, testing, and classification of network activity into normal and intrusive. Further, the preprocessing phase is divided into following parts; (i) feature transformation, and (ii) selection of optimal features.

This part is the actual contribution in the intrusion detection mechanism that prove that my proposed model perform well as compared to existing intrusion detection approaches. The following sections explain these the sub phases of preprocessing.

3.3.1 Feature Transformation

The usage of raw feature set is one of the drawbacks in existing intrusion detection approaches as it causes others problems in the field of intrusion detection like:-

- The classifier or analysis engine of IDS may get confused and will generate false alarm.
- It increases training overhead because the system process on each input feature even it is unimportant for the analysis engine or the classifier.
- This consumes more memory and computational resources of the system during training and testing process of the system.
- This decreases detection rate of an IDS.
- This makes the intrusion detection architecture more complex and malfunction.

3.3.1.1 Principal Component Analysis (PCA)

In order to overcome above issues, this work uses PCA technique to transform original numeric features of dataset into PCA space. In past, the PCA has been use for feature reduction in several different areas; face recognition, hand written text recognition, image compression and intrusion detection. This is a common technique for finding patterns in data of high dimension.

Researchers used PCA to transform raw features into principal features space and select the features based on their sensitivity. The sensitivity is determined by the values of eigenvalues. But here, few other problems are raised:

- Which features are selected?
- How they are selected?

If features are selected based on the values of eigenvalues than there are chances to lose some important features that are more sensitive for the classifier. If all features are selected based on the values of eigenvalues than it will increases training overhead as well as increases architecture complexity that leads towards overall performance degradation. Therefore, This research work uses PCA for feature transformation and organization into new principal features space. This makes the

features more visible, organized, arranged and sensitive that directly impact on the performance of intrusion detection mechanism. The PCA algorithm flow applied for feature transformation and organization is shown in Figure 3.2.

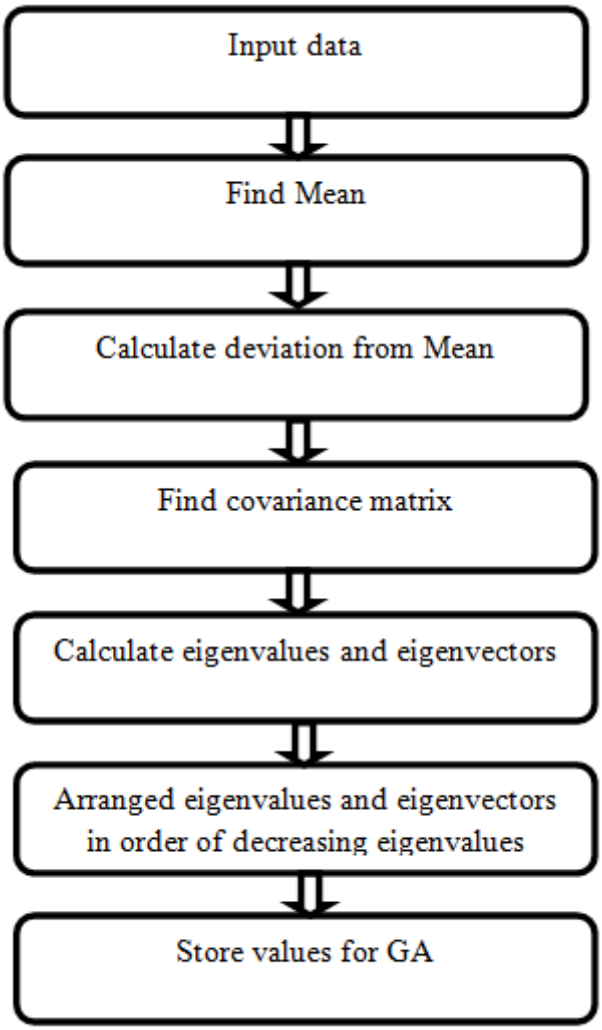


Figure 3.2 PCA algorithm flow

The PCA algorithm used in the pre-processing process for feature transformation and organization is shown in Figure 3.3.

PCA Algorithm:

Suppose $x = (x_{1,}x_{2,}x_{3,} \dots\dots\dots\dots\dots\dots x_M)$ are NX1 vectors. Where M=38.

Step 1: Find Mean:

$$\bar{x} = \frac{1}{M}\sum_{i=1}^{M} x_i$$

Step 2: Calculate deviation from mean: Subtract the mean: $\Phi_i = (x_1 - \bar{x})_i$

Where *i=1, 2,…… M.*

Step 3: Find covariance matrix C:

From the matrix A= [$\Phi 1, \Phi 2, \Phi 3$ ... ... ΦM] (N*M Matrix),

compute C:

$C = \frac{1}{M}\sum_{N=1}^{M} \Phi_N \ \Phi_N \ = AA^T$

Step 4: Compute the eigenvalues of $C: \lambda_1 > \lambda_{2>}\lambda_{3>}\dots\dots\dots\lambda_N$

Step 5: Compute the eigenvectors of $C: \mu_1, \mu_{2,}\mu_{3,}\dots\dots\dots\mu_N$

Since C is symmetric, $C: \mu_1, \mu_{2,}\mu_{3,}\dots\dots\dots\mu_N$ form a basis,

(i.e. any vector x or actually $(x_1 - \bar{x})$, can be written as a linear combination of the eigenvectors):

$$(x_1 - \bar{x}) = b_1\mu_{1+}b_2\mu_{2+} \dots\dots\dots\dots\dots + b_N\mu_N = \sum_{i=1}^{N} b_i\mu_i$$

Step 6: Arranged eigenvalues and eigenvectors in descending order.

Step 7: The dimensionality reduction step (based on largest eigenvalues) is skipped as the selection of principal components is dine using GA.

Figure 3.3 Algorithm for Principal Component Analysis

A set of features from PCA feature space obtained after applying PCA is shown in Table 3.3.

Table 3.3 Feature set from PCA space

F1	F2	F3	F4	F5	F6	F7	F8
-2660.47	-310.209	-456.64	-3.68476	-4.04363	-0.63091	0.063043	-0.00053
F9	F10	F11	F12	F13	F14	F15	F16
-0.00116	-1.25E-05	0.002827	-0.00243	0.000403	0.000198	0.001066	0.000125
F17	F18	F19	F20	F21	F22	F23	F24
0.00010	0.000186	0.000122	-0.00062	1.78E-05	-0.00013	-4.18E-05	0.000115
F25	F26	F27	F28	F29	F30	F31	F32
1.89E-05	-2.47E-05	-6.49E-05	6.79E-06	4.57E-06	9.75E-08	-9.33E-06	1.59E-06
F33	F34	F35	F36	F37	F38		
8.62E-06	1.55E-11	-1.70E-15	6.27E-16	-3.34E-16	1.27E-16		

After feature transformation and organization. The next phase is the feature subset selection. The recent approaches use the PCA to project features space to principal feature space and select features corresponding to the highest eigenvalues, but the features corresponding to the highest eigenvalues may not have the optimal sensitivity for the classifier because of ignoring many sensitive features. Instead of using traditional approach of selecting features with the highest eigenvalues such as PCA, a Genetic Algorithm (GA) is applied to search the principal feature space that offers a subset of features with optimal sensitivity and the highest discriminatory power.

GA method is used to determine subset of feature. An appropriate feature set helps to build efficient decision model as well as to reduce the population of the feature set. Feature reduction will speed up the training and the testing process for the attack identification system considerably but this technically is a compromise between training efficiency (few PCA components) and the accurate results (a large

number of PCA components). So, there is no any effective scheme to select an appropriate set of features in the PCA space in the field of intrusion detection. This is the main problem of feature selection from the principal component space. This problem confuses the classifier or analysis engine when it deals with redundant and irrelevant features.

Advantages

- Training and testing efficiency (few principal components)
- Accurate results (a large number of components)
- Simplify the classifier architecture

Disadvantages

- Selecting some percentage of the top principal components may lose some sensitive features that have higher discriminatory power for the analysis engine.
- Selecting a large number of principal components decrease training and testing efficiency. Hence, it increases memory and computational overheads.
- The classifier architecture becomes more complex as the number of components increases.

3.3.2 Feature Subset Selection

In order to overcome the above issues, GA is applied to search the principal components space so that an optimal subset of features are selected. This is my main contribution that positively impact on the performance of intrusion detection analysis engine.

This section describes feature subset selection process using GA. The block diagram of feature selection is shown in Figure 3.4 and Figure 3.5.

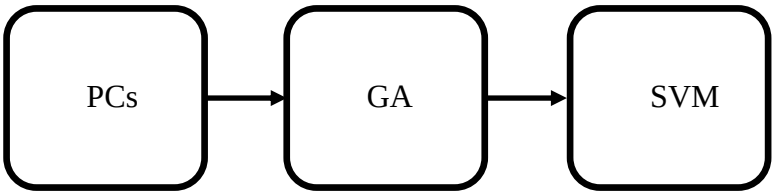


Figure 3.4 Feature subset selection based on GA+SVM

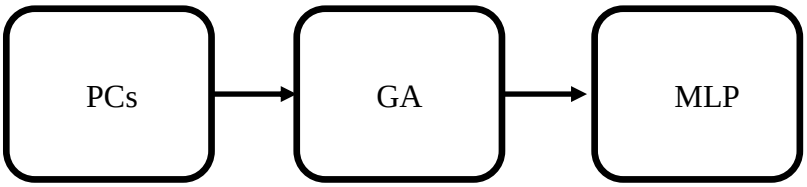


Figure 3.5 Feature subset selection based on GA+MLP

3.3.2.1 Genetic algorithm

Genetic algorithm operates iteratively on a population of structures, each one of which represents a candidate solution to the problem at hand, properly encoded as a string of symbols (e.g., binary). A randomly generated set of such strings forms the initial population from which the GA starts its search.

Three basic genetic operators guide this search such as selection, crossover, and mutation. The genetic search process is iterative consisting of evaluation, selection, and recombination of strings. This continues to iterate in the population (generation) until some termination condition is reached. The GA algorithm flow is shown in Figure 3.6. The general GA algorithmic steps are given in Figure 3.7.

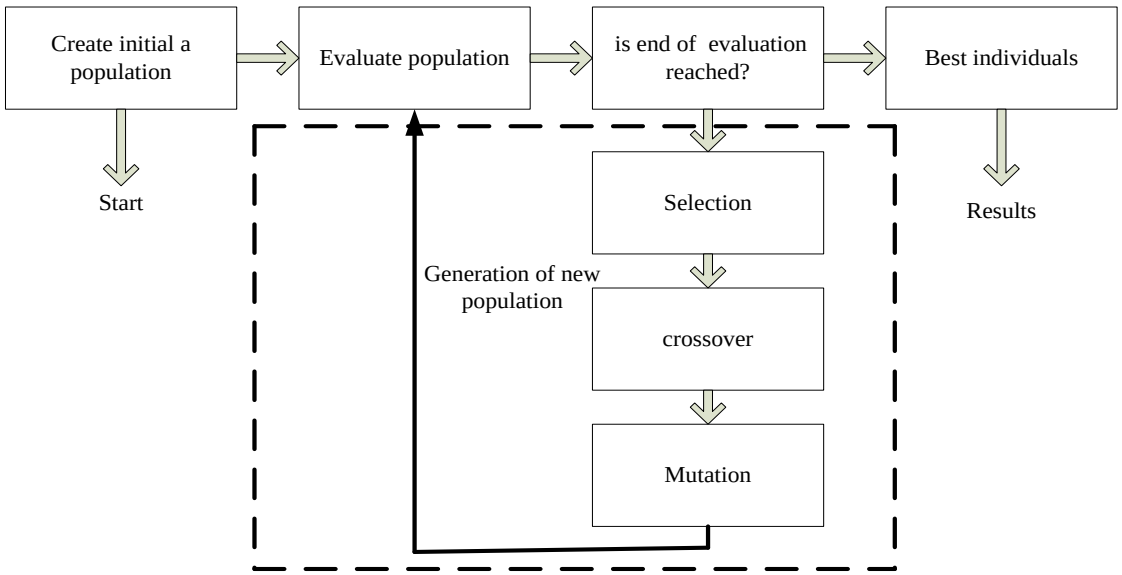


Figure 3.6 Genetic algorithm flow

GA Algorithm:

Step 1. [Start]

Generate random population of n chromosomes

Step 2. [Fitness]

Evaluate the fitness f(x) of each chromosome x in the population

a. [New population] Create a new population by repeating following steps:
b. [Selection] Select two parent chromosomes from a population
c. [Crossover] With a crossover probability cross over the parents to form a new offspring (children). If no crossover was performed, offspring is an exact copy of parents.
d. [Mutation] With a mutation probability, mutate new offspring at each locus (position in chromosome).
e. [Accepting] Place new offspring in a new population

Step 3. [Replace]

Use new generated population for a further run of algorithm

Step 4.

[Test] If the end condition is satisfied, stop, and return the best solution in current population

Step 5.

[Loop] Go to step 2

Figure 3.7 Genetic algorithm

Evaluation of each string is based on a fitness function that is problem-dependent. It determines which of the candidate solutions are better. This corresponds to the environmental determination of survivability in natural selection. Selection of a string, which represents a point in the search space, depends on the string's fitness relative to those of other strings in the population. It probabilistically removes, from the population, those points that have relatively low fitness. Mutation, as in natural systems, is a very low probability operator and just flips a specific bit. Mutation plays a pivotal role of restoring lost genetic material. Crossover in contrast is applied with high probability. It is a randomized yet structured operator that allows information exchange between points. Its goal is to preserve the fittest individuals without introducing any new value.

In brief, selection probabilistically filters out solutions that perform poorly, choosing high performance solutions to concentrate on or exploit. Crossover and mutation, through string operations, generate new solutions for exploration. Given an initial population of elements, Genetic Algorithms use the feedback from the evaluation process to select fitter solutions, eventually resulting into a population of high-performance solutions. Genetic algorithms do not guarantee a global optimum solution. However, they have the ability to search through a very large search spaces and achieve utmost optimal solutions fast. Their ability for fast convergence is explained by the schema theorem (i.e., short-length bit patterns in the chromosomes with above average fitness, get exponentially growing number of trials in subsequent generations) (Sun et al. 2004) and (Goldberg 1989).

a). Feature selection encoding

A simple encoding scheme is used where the chromosome is a bit string whose length is determined by the number of principal components. Each principal component, computed using PCA, is associated with one bit in the string. If the i^{th} bit is 1, then the i^{th} principal component is selected, otherwise, that component is ignored. Each chromosome thus represents a different subset of principal components.

Table 3.4 A sample of five chromosomes (CHR)

CHR	Principal components										
1	1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2	2	2
3	1	3	1	3	1	3	1	3	1	3	1
4	3	4	3	4	3	4	3	4	3	4	3
5	1	5	1	5	1	5	1	5	1	5	1

b). Feature subset fitness evaluation

The main goal of feature subset selection is to use less features to achieve the same or better performance. Therefore, the fitness evaluation contains two terms: (i) accuracy and (ii) the number of features selected. The performance of SVM and MLP is estimated using a validation dataset which guides the GA search. Each feature subset contains a certain number of principal components. If two subsets achieve the same performance, while containing different number of principal components, the subset with fewer principal components is preferred. Between accuracy and feature subset size, accuracy is the major concern. Fitness function is used to demonstrate the combination the two terms:

$$fitness = 10^4 Accuracy + 0.5Zeros \quad (3.1)$$

Where *Accuracy* corresponds to the classification accuracy on a validation set for a particular subset of principal components and *Zeros* corresponds to the number principal components not selected (i.e., zeros in the chromosome). The *Accuracy* term ranges roughly from 0.50 to 0.99, thus, the first term assumes values from 5000 to 9900. The *Zeros* term ranges from 0 to L −1 where L is the length of the chromosome, thus, the second term assumes values from 0 to 37 (L=38). Based on the weights that have been assigned to each term, the *Accuracy* term dominates the fitness value. This implies that individuals with higher accuracy will outweigh individuals

with lower accuracy, no matter how many features they contain. On the whole, the higher the accuracy is, the higher the fitness is. Also, the fewer the number of features is, the higher the fitness is.

Selecting the weights for the two terms of the fitness function is more objective dependent than application dependent. For intrusion classification analysis among many factors, there is need to find the best balance between model compactness and performance accuracy. Under some scenarios, the best performance is preferable, no matter what the cost might be. If this is the case, the weight associated with the *Accuracy* term should be very high. Under different situations, compact models are favoured more over accuracy, as long as the accuracy is within a satisfactory range. In this work, the selection of a higher weight are better for the *Zeros* term.

In this research work, four different experiments performed using GA and the classifiers (SVM and MLP). The fitness is calculated as follows;

$$fitness = 10^4 (.99) + 0.5 (28) = 9900+14=9914 \quad (3.2)$$

Table 3.5 Fitness function

Experiment#	Time taken by experiment	No of selected PCs	No of non selected PCs	Accuracy	Fitness
1-SVM	48 hrs	10	28	0.99	9914
2-MLP	72 hrs	12	26	0.99	9913
3-MLP	78 hrs	20	18	0.98	9808
5-MLP	83 hrs	27	11	0.99	9911

c). Initial population

In general, the initial population is generated randomly, (e.g., each bit in an individual is set by flipping a coin) (Srinivas M and Patnaik L 1995). This, however, would produce a population where each individual contains approximately the same number of 1's and 0's on the average. To explore subsets of different numbers of features, the number of 1's for each individual is generated randomly. Then, the 1's are randomly scattered in the chromosome. In all experiments, this approach used a population size of 50 and 100 generations. In most cases, the GA converged in less than 100 generations.

d). Selection

Selection is a genetic operator that chooses chromosomes from the current generation's population for inclusion in the next generation's population. Before making into the next generation's population, selected chromosomes may undergo crossover and mutation. There are five selection operators; roulette, tournament, top percent, best and random (Eshelman 1989).

- *Roulette:* The chance of a chromosome getting selected is directly proportional to its fitness (or rank). This is where the idea of survival of the fittest comes into play. There is also the option to specify whether the chance of being selected is based on fitness or on rank.
- *Tournament:* it uses roulette selection N times (the Tournament Size") to produce a tournament subset of chromosomes. The best chromosome in this subset is then chosen as the selected chromosome. This method of selection applies addition selective pressure over plain roulette selection. There is also the option to specify whether the chance of being selected is based on fitness or on rank.
- *Best:* Selects the best chromosome (as determined by the lowest cost of the training run). If there are two or more chromosomes with the same best cost, one of them is chosen randomly.
- *Random:* Randomly selects a chromosome from the population.

- *Top Percent:* Randomly selects a chromosome from the top N percent (the Percentage") of the population.

This research work used top percent selection method in experiments because it gives better performance as compared to other selection operators.

Table 3.6 Selection method

Experiment#	Time taken by experiment	No of selected PCs	Selection method	Accuracy	Fitness
1-SVM	48 hrs	10	Top percent	0.99	9914
2-MLP	72 hrs	12	Top percent	0.99	9913
3-MLP	78 hrs	20	Roulette	0.98	9808
5-MLP	83 hrs	27	Roulette	0.99	9911

e). Crossover

There are three fundamental crossovers types: one-point crossover, two-point crossover, and uniform crossover. For one-point crossover, the parent chromosomes are divided at a common point chosen randomly and the resulting sub-chromosomes are swapped. For two-point crossover, the chromosomes are thought of as rings with the first and last gene connected (i.e., wrap-around structure). In this case, the rings are divided at two common points chosen randomly and the resulting sub-rings are swapped. Uniform crossover is different from the above two schemes. In this case, each gene of the offspring is selected randomly from the corresponding genes of the parents (Bebis et al. 2000). For simplicity, this work used one-point crossover here. The crossover probability used in all of my experiments was 0.9.

e). Mutation

Mutation is a genetic operator that alters one or more gene values in a chromosome from its initial state. This can result in entirely new gene values being added to the gene pool. With these new gene values, the Genetic Algorithm may achieve a better solution than the former. Mutation is an important part of the genetic search as it helps to prevent the population from stagnating at any local optima. Mutation occurs during evolution according to the probability defined. This probability should usually be set fairly low. If it is set too high, the search will turn into a primitive random search (Zorana et al. 2007). This work uses the traditional mutation operator which just flips a specific bit with a very low probability. The mutation probability used in all of my experiments was 0.01.

e). Termination

The GA generational process is repeated until a termination condition has been reached. There are many conditions on which GA process can be stopped (Bebis et al. 2000) and (Sun et al. 2004). For example;

- *Population Convergence* – It stops the evolution when the population is deemed converged. The population is deemed converged when the average fitness across the current population is less than the Threshold" percentage away from the best fitness of the current population.
- *Gene Convergence* – It stops the evolution when the *'percentage'* of the genes that make up a chromosome are deemed converged. A gene is deemed converged when the average value of that gene across all of the chromosomes in the current population is less than the *'threshold'* percentage away from the maximum gene value across the chromosomes.
- *Fitness Convergence* – It stops the evolution when the fitness is deemed as converged. Two filters of different lengths are used to smooth the best fitness across the generations. When the smoothed best fitness from the long filter is less than the Threshold" percentage away from the smoothed best fitness from the short filter, the fitness is deemed as converged and the evolution terminates. Both filters are defined by the following equations.

$y(0) = 0.9 * f(0)$ if the objective is set to maximize (3.3)

$y(0) = 1.1 * f(0)$ if the objective is set to minimize (3.4)

$$y(n) = (1- b) f(n) + b\ y(n - 1) \quad (3.5)$$

where *n* is the generation number, y(n) is the filter output, *y(n-1)* is the previous filter output, and *f(n)* is the best cost. The only difference between the short and long filters is the coefficient *b*. From the above equations, the higher the *b,* the more that the past values are averaged in. The short filter uses b = 0.3 and the long filter uses b = 0.9.

- *Fitness Threshold* - Stops the evolution when the best fitness in the current population becomes less than the *fitness threshold* and the objective is set to minimize the fitness. This work uses the threshold value as 0.001.

Table 3.7 Parameters used for genetic feature subset selection

S.No	Genetic operator(s)	Genetic operator value(s)
1	Maximum generation	100
2	Chromosomes	50
3	Selection method	Top percent (10%)
4	Crossover	One-point
5	Crossover probability	0.9
6	Mutation probability	0.01
7	Population size	50
8	Termination type	Fitness threshold (0.001)

S.No	Genetic operator(s)	Genetic operator value(s)
9	Architecture	MLP, SVM
10	Training algorithm	Online backpropagation

The number of features selected during experiments is shown in Table 3.8.

Table 3.8 GA features subset selection based on MLP and SVM

Feature No	SVM(10)	MLP(12)	MLP(20)	MLP(27)
1	X	X	√	√
2	X	√	√	X
3	√	√	√	X
4	√	X	X	√
5	X	X	√	√
6	X	X	X	√
7	X	X	√	√
8	X	X	√	√
9	X	√	X	X
10	X	X	√	√
11	X	√	√	X
12	√	√	√	√
13	X	X	X	X
14	X	X	X	√
15	X	√	√	√
16	X	X	X	X
17	X	√	X	√
18	X	√	X	X
19	X	X	√	X
20	X	X	√	√

Feature No	SVM(10)	MLP(12)	MLP(20)	MLP(27)
21	X	X	X	√
22	√	X	X	√
23	X	X	X	X
24	X	√	√	X
25	X	X	√	√
26	√	X	X	√
27	√	√	X	√
28	√	X	√	√
29	X	X	√	√
30	X	X	X	X
31	X	X	X	√
32	√	X	X	√
33	X	X	√	√
34	√	√	X	√
35	X	X	X	√
36	√	√	√	√
37	X	X	√	√
38	X	X	√	√

This research work found an optimized subset of features with ten features from the above four subsets of features. Consequently, this work used a subset of features with ten principal components with different index values. A sample of final subset of features for ten records is shown in Table 3.9.

Table 3.9 A sample of final subset of features for ten records

Sr	Principal components indexes selected based on genetic algorithm									
No	3	4	12	22	26	27	28	32	34	36
1	-310.209	-456.64	-0.00116	0.0028	-0.00243	0.00106	0.0010	0.00018	0.00011	-6.49E-5
2	-310.209	-456.64	-0.00116	0.00282	-0.00243	0.00106	0.0010	0.00018	0.0011	-6.49E-05
3	-960.96	268.699	0.26708	0.77291	0.01081	-0.0144	-0.060	0.00072	-0.0476	-0.00198
4	442.687	219.551	0.13768	-0.07388	-0.04755	-0.0302	-0.097	-0.0024	-0.0173	0.00040
5	-299.009	275.682	0.32907	0.70899	0.07318	0.01681	-0.061	0.01833	-0.0413	0.00645
6	-310.209	-456.64	-0.00116	0.00282	-0.00243	0.00106	0.0010	0.00018	0.00011	-6.49E-05
7	485.004	274.342	-0.02324	-0.12817	0.10374	0.01239	0.0067	0.00014	-0.0102	0.00409
8	517.0306	244.1791	0.121362	-0.14683	-0.04805	0.00237	-0.097	0.00345	0.00243	0.000388
9	618.482	255.107	-0.24427	-0.00891	0.037715	0.00984	-0.003	-0.0053	0.00075	-0.00093
10	-310.209	-456.64	-0.00116	0.002827	-0.00243	0.00106	0.001	0.0001	0.0001	-6.49E-05

3.4 Classification Approach

This work used two types of approaches for intrusion detection; multilayered perceptron (MLP) and Support Vector Machine (SVM). These two architectures are very popular in different areas of research; image processing, character recognition, speech recognition, bioinformatics, data classification, intrusion detection and machine translation (Sun et al. 2004). The MLP with one hidden layer is equivalent to SVM. In this way, the performance of both architectures can be compared in terms of their discriminatory power and efficiency to classify network activity into normal and intrusive. Both the architectures will be described in details in Chapter 4.

3.4.1 MLP Classifier

There are two important characteristics of the multilayer perceptron (MLP). First, its processing elements (PEs)/neurons are nonlinear. The nonlinearity functionality is provided by the functions; logistic and hyperbolic tangent. Second, they are massively interconnected such that any element of a given layer feeds all the elements of the next layer (Ahmad et al. 2007).

Advantages

The following are some advantages of using MLP in my problem.

- MLPs are very powerful pattern classifiers.
- With one or two hidden layers they can approximate virtually any input-output map.
- They showed better performance to other classifier in difficult problems.
- They efficiently use the information contained in the input data.

Disadvantages

The following are some disadvantages of MLP.

- They need lots of input data. This can slow training process.
- The setting of parameters can be tricky for difficult problem.
- Stuck in local minima

3.4.2 SVM Classifier

SVMs are primarily two-class classifiers that have been shown to be an efficient and possess more systematic approach to learn linear or non-linear decision boundaries (Vapnik V. 1995) and (Burges 1998). Their key characteristic is the mathematical tractability and geometric interpretation. A rapid growth of interest in SVMs has been observed over the last few years, demonstrating remarkable success in fields as diverse as text categorization, bioinformatics, and computer vision (Cristianini et al. 2002). Specific applications include text classification (Tong et al. 2001), speed

recognition (Smith N. and Gales M. 2002) , gene classification (Brown et al. 1999), and webpage classification(Yu H et al. 2002).

This work used SVM using kernel adatron algorithm. The kernel adatron maps inputs to a high dimensional feature space, and then optimally separates data into their respective classes by isolating those inputs that fall close to the data boundaries (Yu et al. 2008). Therefore, kernel adatron is especially effective in separating sets of data that share complex boundaries. SVMs are generally useful for classification problems.

Advantages

The following are some advantages of using SVM in my problem.

- SVMs produce excellent results in classification problems.
- SVM performs better in term of not over generalization. SVM control over training by maximizing the margin.
- There are no parameters specific to the SVM that needs to be configured.
- Some other features of SVMs are the use of kernels, the absence of local minima, the sparseness of the solution and the capacity control obtained by optimizing the margin.

Disadvantages

The following are some disadvantages of SVM.

- SVMs assign one gaussian function for each input exemplar in the training set. This can slow training process.
- SVMs are not suitable for huge datasets.

3.5 Training the System

During the training of the system, both input patterns and desired outputs related to each input exemplar. The aim of the system's training is minimizing the difference between the output produced by the system and the desired output. In order to achieve this goal, weights are updated by carrying out certain steps known as training. First

of all, 20,000 samples of network connections are selected randomly from KDD cup dataset. The selected dataset consists of 12,800 (64%) normal and 7200 (36%) intrusive ones (DOS, Probe, U2R and R2L). After that, the selected dataset is transformed into another space (the PCA space). Then, GA is applied for the selection of optimal features subset as described in section pre-processing. The resultant dataset is further divided into two parts; training and production datasets.

3.5.1 Training Dataset

The training dataset consists of five thousand (5000) labeled connections (network records with label as normal or intrusive) that are randomly selected from 20,000 connections. Further, the training dataset (five thousand) is divided into three parts; (i) cross validation dataset, (ii) test dataset and (iii) training dataset. This section describes each of these datasets.

3.5.1.1 Cross-validation Dataset

Cross-validation is highly recommended method for training the system. This method monitors the error on an independent set of data and stops training when this error begins to increase. The size of dataset for cross-validation is recommended as follows:

- *Normal* generalization protection specifies that 20% of data should be for cross validation.
- *High* generalization protection specifies that 40% of data should be for cross validation.

Hence, This work used one thousand (1000) dataset for cross-validation.

3.5.1.2 Testing Dataset

The testing dataset is used to test the performance of the system. Once the system is trained the weights are then frozen, the testing dataset is fed into the system and the system output is compared with the desired output. In this work, 30 % of dataset (5000) that is fifteen hundred (1500) is used to test the performance of trained system.

3.5.1.3 Training Dataset

The training dataset is used to train the system. The 50% of the dataset (5000) that is 2500 is used to train the system. The training of the system (MLP & SVM) should be stopped when the system has learned the task. There are no direct indicators that measures how and when to stop the training of the system. However, there are some ways on the bases of which the training process can be stopped. These methods are explained in next section.

3.5.2 Training Stop Criteria

There are three common methods; (i) Number of iterations, (ii) Mean Squared Error (MSE), and (iii) Generalization to stop the training of the system (Principe et al. 2000). This section explains each of these methods.

3.5.2.1 Number of Iterations

First method is the simplest way to stop the training phase. This uses a predefined value. It does not use any information or feedback from the system before or during training. When the number of iterations is reached at a predefined value, there is no guarantee that the learning system has found coefficients that are close to the optimal values.

3.5.2.2 Mean Squared Error (MSE)

This method uses a recursive analysis of the output MSE to stop the training. There are two common approaches to stop training that are based on MSE.

- The training is set to terminate when the MSE drops to some threshold.
- The training is set to terminate when the change in the error between epochs is less than some threshold.

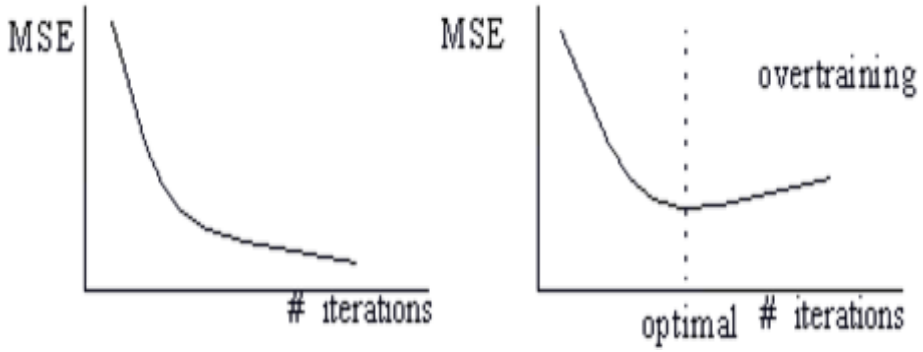


Figure 3.8 Behavior of MSE for training and test datasets

Stop criteria are all based on monitoring the mean square error. Monitor the MSE for the test set, as in cross validation. One should stop the learning when error in the test set starts increasing as shown in Figure 3.8. This is where the maximum generalization takes place.

3.5.2.3 Generalization

The above two methods did not deal with the trouble of generalization, that is, how well the learning system performs with data that does not belong to the training set. Recent development in learning theory indicate that after a critical point an MLP trained with backpropagation will continue to do better in the training set, but the test set performance will begin to deteriorate. This process is called overtraining. One method to solve this problem is to stop the training at the point of maximum generalization. This method is called early stopping or stopping with cross-validation.

It has been experimentally verified that the training error always decreases when the number of iterations is increased as shown in Figure 3.9. If the error is plot in a set of data with which the network was not trained (the validation set), than the error initially decreases with the number of iterations but eventually starts to increase again. Therefore, training should be stopped at the point of the smallest error in the validation set and when the error in the cross-validation set starts to increase. This method has one advantage and one disadvantage.

- This provides an accurate stopping point.
- The cross-validation dataset decreases the size of the training dataset.

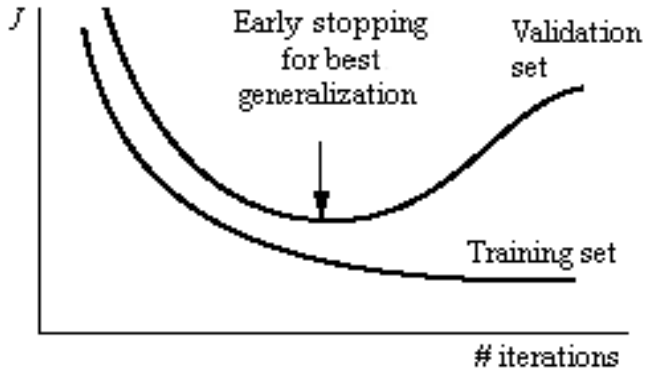


Figure 3.9 Cross validation vs. training dataset

This method is recommended for real world applications. Even though, the MSE is a good overall measure of whether a training run was successful, sometimes it can be misleading (Principe et al. 2000). This is particularly true for classification problems.

3.5.2.4 Confusion matrix

This work deals the classification problem and there is a chance of misleading. Hence, in this research work, a confusion matrix is used to resolve this problem of misleading and to verify the training. The confusion matrix tallies the results of all exemplars of the last epoch and computes the classification percentages for every output vs. desired combination.

There are four parameters in a confusion matrix; (i) true positive, (ii) false positive, (iii) true negative , and (iv) false negative as shown in Table 3.10.

- True positives: when system classifies normal as normal packet then it will be called as true positive. True positives indicate correctly prediction of normal packets.
- False positives: when system classifies normal as intrusive packet then it will be called as false positive. False positives indicate incorrectly prediction of normal packets.
- True negative: when system classifies intrusive as intrusive packet then it will be called as true negative. True negatives indicate correctly prediction of intrusive packets.
- False negative: when system classifies intrusive as normal packet then it will be called as false negative. False negatives indicate incorrectly prediction of intrusive packets.

Table 3.10 A confusion matrix

	Normal	Attack
Normal	True Positive Normal as normal Correctly predicted	False Positive Normal as intrusive Minimize
Attack	False Negative Intrusive as Normal Minimize	True Negative Intrusive and Intrusive Correctly predicted

3.6 Testing the System

When the training is completed then weights of the system are frozen and performance of the system is evaluated. Testing the system involves two steps; (i) verification step, and (ii) generalization step.

3.6.1 Verification Step

In verification step, system is tested against the data which are used in training. Aim of the verification step is to test how well trained system learned the training patterns in the training dataset. If a system was trained successfully, outputs produced by the system would be similar to the actual outputs. This work used 30% of the training dataset (5000) that is 1500.

3.6.2 Generalization Step

In generalization step, testing is conducted with data which is not used in training. Aim of the generalization step is to measure generalization ability of the trained network. After training, the system only involves computation of the feedforward phase. For this purpose, a production dataset is used that has input data but no desired data. This research work uses a dataset of fifteen thousand (15,000) as a production dataset. Further, this technique is also tested on total dataset (20,000) that consist of both training dataset and production dataset. Table 3.11 shows statistics of the dataset used for experiments.

Table 3.11 Statistics of dataset used in experiments

S.No	Dataset(s)	Number of network connections
1	Selected dataset (64% normal and 36% intrusive)	20,000 network connections are selected randomly from KDD cup dataset, in which 12800 are normal and 7200 are intrusive connections.
2	Training dataset	5,000 connections are randomly selected
3	Cross-validation dataset	1,000 connections (20 % of 5000)
4	Testing dataset	1,500 connections (30% of 5000)
5	Production dataset	1,5000

3.7 Summary

This chapter describes the five-phased methodology incorporated in my research. Explains different sections of applied methodology: (i) selection of dataset: There are three ways of selecting dataset for training and testing purpose such as real, sanitized and simulated. However, due to security issues and cost ineffectiveness and complexity this research work used MIT KDD cup, as it is considered a benchmark in the intrusion detection evaluation world. (ii) Pre-processing of dataset by the application of PCA and GA. Searching of PCA feature space using GA space for the selection of optimal feature subset selection is my principal contribution, which has a magnanimous effect on the overall performance (accuracy improvement, simplification of the architecture and minimizes training and testing overheads) of the intrusion analysis engine. The necessity of the preprocessing of the dataset is of prime importance for the classifier to discriminate data into classes such as normal and intrusive. (iii) Classification Approach: this technique used MLP and SVM as a classifier to classify network activity into normal and intrusive. (iv) Training the MLP and SVM classifier using the back-propagation algorithm and kernel adatron respectively. (v) Testing: Post-training, the system's performance is evaluated by freezing the weights of the system and this is done in two steps (a) verification and (b) generalization.

CHAPTER 4

SYSTEM DESIGN AND ARCHITECTURE

4.1 Introduction

This chapter describes the proposed model with its basic architecture via block diagram, and then details of each part or block of its main architecture. Explains features description of the dataset used for experiments, feature transformation process using PCA and optimal features subset selection using GA. The chapter also describes the details of classification architectures with basic algorithms and mathematical foundation of multilayered perceptron (MLP) and Support Vector Machine (SVM). Thus, the chapter explains system implementation, and the basic parameters used during training and testing followed by the description of the contributions and summary.

4.2 Proposed Model

The proposed model consist of different parts; dataset used for experiments, feature transformation and organization, optimal feature subset selection, classification architectures, implementation, training and testing, and results comparison. The block diagram of proposed model is shown in Figure 4.1

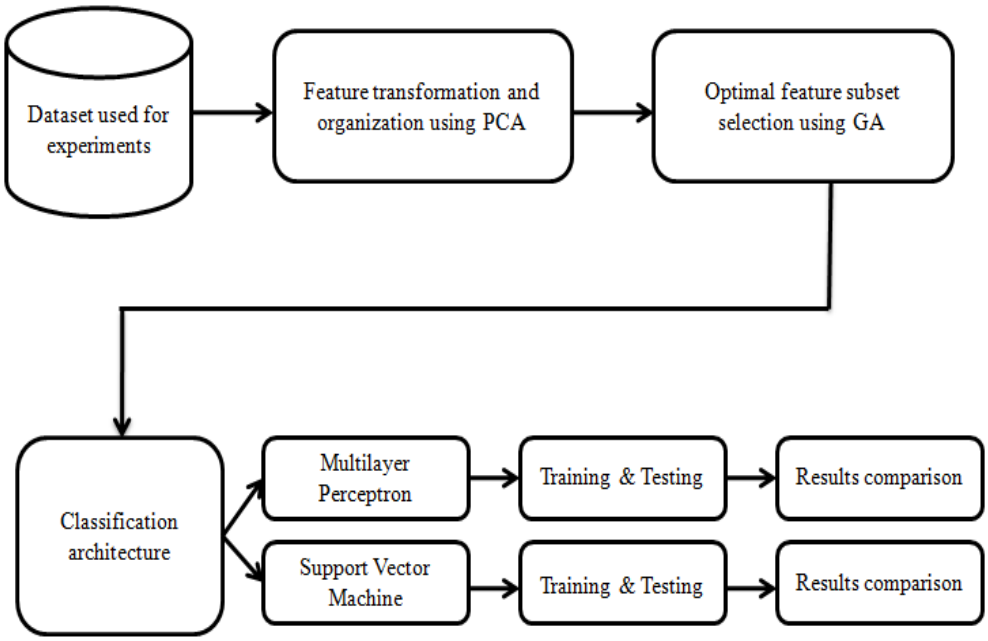


Figure 4.1 Block diagram of proposed model

4.2.1 Dataset used for Experiments

This research work used KDD cup 99 dataset for my experiments. The selection of this dataset is due to its standardization, content richness and it helps to evaluate my results with existing researches in the area of intrusion detection. This has already been described in Chapter 3 . The raw dataset consists of 41 features.

$$x_1, x_2, \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots x_n \qquad \text{Where } n=41 \qquad (4.1)$$

4.2.2 Dataset Pre-processing for Experiments

After selection of the dataset, the raw dataset is pre-processed so that it can be given to the selected classifiers; MLP and SVM. The raw dataset is pre-processed in three ways; (i) discarding symbolic values, (ii) feature transformation and organization using PCA, and (iii) optimal features subset selection using GA.

5.2.2.1 Discarding Symbolic Values

In first step of pre-processing, three symbolic values (e.g. udp, private & SF) are discarded out of 41 features of the dataset. The resultant features are;

$$x_1, x_2, \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots x_m \tag{4.2}$$

Where m=38

4.2.2.2 Feature Transformation and Organization

In second step of pre-processing, PCA has applied on 38 features of the dataset. Mostly, PCA is used for data reduction, but here, PCA is used for feature transformation into principal components feature space and then organized principal components in descending order.

$$pc_1 > pc_2 > pc_3 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots > pc_l \tag{4.3}$$

Where l=38

4.2.2.3 Optimal Feature Subset Selection

In third step of pre-processing, GA is applied for optimal features subset selection from principal components search space. Four different experiments are performed as described in Chapter 3 and selected a subset of ten features that indicated better performance as compared to others. The aim is to select minimum features that produce optimal results in accuracy. This definitely impact on overall performance of the system.

$$\mathrm{pc}_3, \mathrm{pc}_4, \mathrm{pc}_{12}, \mathrm{pc}_{22}, \mathrm{pc}_{26}, \mathrm{pc}_{27}, \mathrm{pc}_{28}, \mathrm{pc}_{32}, \mathrm{pc}_{34}, \mathrm{pc}_{36} \tag{4.4}$$

Ten different principal components are selected using GA process

After features subset selection, this approach used this dataset for training and testing in the experiments. The features are reduced to 10 from the 41 raw features set.

4.2.3 Classification Architectures

This work used two classifier approaches; MLP and SVM as an analysis engine for intrusion classification into *normal* and *intrusive*. These both approaches are commonly used in different areas due to their effective discrimination power as described in chapter 3. This section explains the MLP and the SVM architectures applied for experiments.

4.2.3.1 Multilayer Perceptron (MLP)

A MLP is a feed forward neural network that maps sets of input data onto a set of appropriate output. Here, a MLP architecture is used that consists of three layers; input, hidden and output. In this architecture, hidden layer and output layer consist of neurons (processing elements) and each neuron has a nonlinear activation function. The layers are fully connected from one layer to the next. MLP is an amendment of the standard linear perceptron, which can discriminate data that is not linearly separable. The MLP architecture, used in this work is shown in Figure 4.2.

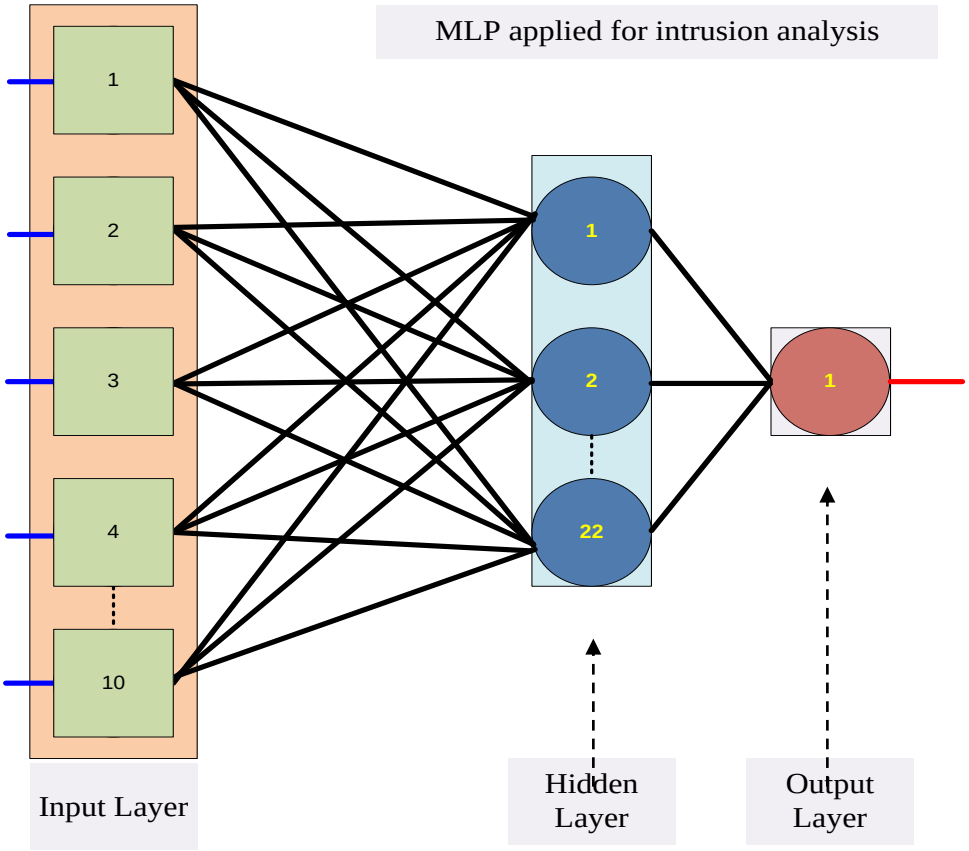


Figure 4.2 MLP architecture

MLP network used to make basic input-output mapping. MLP network is trained in such way that, it produces value of 1 if the presented input pattern is *intrusive* and 0 if the presented input pattern is *normal* network packet. This section describes the main components; layers and synapses of above MLP architecture. The architecture consists of three layers; input, hidden and output that are connected through synapses.

a). Input Layer

The input layer takes input from the input file that contains dataset for training of the network. The row of the dataset is called a pattern representing an instance of the input dataset. The neural network reads and elaborates sequentially all the input rows, and for each one it generates an output pattern representing the outcome of the entire process. For this purpose, an axon is used that has its activation function as;

$$f(x_i, \omega_i) = x_i \tag{4.5}$$

Where x_i input, $\omega_i$associated weight and output of input layer is x_i

b). Hidden Layer

This model uses one hidden layer in MLP architecture that represents a good non-linear element of the neural network. The *TanhAxon* is used as hidden layer in the architecture. It can also be used to build whatever layer (hidden or output) of a neural network. This hidden layer takes inputs from the outputs of the input layer, and applies its activation function. Then, it sends its output to the output layer. The *TanhAxon* applies a *bias* and *tanh* function to each neuron in the layer. This will squash the range of each neuron in the layer to between -1 and 1. Such nonlinear elements provide a network with the ability to make soft decisions. For this purpose, the MLP used *TanhAxon* as shown in Figure 4.*3* that has its activation function as;

$$f(x_i, \omega_i) = \tanh[x_i^{\text{lin}}] \tag{4.6}$$

Where $x_i^{lin} = \beta x_i$ is the scaled and offset activity inherited from the LinearAxon and β parameter represent slope which is not adaptive

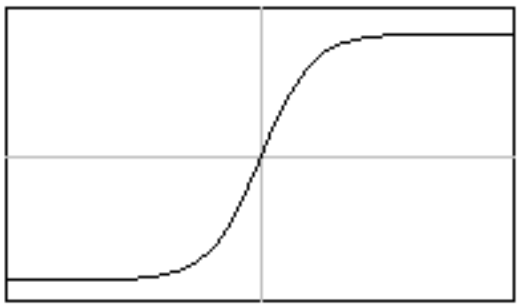


Figure 4.3 Activation function of Tanh

c). Output Layer

The output layer allows a neural network to write output patterns in a file that are used for analysis of intrusion. For this purpose, MLP used *TanhAxon* that has already described above in hidden layer section.

d). Synapses

The *synapse* represents the connection between two layers, permitting a pattern to be passed from one layer to another. The *synapse* is also the 'memory' of a neural network. During the training process the weight of each connection is modified according the learning algorithm. The layers are fully connected with each other. For this purpose *FullSynapse* object is used that connects all the nodes of a layer (axon) with all the nodes of the other layer (axon), as showed in Figure 4.2.

Since each *axon* represents a vector of PEs, the *FullSynapse* simply performs a matrix multiplication. For each PE in its output *axon*, the *FullSynaps*e accumulates a weighted sum of activations from all neurons in its input axons. The activation function is described here.

$$f\left(x_i(t-d), \omega_{ij}\right) = \omega_{ij} x_j(t-d) \tag{4.7}$$

Where ω_{ij} is a connection weight linking PEj to PEi . Time t is discrete, and it relates to one simulation step and discrete time delay is d.

e). Training algorithms

This work used backpropagation algorithm that is one of the most popular supervised learning algorithms (Ahmad et al. 2008). The algorithm consists of two phases: f*orward phase and backward phase*. In the forward phase, first, the weights of the network are randomly initialized. Then, the input signals are propagated through the

network. Afterwards, the output of the network is calculated and compared to the desired value. In the end of the forward phase, the error of the network is calculated. Error of the output neuron i (ei) is calculated by the formula:

$$e_i = d_i - y_i \tag{4.8}$$

Where di is the desired response and yi is the output produces by the neural network in response to the input xi.

Aim of the backpropagation algorithm is to reach global minimum value on the error surface as shown in Figure 4.4.

Figure 4.4 Global and local minimum in error surface

In backward phase, calculated error signal is propagated backward and in order to minimize the error, weights are updated. Change in weights can be calculated by gradient descent learning rule .According to the gradient descent learning rule, correction applied to the weight wji at the iteration *n* is denoted by $\Delta\omega_{ji}$(n), and calculated by

$$\Delta\omega_{ji}(\mathrm{n}) = \eta\delta_{\mathrm{j}}(\mathrm{n})\mathrm{y_i}\ (\mathrm{n}) \tag{4.9}$$

Where η is a numerical constant (learning-rate parameter of the backpropagation algorithm) and $\delta_j(n)$ is local gradient.

Local gradient of output neurons is equal to the product of the derivative of activation function, $\mathrm{f}'(.)$), and error signal,$e_j(n)$, and defined by

$$\delta_{\mathrm{j}}(\mathrm{n}) = \mathrm{e_j}(\mathrm{n})\mathrm{f}'(\mathrm{s_j}(n)) \tag{4.10}$$

*Where error signal is $e_j(n)$ and $\delta_j(n)$ is local gradi*ent.

Local gradient for neurons in hidden layer is defined by

$$\delta_{\mathrm{j}}(\mathrm{n}) = \mathrm{f}'(\mathrm{s_j}(n))\ \textstyle\sum_k \delta_{\mathrm{k}}(\mathrm{n})\mathrm{w_{kj}}\ (n) \tag{4.11}$$

Learning rate parameter, η, is used to reduce the training time. But if the learning rate parameter is chosen too high (e.g. 0.9), algorithm oscillate between local minimums, and may not achieved to reach the global minimum, whereas selecting learning rate too small results in long training periods.

One way to speed up the learning when learning rate is chosen small or avoid oscillation between local minimums when learning rate is chosen to big is to utilize a parameter, *momentum*. By introducing the momentum parameter, change in weight, Δwji(n), is made dependent to the previous weight change, Δwji(n-1). Modified backpropagation algorithm which uses momentum, α, is given;

$$\Delta\mathrm{w_{ij}}(\mathrm{n}) = \eta\delta_{\mathrm{j}}(\mathrm{n})\mathrm{y}(\mathrm{n}) + \alpha\Delta\mathrm{w_{ij}}\ (\mathrm{n}-1) \tag{4.12}$$

After the training was completed, connection weights are frozen. Afterwards, in order to validate whether the neural network was trained sufficiently or not, a test set, which is not part of the training set, was presented to the trained network and its

performance is evaluated. Backpropagation algorithm is simple to implement. However, when dealing with difficult learning tasks, training time of the backpropagation networks can be lengthy and even algorithm may not converge to the desired error rate. The pseudo code of the backpropagation algorithm is given in Figure 4.5.

Backpropagation Algorithm:

Input: training-examples, η, Ø, net,

Output: trained network

Initialize all weights of net;

for each pair < $\vec{x}, \vec{t}$ > ϵ training-examples **do**

Step 1 Forward phase;

Present the input $\vec{x}$ to the input layer of the net;

for each unit $u \epsilon net$ **do**

calculate the output o_u of unit u;

Step 2 Backward phase:

Calculate errors:

for each unit $k \epsilon$ output layer, calculate its error δ_k **do**

$$\delta_k \leftarrow o_k\ (1 - o_k)\ (t - o_k);$$

for each hidden unit h, calculate its errors δ_h **do**

$$\delta_h \leftarrow o_h\ (1 \leftarrow o_h)\ \sum_{k \epsilon outputs}^{\infty} \omega khk;$$

Update weights:

for each weight $\omega_{ij} \epsilon$ net **do**

$$\Delta\omega_{ij}(t+1) = \alpha\delta x_{ji} + \eta\, \Delta\omega_{ij}\ (t);$$

$$\omega_{ij}(t+1) = \omega_{ij}(t) + \Delta\omega_{ij}\ (t+1);$$

Figure 4.5 Backpropagation Algorithm

There is another algorithm used in my implementation known as Levenberg-Marquardt (LM) algorithm that is one of the most appropriate higher-order adaptive algorithms used for minimizing the MSE of a neural network (Hagan and Menhaj 1994). It can be used to update the weights in the network just as backpropagation algorithm. It is reputably the fastest algorithm available for such training. The Levenberg-Marquardt algorithm is designed specifically to minimize the sum-of-squares error function, using a formula that assumes that the underlying function modelled by the network is linear. A move is only accepted if it improves the error, and if necessary the gradient-descent model is used with a sufficiently small step to guarantee downhill movement. The weight update vector $\Delta\underline{X}$ is calculated as

$$\Delta\underline{X} = \left[J^T(\underline{X})J(\underline{x}) + \mu I\right]^{-1} J^T(X)\varepsilon \tag{4.13}$$

Where ε is the vector of errors, μ is the learning rate parameter, and J(x) is the Jacobian matrix that is the matrix of partial derivatives of the errors with respect to the weights. Jacobian matrix can be calculated with the following formula:

$$J(\underline{x}) = \begin{bmatrix} \frac{\partial e_1(\underline{x})}{\partial X_1} & \frac{\partial e_1(\underline{x})}{\partial X_2} & \cdots & \frac{\partial e_1(\underline{x})}{\partial X_n} \\ \frac{\partial e_2(\underline{x})}{\partial X_1} & \frac{\partial e_2(\underline{x})}{\partial X_2} & \cdots & \frac{\partial e_2(\underline{x})}{\partial X_n} \\ \cdots & & & \\ \frac{\partial e_m(\underline{x})}{\partial X_1} & \frac{\partial e_m(\underline{x})}{\partial X_2} & & \frac{\partial e_m(\underline{x})}{\partial X_n} \end{bmatrix} \tag{4.14}$$

Levenberg-Marquardt outperforms the basic backpropagation and its variations with variable learning rate in terms of training time and accuracy. However the computation and memory requirements of the algorithm are high.

4.2.3.2 Support Vector Machine (SVM)

Support vector machines (SVMs) are a very different type of classifier that have attracted a great deal of attention recently due to the novelty of the concepts that they

bring to pattern recognition, their strong mathematical foundation, and their excellent results in practical problems. There are two motivating concepts behind SVMs:

- The idea that transforms the data into a high- dimensional space makes linear discriminant functions practical.
- The idea of large margin classifiers to train the perceptron.

This research work used these two concepts and created the SVM. The advantage of a kernel machine is that its capacity (number of degrees of freedom) is decoupled from the size of the input space. By mapping the input to a sufficiently large feature space, patterns become linearly separable, so a simple perceptron in feature space can do the classification. Here SVM used the Radial basis function (RBF) network, which can be considered a kernel classifier. Actually, the RBF places Gaussian kernels over the data and linearly weights their outputs to create the system output.

When used as an SVM, the RBF network places a Gaussian in each data sample such that the feature space becomes as large as the number of samples. But an SVM is much more than an RBF. To train RBF network as an SVM, this work use the idea of large margin classifiers which uses the Adatron algorithm, which works only with perceptrons. Training an RBF for large margins will decouple the capacity of the classifier from the input space and at the same time provides good generalization. This approach directs towards powerful classifiers. The adatron algorithm can be extended here in two ways: (i) apply it to kernel-based classifiers such as RBFs, and (ii) modify the training for nonlinearly separable patterns.

a). Extension of the Adatron to Kernel Machines

The Adatron algorithm is able to adapt the perceptron to maximize its margin. The idea is to work with data-dependent representations, which lead to a very simple on-line algorithm to adapt the multipliers. The discriminant function of the RBF in terms of the data-dependent representation is given in the Equation:

$$g(x) = \sum_{k=1}^{L} w_k \, G_k(x, \sigma^2) + b =< \sum_{i=0}^{N} \alpha_i \, G(x, \sigma^2). \, G(x_i, \sigma^2) > + b$$

$$g(x) = \sum_{i=0}^{N} \alpha_i \, G\left(x - x_{i,} \, 2\sigma^2\right) + b \tag{4.15}$$

Where $G(x, \sigma^2)$ represents a Gaussian function, L is the number of PEs in the RBF, w_k are the weights, N is the number of samples, α_i are a set of multipliers (one for each sample), and this approach consider the input space augmented by one dimension with a constant value of 1 to provide the bias.

The inner product of Gaussians is a Gaussian. The kernel function (the Gaussian) first projects the inputs (x, xi) onto a high-dimensional space and then computes an inner product there. The amazing thing is that the Gaussian kernel avoids the explicit computation of the pattern projections into the high-dimensional space, as shown in Eq. 5.13 (the inner product of Gaussians is still a Gaussian). Any other symmetric function that obeys the Mercer condition has the same properties. The topology used in this work is depicted in Figure 5.5, where one can easily see that it is an RBF, but where each Gaussian is centered at each sample and the weights are the multipliersα_i.

The adatron algorithm can be easily extended to the RBF network by substituting the inner product of patterns in the input space by the kernel function, leading to the following quadratic optimization problem:

$$J(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \, \alpha_j d_i d_j G\left(x - x_j, 2\sigma^2\right) \tag{4.16}$$

Subject to $\sum_{i=1}^{N} d_i \alpha_i = 0 \;\; \alpha_i \geq 0 \;\; \forall \, i \not\exists \in \{1, \ldots \; N\}$

This can define as,

$$g(x_i) = d_i \left(\sum_{j=1}^{N} d_j , \alpha_j G\left(x_i - x_j, 2\sigma^2\right) + b \quad \text{and } M = \min_i g(x_i)\right) \tag{4.17}$$

and choose a common starting multiplier (e.g. α_i =0.1), learning rate η, and a small threshold (e.g., t = 0.01).

While M>t, this approach choose a pattern x_i and calculate an update $\Delta\alpha_i = \eta(1 - g(x))$ and perform the update

$$\begin{cases} \alpha_i(n+1) = \alpha_i(n) + \Delta\alpha_i(n), b(n+1) = b(n) + \\ \quad d_i \Delta\alpha_i \; if \; \alpha_i(n) + \Delta\alpha_i > 0 \\ \\ \alpha_i(n+1) = \alpha_i(n), \quad b(n+1) = b(n) \\ \\ \qquad if \; \alpha_i(n) + \qquad \Delta\alpha_i > 0 \end{cases} \tag{4.18}$$

After adaptation, only some of the α_i are different from zero (called the support vectors). They correspond to the samples that are closest to the boundary between classes. This algorithm is called the *kernel adatron* and can adapt an RBF to have an optimal margin. This algorithm can be considered the "on-line" version of the quadratic optimization approach utilized for SVMs, and it can find the same solutions as Vapnik's original algorithm for SVMs. It is easy to implement the kernel adatron algorithm since *g*(*x*i) can be computed locally to each multiplier, provided that the desired response is available in the input file. In fact, the expression for *g*(*x*i) resembles the multiplication of an error with an activation, so it can be included in the framework of neural network learning. The Adatron algorithm essentially prunes the RBF network of Figure 4.5 so that its output for testing is given.

$$f(x) = sgn\,(\sum_{i\epsilon\ SVs} d_i\,\alpha_i G(x - x_i, 2\sigma^2) - b) \tag{4.19}$$

b). The Adatron with a Soft Margin

If the patterns in feature space are not linearly separable than an idea is introduce a soft margin using a slack variable $\xi_i = 0$ and a function $F(\xi) = \sum_{i=1}^{N} \xi_i$, which penalize the cost function.

Further minimize the function F, but now subject to the constraints $d_i(w.x_i + b) = 1 - \xi_i\ ,\ where\ i = 1, \ldots\ldots\ N$ and $w.w = c_n$

The new cost function becomes

$$J(\alpha, C) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2C}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i\,\alpha_j\left(d_i d_j G\left(x_i - x_j, 2\sigma^2\right)\right) - \frac{c_n C}{2} \tag{4.20}$$

$$subject\ to \sum_{i=1}^{N} d_j\alpha_i = 0$$

Where $1 \geq \alpha_i \geq 0, \forall i\ \in \{1, \ldots\ldots N\}\quad C \geq 0$

Normally, instead of computing the optimal C, this method chooses a value. A priori C can be regarded as a regularizer. This means that the matrix of kernel inner products is augmented in the diagonal by the factor $1/C$, that is,

$$if\ i = j\ \Omega\left(x_i, x_j\right) = G\left(x_i, x_j\right) + 1/C\ \ else\quad \Omega\left(x_i, x_j\right) = G\left(x_i, x_j\right) \tag{4.21}$$

The only difference in the algorithm for this case is the calculation of $g(x_i)$ which becomes

$$g(x_i) = d_i(\sum_{j=1}^{N} d_j\,\alpha_i\Omega\left(x_i, x_j\right) + b) \tag{4.22}$$

These calculations can be easily implemented as an iterative algorithm, but large data sets produce very large RBF network (one gaussian per data). One disadvantage is that it does not directly specify the number of support vectors to solve the problem. In principal, SVMs should be sensitive to outlier, even when using soft computing. Therefore, this approach used the SVM to transform the data into high-dimensional space using RBF that places a gaussian at each data sample. The RBF uses the backpropagation to train a linear combination of the gaussian to produce the results. The SVM used in my work, however, uses the idea of large margin classifiers for training. This decouples the capacity of the classifier from the input space and at the same time provides good generalization. This is an ideal combination for classification.

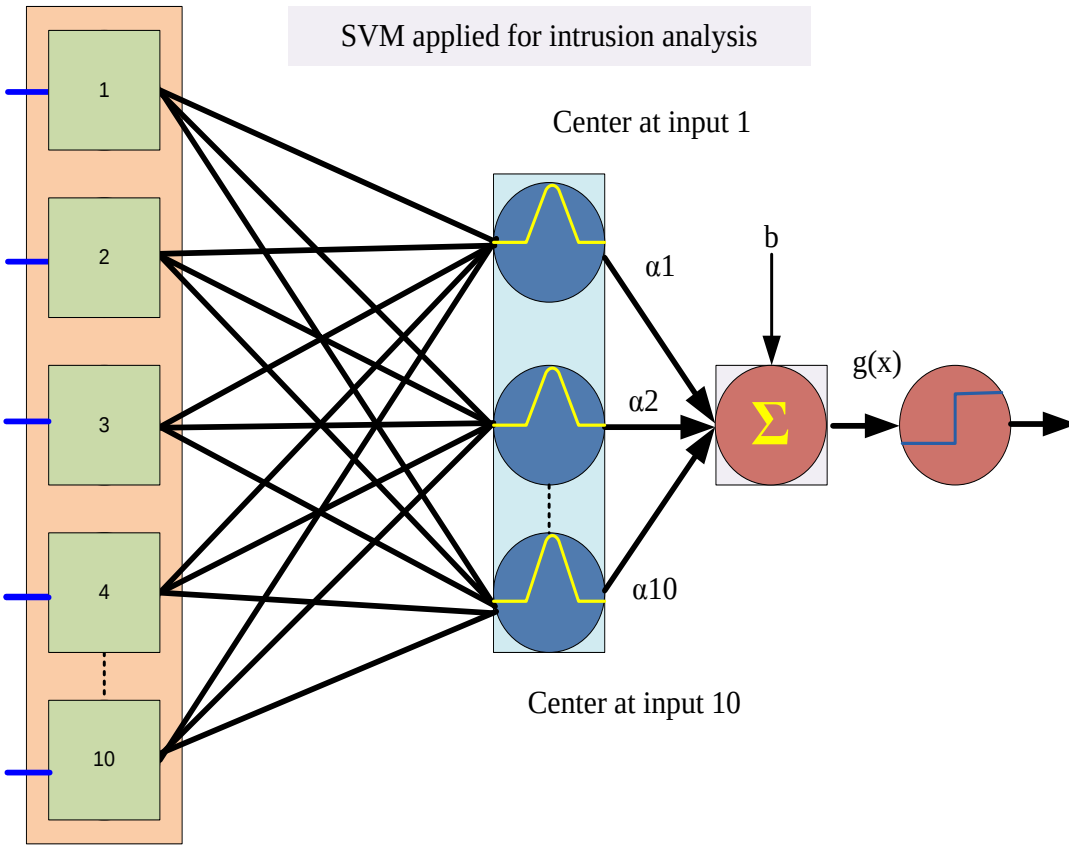


Figure 4.6 SVM applied for intrusion analysis

The SVM is implemented using the kernel adatron algorithm. The kernel adatron maps inputs to a high-dimensional feature space, and then optimally separates data into their respective classes by isolating those inputs which fall close to the data

boundaries. Therefore, the kernel Adatron is especially effective in separating sets of data which share complex boundaries. The kernel Adatron algorithm is given in Figure 4.7.

Kernel Adatron Algorithm:

Step 1: Initialize $\alpha_i = 1$.

Step 2: Starting from pattern $i = 1$, for labeled points

(x_i, y_i) Calculate $z_i = y_i \sum_{j=1}^{p} a_j y_j k(x_i, x_j)$.

Step 3: For all patterns i calculate $\gamma_i = y_i z_i$ and execute steps **4** to **5** below.

Step 4: Let $\delta\alpha^i = \eta(1 - \gamma^i)$ be the proposed change to the multipliers α^i.

Step 5.1: If $(\alpha^i + \delta\alpha^i) \leq 0$ then the proposed change to the multipliers would result in a negative α_i. Consequently to avoid this problem set $\alpha_i = 0$

Step 5.2: If $(\alpha^i + \delta\alpha^i) > 0$ then the multipliers are updated through the addition of the $\delta\alpha^i$ i.e. $\alpha^i \leftarrow \alpha^i + \delta\alpha^i$.

Step 6: Calculate the bias b from

$$b = \frac{1}{2}(min\ (z_i^+) + max\ (z_i^-))$$

Where z_i^+ are those patterns i with class label $+1$ and z_i^- are those with class label -1.

Step 7: If a maximum number of presentations of the pattern set has been exceeded then stop, otherwise return to step 2.

Figure 4.7 Kernel Adatron Algorithm

4.2.4 Implementation

The proposed model is implemented using different types of softwares: NeuroSolutions, PCA and MS-excel. This work implemented the MLP and the SVM in two different modules. This section explains briefly their implemented architectures.

4.2.4.1 MLP Implementation

The MLP architecture consists of different components is shown in Figure 4.8. This section describes the components that constitute multilayered perceptron neural network architecture. Further, these components are explained in Table 4.1.

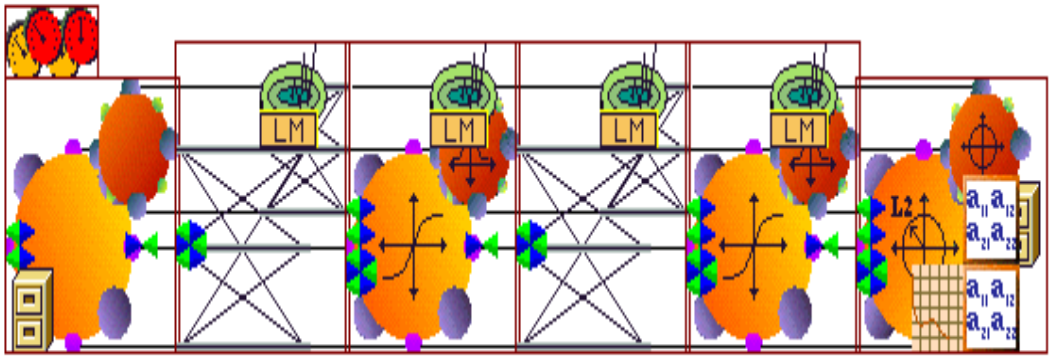


Figure 4.8 MLP implemented architecture

Table 4.1 Components of MLP architecture

S.No	Components	Description
1	StaticControl	It controls the forward activation phase of network.
2	BackStaticControl	It controls the backward activation phase of network (backpropagation).
3	Axon	It is a layer of PE's with identity transfer function. It can act as a placeholder for the File component at the input layer.

S.No	Components	Description
4	File	It is used for network input and desired data from a file.
5	FullSynapse	It is a Full matrix multiplication and it is used to connect two axon layers.
6	BackFullSynapse	It is a Back full matrix multiplication. Attaches to "dual" forward FullSynapse, for use in backpropagation network.
7	Levenberg-Marquardt (LM)	This pseudo second-order learning algorithm tends to train in fewer epochs and arrive a lower error. The Levenberg-Marquardt (LM) algorithm is one of the most appropriate higher-order adaptive algorithms known for minimizing the MSE of a neural network
8	TanhAxon	It is Layer of PE's with hyperbolic transfer function (output range –1 to 1).
9	BackTanhAxon	It is Layer of PE's with transfer function that is the derivative of the TanhAxon. It Attaches to "dual" forward TanhAxon, for use in backpropagation network.
10	L2Criterion	It is Square error criterion. Computes the error between the output and desired signal, and passes it to the backpropagation network.
11	BackCriteriaControl	It provides Input to backpropagation network. It Attaches to Criterion, for use in backpropagation network. Receives error from Criterion.
12	MatrixViewer	It is a Numerical probe. Displays numerical values at the current instant in time.
13	DataGraph	It is Graphing probe. Displays data versus time.
14	File	It is used for network input and desired data from a file.

4.2.4.1 SVM Implementation

The SVM architecture consists of different components is shown in Figure 4.9. This section describes the components that constitute SVM network architecture.

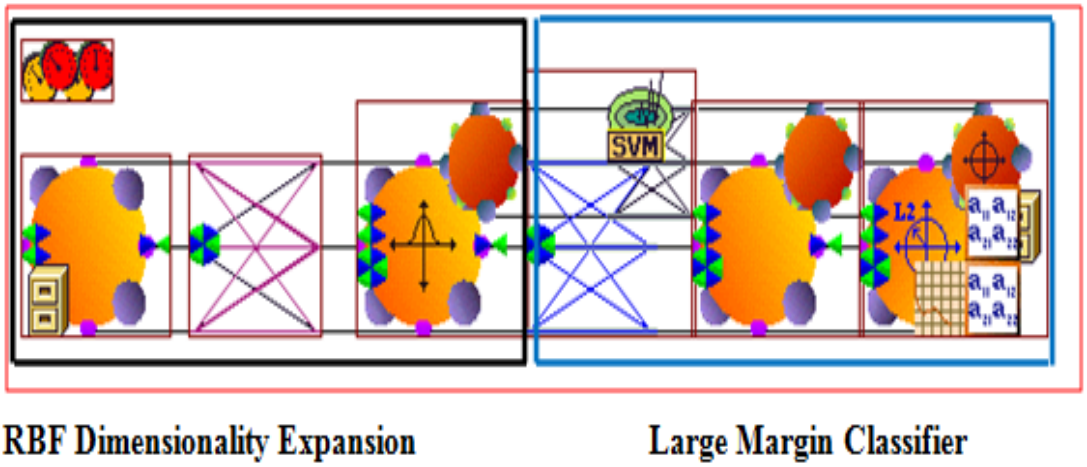


Figure 4.9 SVM implemented architecture

Table 4.2 Components of SVM architecture

S.No	Components	Description
1	StaticControl	It controls the forward activation phase of network.
2	BackStaticControl	It controls the backward activation phase of network (backpropagation).
3	Axon	It is a Layer of PE's with identity transfer function.
4	File	It is used for network input and desired data.
5	FullSynapse	It is a Full matrix multiplication and it is used to connect two axon layers.
6	GaussianAxon	The GaussianAxon implements a radial basis function.
7	Back axon	The back axon Attaches to "dual" forward Axon, for use in backpropagation network.

S.No	Components	Description
8	SVM Output Synapse	This is used to implement the "Large Margin Classifier" segment of the SVM model.
9	BackFullSynapse	It is a Back full matrix multiplication. Attaches to "dual" forward FullSynapse, for use in backpropagation network.
10	SVM	This component is used to implement the "Large Margin Classifier" segment of the SVM model.
11	L2Criterion	L2Criterion is Square error criterion. Computes the error between the output and desired signal, and passes it to the backpropagation network.
12	BackCriteriaControl	It provides Input to backpropagation network. It Attaches to Criterion, for use in backpropagation network. Receives error from Criterion.
13	MatrixViewer	MatrixViewer is a Numerical probe. Displays numerical values at the current instant in time.
14	DataGraph	DataGraph is Graphing probe. Displays data versus time.
15	File	File component is used for network input and desired data from a file.

4.2.5 Training and Testing of the System

The evaluation of system's performance consists of two phases; training and testing as described in chapter 3. This section describes these two phases and their parametric setting during training and test of the system.

4.2.5.1 Training the System

The system is trained on labeled data set such as *intrusive* and *normal*. The aim of training is the adjustment of networks weights on base of the difference between the

output produced by the system and the desired output. This process of weights adjustment and training is called learning. The parameters used in my experiments to tune the neural network are given in the following tables. The parametric specification used for MLP architecture during training phase is given in Table 4.3.

Table 4.3 MLP tuning parameters during training

S.No	Parameter Name	Value
1	Architecture	MLP Feedforward
2	Layers	03 (input, hidden and output)
3	Input samples features	38 (original), 22 (PCA), and 10 (GA)
4	PEs in Input layer	It depends on features subset selections. For examples; 38, 22, & 10.
5	PEs in Hidden Layer	If number of features are 10 than PEs are 22 in hidden layer.
6	Epochs	1000
7	PE in output layer	One that has value 0 and 1
8	Activation function	Tanh
9	Training algorithm	Backpropagation (Forward & Backward)
10	Training dataset	5000 connections in which 20% for cross-validation and 30% for testing
11	Production dataset	20,000 connections

The parametric specification used for SVM architecture during training phase is given in Table 4.4.

Table 4.4 SVM parameters during training

S.No	Parameter Name	Value
1	Architecture	SVM
2	Layers	03 (input, gaussian and output)
3	Input samples features	38 (original), 22 (PCA), and 10 (GA)
4	PEs in Input layer	It depends on features subset selections. For examples; 38, 22, & 10.
5	SVM Input Synapse	If input are 10 then its outputs are 2500
6	PEs in Gaussian Layer	If number of features are 10 than PEs are 2500 in gaussian layer.
7	SVM output Synapse	Inputs 2500 and output 1
8	SVM step size	0.01
9	Weight decay	0.01
10	Epochs	1000
11	PE in output layer	One that has value 0 and 1
12	Activation function	Gaussian
13	Training algorithm	Backpropagation (RBF) and Kernel Adatron (SVM)
14	Training dataset	5000 connections in which 20% for cross-validation and 30% for testing
15	Production dataset	20,000 connections

4.2.5.2 Testing the System

When the system is trained well then weights of the system are frozen and performance of the system is evaluated. Testing of trained system involves two steps; (i) verification step, and (ii) generalization step.

a). Verification Step

In verification step, trained system is tested against the data which are used in training. The purpose of the verification step is to investigate how well trained system learned the training patterns in the training dataset. If a system was trained successfully than the outputs produced by the system would be similar to the real outputs. In this research work 30% of the training dataset (5000) is used as verification that is 1500.

b). Generalization Step

In generalization step, testing is conducted with data which is not used in training. The purpose of the generalization step is to measure generalization ability of the trained network. After training, the system only involves computation of the feed forward phase. For this purpose, a production dataset is used that has input data but no desired data.

This work used a dataset of fifteen thousand (15,000) as a production dataset. Further, the system performance is also tested on total dataset (20,000) that consist of both training dataset and production dataset.

The parameters used during MLP testing phase are given in Table 4.5.

Table 4.5 MLP parameters during testing

S.No	Parameter Name	Value
1	Architecture	MLP Feedforward
2	Layers	03 (input, hidden and output)
3	Input samples features	38 (original), 22 (PCA), and 10 (GA)
4	PEs in Input layer	It depends on features subset selections. For examples; 38, 22, & 10.
5	PEs in Hidden Layer	If number of features are 10 than PEs are 22 in hidden layer.
6	Epochs	1
7	PE in output layer	One that has value 0 and 1
8	Activation function	Tanh
9	Supervised/Teacher layer	No need of this layer in testing phase.
10	Training algorithm	NO. But it involves feedforward phase only.
11	Desired dataset	No need of desired dataset in testing phase.
12	Testing dataset	1500 that is 30% of training dataset (5000).
13	Production dataset	20,000 connections

The parametric specification used for SVM architecture during testing phase is given in Table 4.6.

Table 4.6 SVM parameters during Testing

S.No	Parameter Name	Value
1	Architecture	SVM
2	Layers	03 (input, gaussian and output)
3	Input samples features	38 (original), 22 (PCA), and 10 (GA)
4	PEs in Input layer	It depends on features subset selections.
5	SVMInputSynapse	If input are 10 then its outputs are 2500
6	PEs in Gaussian Layer	If number of features are 10 than PEs are 2500 in gaussian layer.
7	SVMoutputSynapse	Inputs 2500 and output 1
8	SVM step size	0.01
9	Weight decay	0.01
10	Epochs	1
11	PE in output layer	One that has value 0 and 1
12	Activation function	Gaussian
13	Supervised/Teacher layer	No need of this layer in testing phase.
14	Training algorithm	Feedforward phase only of Backpropagation (RBF) and Kernel Adatron (SVM)
15	Training dataset	5000 connections in which 20% for cross-validation and 30% for testing
16	Production dataset	20,000 connections

4.2.6 Results Comparison

The results of MLP and SVM based systems and their comparison are discussed in Chapter 5. Further, the detail analysis of results will be discussed as well as their performance comparison with other recent approaches in the area of intrusion detection.

4.3 Summary

This chapter decomposes the methodology described in Chapter 3 in general and the special focus being classification architectures in particular. This is represented using block diagram, mathematical foundations. Then the block diagram is treated block by block, as a phase and details of each part or block of its main architecture. The chapter then explained the features description of the dataset used for experiments, feature transformation process using PCA and optimal features subset selection using GA. Thus, the chapter describes the details of classification architectures with basic algorithms and mathematical foundation of multilayered perceptron model (MLP) and SVM. The explanation continues to the system implementation level, and the basic parameters used during training and testing. The selection of optimal features subset using GA by searching the PCA features space as mentioned before is the main contribution that makes the architecture simple as well as increases the overall performance of the intrusion analysis engine. The application of LM learning rule enhances the MLP training and testing performance as compared to traditional propagation algorithm. The hybrid architecture of SVM that consists of RBF and large margin classifier enhanced its performance in intrusion analysis. The evaluation of these approaches; MLP and SVM will be discussed in Chapter 5.

CHAPTER 5

RESULTS AND DISCUSSION

5.1 Introduction

This chapter describes the experimental results obtained by the utilization of techniques like Principal Component Analysis (PCA), Genetic Algorithm (GA), Multilayer Perceptron (MLP) and Support Vector Machine (SVM) in the proposed network intrusion detection mechanism, whose methodology and designed architecture already covered in Chapters 3 & 4. After the training process was completed, testing was conducted in two steps. In the first step, both classifiers (MLP and SVM) were tested against the training dataset, in order to examine how well the system 'learned' the training dataset after the training process. In the second step of the testing, trained systems were tested against a dataset, which is not a part of the training set, in order to observe generalization performance of the trained systems. In both testing steps, performance of the systems was tested by investigating the number of false positives, false negatives, true positives and the true negatives that they generated.

5.2 Experimental Results

The SVM and MLP architectures implemented independently to conduct different experiments with different scenarios. The implemented systems based on MLP and SVM for network intrusion detection tested on a system having the following specification as shown in Table 5.1.

Table 5.1 System specification for experiments

Hardware/Software	Specification
Operating system	Windows vista with service pack 2
System Manufacturer	Toshiba
Processor	Intel (R) core(TM)2 Duo CPU T5800 2GHz
Memory (RAM)	3.00 GB

The experiments of both classifier architectures; MLP and SVM for intrusion analysis are performed on different size of datasets to testify the proposed mechanism for network intrusion detection. This research work performed several experiments on different feature subsets with GA and without GA. First, the following section present results obtained using MLP architecture. Then, it illustrate results obtained using SVM.

5.2.1 MLP Experimental Results

The MLP based intrusion analysis engine is evaluated on different feature subsets. This section presents MLP results and their sensitivity analysis in different scenarios. First of all, MLP is tested on original dataset without using PCA and GA, which consists of 38 features. Five thousand exemplars or input samples are randomly selected from twenty thousand dataset. Five thousand exemplars contains two types of connections; *normal* and *intrusive*, in which 3,223 are *normal* and 1,777 are *intrusive*. The five thousand dataset is further divided into three subsets; training dataset (2500), cross-validation dataset (1000) and testing dataset (1500).

Scenario 1: MLP with original 38 feature set

- *Testing Phase Analysis*

The purpose of testing phase in scenario 1 is to observe the system's learning ability after the training process with original 38 feature set. The sensitivity results of mentioned above three datasets are shown in Table 5.2-5.4.

Table 5.2 MLP-org-38: Sensitivity analysis of training dataset

True Positive (%)	False Positive (%)	False Negative (%)	True Negative (%)
100	0.0	2.59	97

Table 5.3 MLP-org-38: Sensitivity analysis of cross-validation dataset

True Positive (%)	False Positive (%)	False Negative (%)	True Negative (%)
98.71	1.28	2.25	97.74

Table 5.4 MLP-org-38: Sensitivity analysis of testing dataset

True Positive (%)	False Positive (%)	False Negative (%)	True Negative (%)
97.07	2.92	2.54	97.45

The overall performance of testing phase based one time, epochs, detection rate and false alarm are expressed in Table 5.5.

Table 5.5 MLP-org-38: Overall performance of testing phase

Training Time (H:M:S)	Training Epochs (Number)	Detection rate (%)	False Alarm (%)
1:29:36	1000	97.26	2.73

- *Verification Phase Analysis*

In verification phase, the trained system is tested against a dataset, which is not a part of the training set, in order to examine generalization performance of the trained system. The results are presented in Table 5.6 shows system's performance on production dataset.

Table 5.6 MLP-org-38: Overall performance of verification phase

No. of Features out of 20K	True Positives (Number)	True Negative (Number)	Detection rate (%)		False Alarm (Number)
			Normal (64 %)	Intrusive (36 %)	
760000	1456	18544	7.28	92.72	11344

Scenario 2: MLP with PCA 38 features set

- *Testing Phase Analysis*

The purpose of testing phase is to observe the system how well the system 'learned' the training dataset with PCA38 feature set after the training process. The sensitivity results of above mentioned datasets are presented in Table 5.7-5.9.

Table 5.7 MLP-TF38: Sensitivity analysis of training dataset

True Positive (%)	False Positive (%)	False Negative (%)	True Negative (%)
100	0.0	0.0	100

Table 5.8 MLP-TF38: Sensitivity analysis of cross-validation dataset

True Positive (%)	False Positive (%)	False Negative (%)	True Negative (%)
99.84	0.153	0.0	100

Table 5.9 MLP-TF38: Sensitivity analysis of testing dataset

True Positive (%)	False Positive (%)	False Negative (%)	True Negative (%)
100	0.0	0.0	100

Table 5.10 MLP-TF38: Overall performance of testing phase

Training Time (H:M:S)	Training Epochs (Number)	Detection rate (%)	False Alarm (%)
1:20:07	1000	100	0.0

Table 5.10 shows overall performance of MLP in terms of time, epochs, detection rate and false alarm.

- *Verification Phase Analysis*

In verification phase, the trained system is tested against a dataset, which is not a part of the training set (such as production dataset), in order to examine generalization performance of the trained system. Table 5.11 shows MLP performance on production dataset.

Table 5.11 MLP-TF38: Overall performance of verification phase

No. of Features out of 20K	True Positives (Number)	True Negative (Number)	Detection rate (%)		False Alarm (Number)
			Normal (64 %)	Intrusive (36 %)	
760000	12793	7207	63.965	36.035	07

Scenario 3: MLP with PCA 22 feature set

- *Testing Phase Analysis*

The purpose of testing phase is to look at the system how well the system 'learned' the training dataset in scenario 3 after the training process. The sensitivity analysis of datasets (training, cross-validation & testing) is given in Table 5.12-5.14.

Table 5.12 MLP-PCA22: Sensitivity analysis of training dataset

True Positive (%)	False Positive (%)	False Negative (%)	True Negative (%)
100	0.0	0.0	100

Table 5.13 MLP-PCA22: Sensitivity analysis of cross-validation dataset

True Positive (%)	False Positive (%)	False Negative (%)	True Negative (%)
100	0.0	0.0	100

Table 5.14 MLP-PCA22: Sensitivity analysis of testing dataset

True Positive (%)	False Positive (%)	False Negative (%)	True Negative (%)
100	0.0	0.0	100

The overall performance of testing phase in terms of time, epochs, detection rate and false alarm is shown in Table 5.15.

Table 5.15 MLP-PCA22: Overall performance of testing phase

Training Time (H:M:S)	Training Epochs (Number)	Detection rate (%)	False Alarm (%)
0:53:28	1000	100	0.0

- *Verification Phase Analysis*

In verification phase, the trained system is verified against a dataset, which is not a part of the training set, in order to examine generalization performance of the trained system in this scenario 3. Table 5.16 shows whole performance on production dataset.

Table 5.16 MLP-PCA22: Overall performance of verification phase

No. of Features out of 20K	True Positives (Number)	True Negative (Number)	Detection rate (%)		False Alarm (Number)
			Normal (64 %)	Intrusive (36 %)	
440000	12789	7211	63.945	36.05	11

Scenario 4: MLP with GA12 feature set

- *Testing Phase Analysis*

The purpose of testing phase is to study the system with GA12 feature set in scenario 4 and monitor how well the system 'learned' the training dataset after the training process. The sensitivity analysis of training phase is shown in Table 5.17-5.20.

Table 5.17 MLP-GA12: Sensitivity analysis of training dataset

True Positive (%)	False Positive (%)	False Negative (%)	True Negative (%)
100	0.0	0.0	100

Table 5.18 MLP-GA12: Sensitivity analysis of cross-validation dataset

True Positive (%)	False Positive (%)	False Negative (%)	True Negative (%)
100	0.0	0.0	100

Table 5.19 MLP-GA12:Sensitivity analysis of testing dataset

True Positive (%)	False Positive (%)	False Negative (%)	True Negative (%)
100	0.0	0.0	100

Table 5.20 MLP-GA12: Overall performance of testing phase

Training Time (H:M:S)	Training Epochs (Number)	Detection rate (%)	False Alarm (%)
0:53:28	1000	100	0.0

- *Verification Phase Analysis*

In verification phase, the trained system is assessed against a dataset, which is not a part of the training set, in order to inspect generalization performance of the trained system in this scenario 4. The whole performance of MLP with GA 12 feature set on the production dataset is presented in Table 5.21. This approach used different parameters (number of features, true positives, true negatives, number of normal connections , number of intrusive connections, detection rate and false alarms) to verify the MLP classifier with GA 12 feature set.

Scenario 5: MLP with GA10 feature set

- *Testing Phase Analysis*

Table 5.21 MLP-GA12: Overall performance of verification phase

No. of Features out of 20K	True Positives (Number)	True Negative (Number)	Detection rate (%)		False Alarm (Number)
			Normal (64 %)	Intrusive (36 %)	
440000	12797	7203	63.945	36.05	03

The purpose of testing phase is to study the system with ten features as selected by GA and to observe the behaviour how well the system 'learned' the training dataset after the training process. Table 5.22- 5.25 show sensitivity results of testing phase.

Table 5.22 MLP-GA10: Sensitivity analysis of training dataset

True Positive (%)	False Positive (%)	False Negative (%)	True Negative (%)
100	0.0	0.0	100

Table 5.23 MLP-GA10: Sensitivity analysis of cross-validation dataset

True Positive (%)	False Positive (%)	False Negative (%)	True Negative (%)
100	0.0	0.0	100

Table 5.24 MLP-GA10: Sensitivity analysis of testing dataset

True Positive (%)	False Positive (%)	False Negative (%)	True Negative (%)
100	0.0	0.0	100

Table 5.25 MLP-GA10: Overall performance of testing phase

Training Time (H:M:S)	Training Epochs (Number)	Detection rate (%)	False Alarm (%)
0:23:28	174	100	0.0

- *Verification Phase Analysis*

In verification phase, the trained system is verified on a production dataset in order to check up generalization performance of the trained system. The whole performance of MLP with GA 10 feature set is shown in Table 5.26.

Table 5.26 MLP-GA10: Overall performance of verification phase

No. of Features out of 20K	True Positives (Number)	True Negative (Number)	Detection rate (%)		False Alarm (Number)
			Normal (64 %)	Intrusive (36 %)	
200000	12797	7203	63.985	36.01	03

5.2.2 SVM Experimental Results

Scenario 1: SVM with original 38 feature set

- *Testing Phase Analysis*

The purpose of testing phase is to observe the system how well the system 'learned' the training dataset after the training process. The sensitivity analysis of confusion matrix of testing phase is shown in Table 5.27-5.30.

Table 5.27 SVM-org-38: Sensitivity analysis of training dataset

True Positive (%)	False Positive (%)	False Negative (%)	True Negative (%)
100	0.0	0.0	100

Table 5.28 SVM-org-38: Sensitivity analysis of cross-validation dataset

True Positive (%)	False Positive (%)	False Negative (%)	True Negative (%)
93.65	6.34	2.47	97.52

Table 5.29 SVM-org-38: Sensitivity analysis of testing dataset

True Positive (%)	False Positive (%)	False Negative (%)	True Negative (%)
93.65	6.34	2.47	97.52

Table 5.30 SVM-org-38: Overall performance of testing phase

True Positive (%)	False Positive (%)	False Negative (%)	True Negative (%)
2:21:17	1000	95.585	4.405

In verification phase, the trained SVM with original 38 feature set is tested on production dataset, which is not a part of the training set, in order to observe generalization performance of the trained system. The overall performance of SVM with 38 raw feature set is shown in Table 5.31.

Table 5.31 SVM-org-38: Overall performance of verification phase

No. of Features out of 20K	True Positives (Number)	True Negative (Number)	Detection rate (%)		False Alarm (Number)
			Normal (64 %)	Intrusive (36 %)	
760000	1345	18655	6.72	93.27	11455

Scenario 2: SVM with PCA 38 feature set

- *Testing Phase Analysis*

The purpose of testing phase of SVM with PCA 38 feature set is to monitor the system how well the system 'learned' the training dataset after the training process. The sensitivity analysis of SVM with PCA 38 feature set (transformed feature set) is shown in Table 5.32-5.35.

Table 5.32 SVM-TF38: Sensitivity analysis of training dataset

True Positive (%)	False Positive (%)	False Negative (%)	True Negative (%)
100	0.0	0.0	100

Table 5.33 SVM-TF38: Sensitivity analysis of cross-validation dataset

True Positive (%)	False Positive (%)	False Negative (%)	True Negative (%)
99.07	0.93	0.58	99.42

Table 5.34 SVM-TF38: Sensitivity analysis of testing dataset

True Positive (%)	False Positive (%)	False Negative (%)	True Negative (%)
98.66	1.33	0.759	99.24

Table 5.35 SVM-TF38: Overall performance of testing phase

True Positive (%)	False Positive (%)	False Negative (%)	True Negative (%)
2:39:04	1000	98.95	1.0445

- *Verification Phase Analysis*

In verification phase, the trained system is tested on production dataset, which is not a part of the training set, in order to check generalization performance of the trained system with PCA 38 feature set. The whole performance of SVM with TF38 (transformed feature) set is given in Table 5.36.

Table 5.36 SVM-TF38: Overall performance of verification phase

No. of Features out of 20K	True Positives (Number)	True Negative (Number)	Detection rate (%)		False Alarm (Number)
			Normal (64 %)	Intrusive (36 %)	
760000	12721	7279	63.605	36.395	79

Scenario 3: SVM with PCA 22 feature set

- *Testing Phase Analysis*

The purpose of testing phase is to observe the SVM with PCA 22 feature set how well it 'learned' the training dataset after the training process. The sensitivity analysis of SVM with PCA 22 feature set is shown in Table 5.37-5.40.

Table 5.37 SVM-PCA-22: Sensitivity analysis of training dataset

True Positive (%)	False Positive (%)	False Negative (%)	True Negative (%)
99.37	0.63	0.56	99.44

Table 5.38 SVM-PCA-22: Sensitivity analysis of cross-validation dataset

True Positive (%)	False Positive (%)	False Negative (%)	True Negative (%)
99.50	0.46	0.85	99.14

Table 5.39 SVM-PCA-22: Sensitivity analysis of testing dataset

True Positive (%)	False Positive (%)	False Negative (%)	True Negative (%)
99.48	0.51	0.95	99.05

Table 5.40 SVM-PCA-22: Overall performance of testing phase

Training Time (H:M:S)	Training Epochs (Number)	Detection rate (%)	False Alarm (%)
2:08:18	1000	99.26	0.735

- *Verification Phase Analysis*

In verification phase, the trained system (SVM with PCA 22 feature set) is evaluated against a dataset, which is not a part of the training set (i.e. the production dataset), in order to observe generalization performance of the trained system. The overall performance of this system is given in Table 5.41.

Table 5.41 SVM-PCA-22: Overall performance of verification phase

No. of Features out of 20K	True Positives (Number)	True Negative (Number)	Detection rate (%)		False Alarm (Number)
			Normal (64 %)	Intrusive (36 %)	
440000	12776	7224	63.88	36.12	24

Scenario 4: SVM with GA12 feature set

- *Testing Phase Analysis*

The purpose of testing phase is to observe the system (the SVM with GA 12 feature set) how well the system 'learned' the training dataset after the training process. The sensitivity analysis of testing phase is shown in Table 5.42-5.45.

Table 5.42 SVM-GA12: Sensitivity analysis of training dataset

True Positive (%)	False Positive (%)	False Negative (%)	True Negative (%)
98.30	1.7	0.0	100

Table 5.43 SVM-GA12: Sensitivity analysis of cross-validation dataset

True Positive (%)	False Positive (%)	False Negative (%)	True Negative (%)
100	0.0	0.0	100

Table 5.44 SVM-GA12: Sensitivity analysis of testing dataset

True Positive (%)	False Positive (%)	False Negative (%)	True Negative (%)
99.79	0.2	0.75	99.24

Table 5.45 SVM-GA12: Overall performance of testing phase

Training Time (H:M:S)	Training Epochs (Number)	Detection rate (%)	False Alarm (%)
0:53:28	1000	99.51	0.485

- *Verification Phase Analysis*

In verification phase, the trained system (the SVM with GA12) is tested on production dataset, which is not a part of the training set, in order to observe its generalization performance with GA 12 feature set. The whole performance is given in Table 5.46.

Table 5.46 SVM-GA12: Overall performance of verification phase

No. of Features out of 20K	True Positives (Number)	True Negative (Number)	Detection rate (%)		False Alarm (Number)
			Normal (64 %)	Intrusive (36 %)	
240000	12811	7189	64.055	35.945	11

Scenario 5: SVM with GA10 feature set

- *Testing Phase Analysis*

The purpose of testing phase is to observe the system how well the system 'learned' the training dataset after the training process. The sensitivity analysis of SVM with GA 10 feature set is shown in Table 5.47-5.50.

Table 5.47 SVM-GA10: Sensitivity analysis of training dataset

True Positive (%)	False Positive (%)	False Negative (%)	True Negative (%)
99.38	0.61	0.0	100

Table 5.48 SVM-GA10: Sensitivity analysis of cross-validation dataset

True Positive (%)	False Positive (%)	False Negative (%)	True Negative (%)
99.38	0.61	0.0	100

Table 5.49 SVM-GA10: Sensitivity analysis of testing dataset

True Positive (%)	False Positive (%)	False Negative (%)	True Negative (%)
99.89	0.10	0.94	99.05

Table 5.50 SVM-GA10: Overall performance of testing phase

Training Time (H:M:S)	Training Epochs (Number)	Detection rate (%)	False Alarm (%)
0:16:14	1000	99.47	0.52

- *Verification Phase Analysis*

In verification phase, the trained system (the SVM) is verified against a dataset (the production dataset), which is not a part of the training set, in order to check up generalization performance of the trained system with GA 10 feature set. The overall performance of this scenario is shown in Table 5.51.

Table 5.51 SVM-GA10: Overall performance of verification phase

No. of Features out of 20K	True Positives (Number)	True Negative (Number)	Detection rate (%)		False Alarm (Number)
			Normal (64 %)	Intrusive (36 %)	
200000	12807	7193	64.035	35.965	07

5.2.3 Comparison between MLP and SVM

This section makes a tabular comparative analysis between five different scenarios for MLP and SVM respectively based on above-mentioned results. This comparison is based on number of false alarm, number of epochs, number of features, training time and the results of confusion matrix and is presented in Table 5.52 and Table 5.53.

Table 5.52 MLP performance for intrusion analysis

Classifier	MLP-A10	MLP-A12	MLP-22	MLP-TF38	MLP-org-38
False Alarm	03	03	11	07	11344
Epochs	174	217	1000	1000	1000
Time	00:20:07	00:23:00	01:08:07	01:28:07	01:29:36
Features	200000	240000	440000	760000	760000
False +	03	03	11	07	11344
False -	0	0	0	0	0
True +	12797	12797	12789	12793	1456
True -	7203	7203	7211	7207	18544

Table 5.53 SVM performance for intrusion analysis

Classifier	SVM-GA10	SVM-A12	SVM-F22	SVM-F38	SVM-org-38
False Alarm	07	12	24	79	11455
Epochs	1000	1000	1000	1000	1000
Time	01:16:14	01:36:01	02:08:18	02:39:04	02:21:17
Features	200000	240000	440000	760000	760000
False +	0	0	24	79	11455
False -	01	11	0	0	0
True +	12807	12811	12776	12721	1345
True -	7193	7189	7224	7279	18655

The above comparison proved that my mechanism using GA to search the PCA features space provides optimal performance as compared to traditional way of selecting features from PCA search space. The key focus of my research was to select sensitive features and minimum features as well as to increase accuracy of the system. Thus, research work achieved this objective by using GA and PCA that made the classifier simpler as well as more efficient in performance. Hence, this method shows that proposed method provides an optimal intrusion detection mechanism that outperforms the existing approaches and has the capability to minimize the number of features and maximize the detection rates

5.2.4 Comparative analysis of applied approach with other approaches

This section presents here, a visual comparative analysis of applied approach with other approaches in the literatures (Liu et al. 2007) as described in Chapter 2. The analysis is presented in various graphs using the Multi-criteria Decision Making Technique (MCDM). The main criteria consist of accuracy, minimum training

overhead, more simple architecture, generalization ability, and computational overhead. The main criteria are further divided into sub criteria. The criterion 'accuracy' is subdivided into 'detection rate' and 'minimum false alarm'. The criterion 'more simple architecture' is sub-divided into 'minimum features' and 'minimum processing elements (PEs)'. The criterion 'computational overhead' is divided into 'memory' and 'processing'. The criteria hierarchy is shown in Figure 5.1.

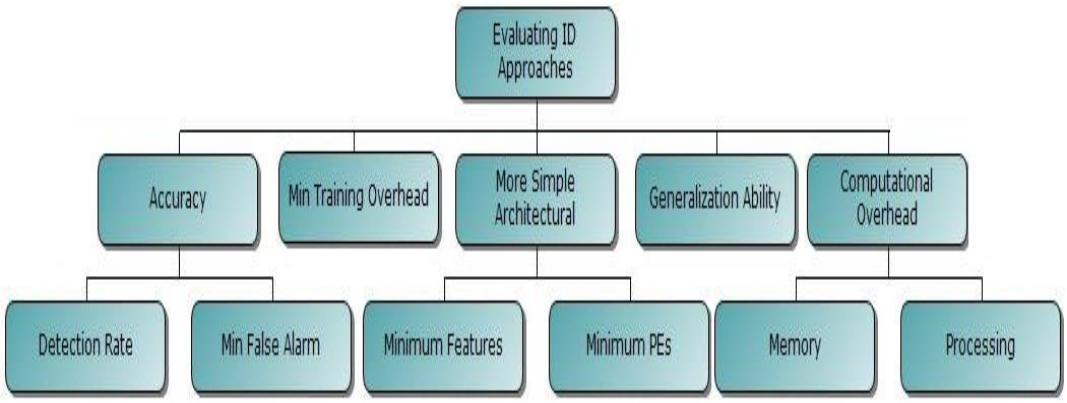


Figure 5.1 Criteria hierarchy

5.2.4.1 Comparative Analysis of applied approach based on MLP with other MLP approaches

This section compares MLP approach based on defined criteria as shown in Figure 5.1 with other approaches based on the obtained results as aforementioned in Table 5.52. The detail comparative analysis is presented in graphs in Figure 5.2 - 5.10.

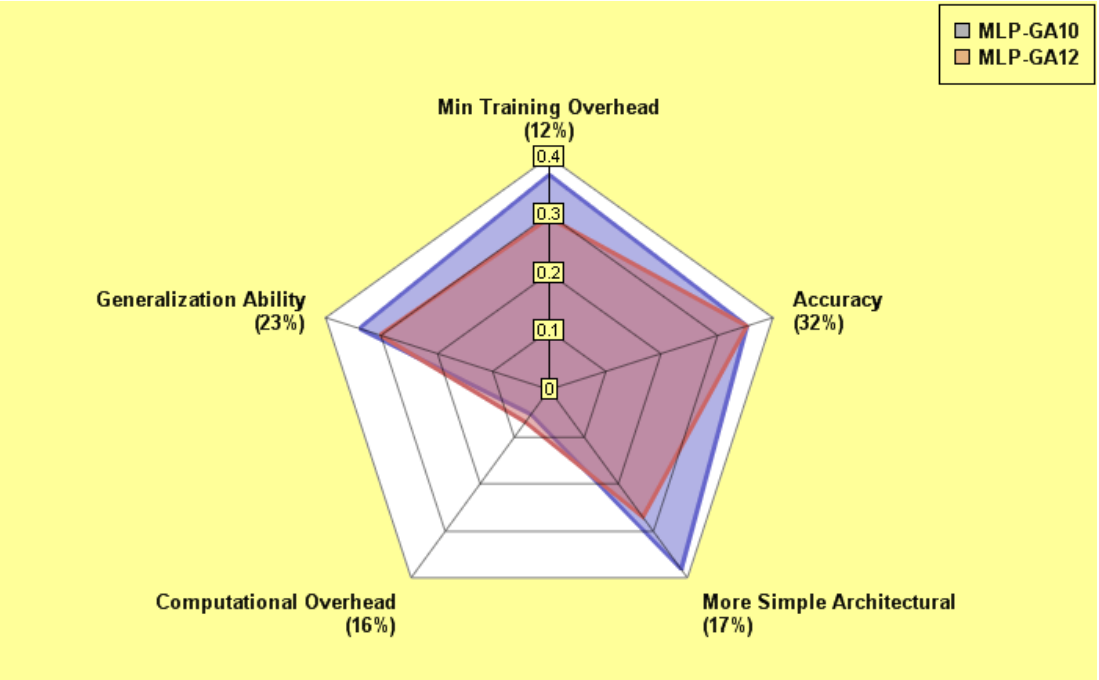


Figure 5.2 MLPGA10 vs. MLPGA12 (criteria)

Figure 5.2 shows MLP results comparison between two datasets: GA 10 feature set and GA 12 feature set based on main criteria. The MLP shows better performance with GA10 feature set as compared to GA12 feature set based on main criteria: accuracy, minimum training overhead, generalization ability, computational overhead and in architectural simplification.

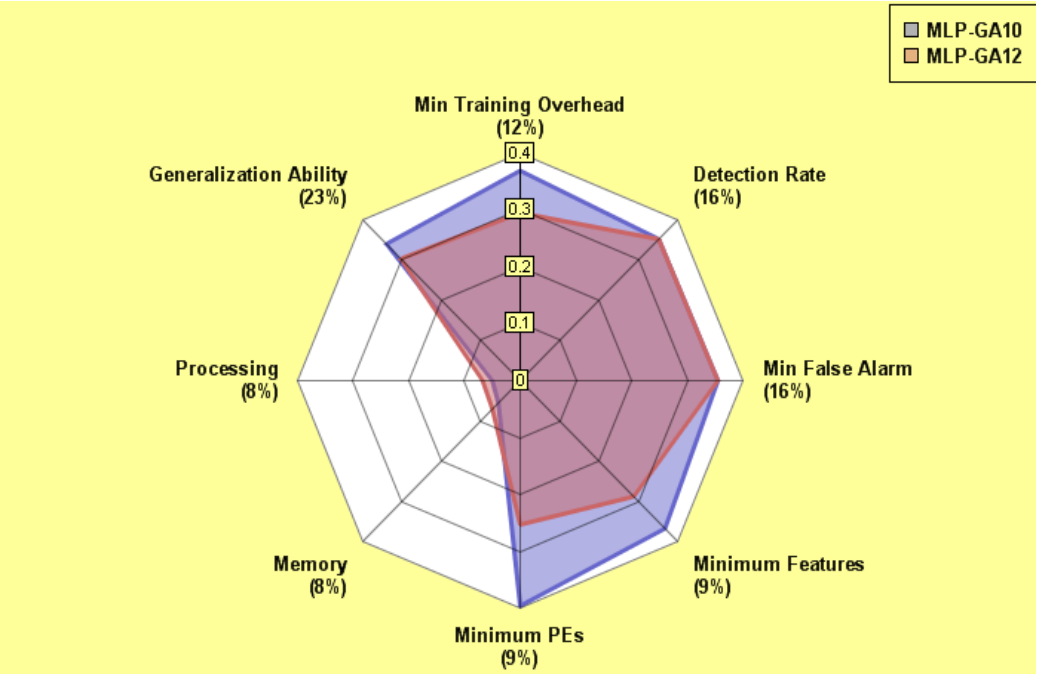


Figure 5.3 MLPGA10 vs. MLPGA12 (sub-criteria)

Figure 5.3 shows MLP results comparison between two datasets: GA 10 feature set and GA 12 feature set based on sub criteria: detection rate, minimum false alarm, minimum features, minimum processing elements (PEs), usage of memory, processing time, generalization ability and minimum training overhead. The use of GA12 feature set increases training and computational overhead as compared to GA10 feature set. Thus, the MLP with GA10 feature set demonstrates better performance as compared to MLP with GA12 feature set based on aforementioned sub criteria.

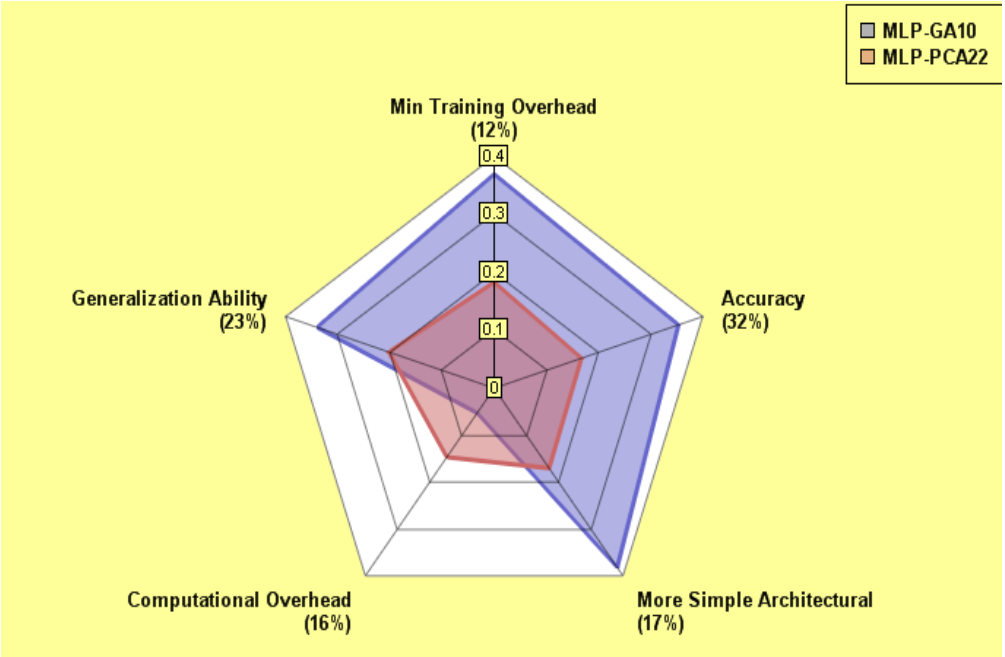


Figure 5.4 MLPGA10 vs. MLP-PCA22 (criteria)

Figure 5.4 shows MLP results comparison between two datasets: GA10 feature set and PCA12 feature set based on main criteria. The MLP shows better performance with GA10 feature set as compared to PCA22 feature set.

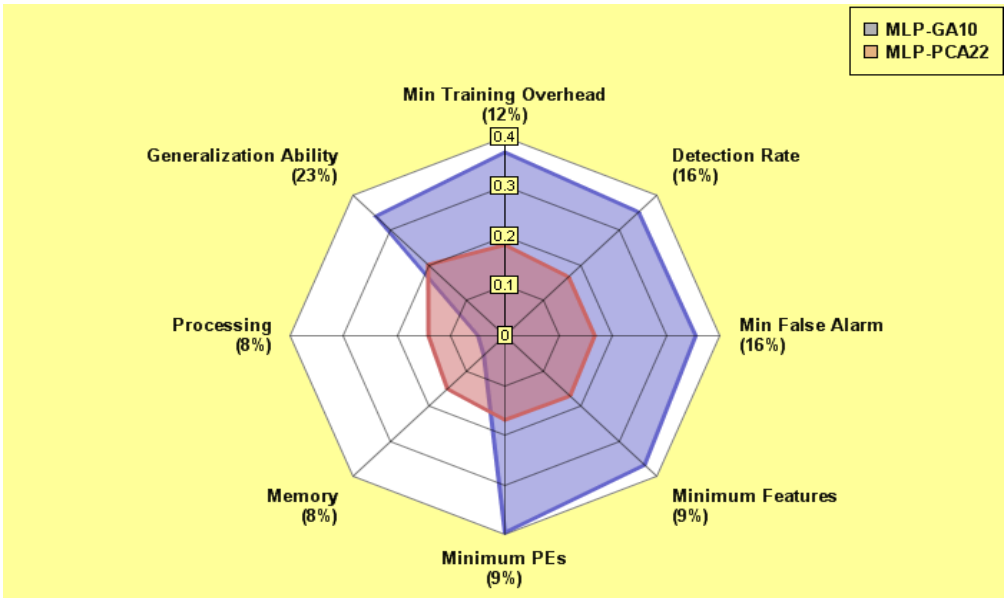


Figure 5.5 MLPGA10 vs. MLP-PCA22 (sub-criteria)

Figure 5.5 shows MLP results comparison between two datasets: GA 10 feature set and PCA22 feature set based on sub criteria: detection rate, minimum false alarm, minimum features, minimum processing elements (PEs), usage of memory, processing time, generalization ability and minimum training overhead. The use of PCA22 feature set increases training and computational overhead as compared to GA10 feature set. Thus, the MLP with GA10 feature set demonstrates better performance as compared to MLP with PCA22 feature set based on aforementioned sub criteria.

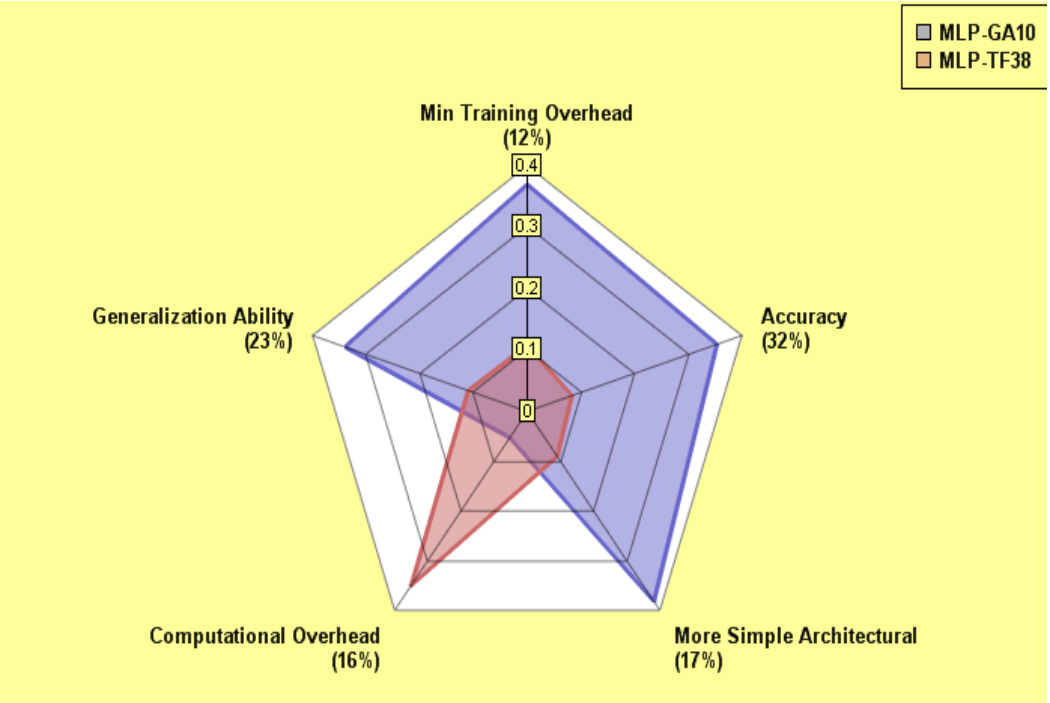


Figure 5.6 MLPGA10 vs. MLPTF38 (criteria)

Figure 5.6 shows MLP results comparison between two datasets: GA 10 feature set and TF38 (Transformed features from raw dataset using PCA) feature set based on main criteria. The computational overhead increases as used TF38 instead of GA10 feature set. Hence, the MLP shows better performance with GA10 feature set as compared to TF38 feature set based on main criteria: accuracy, minimum training overhead, generalization ability, and in architectural simplification.

Figure 5.7 illustrates comparison between MLPs with GA10 and TF38 feature set on the bases of above mentioned sub criteria. The MLP with GA10 outperforms as compared to MLP with TF38 feature set.

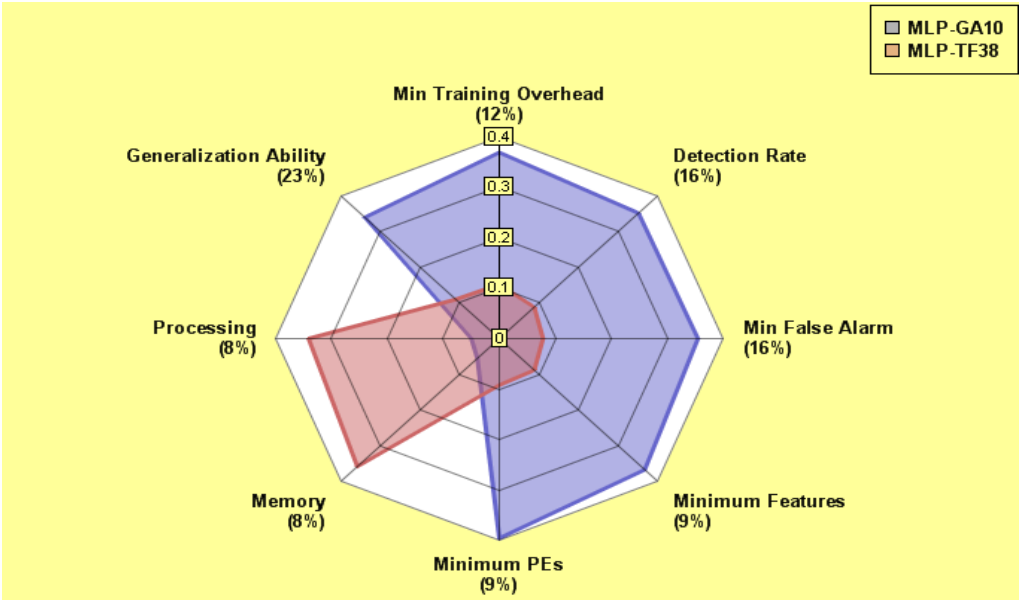


Figure 5.7 MLPGA10 vs. MLP-org-38 (criteria)

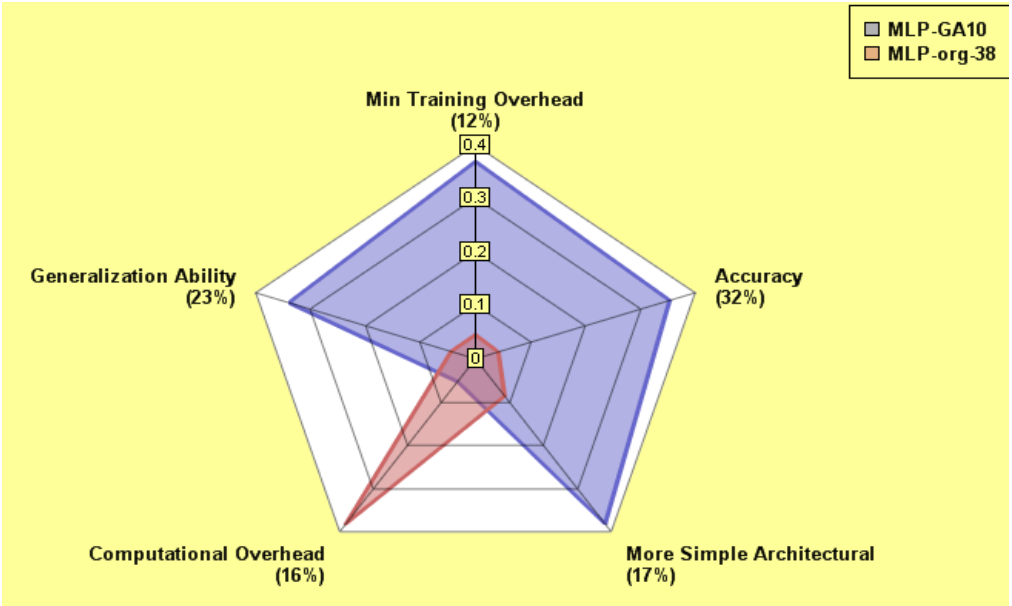


Figure 5.8 MLPGA10 vs. MLP-TF38 (sub-criteria)

Figure 5.8 demonstrate that the MLP with GA10 feature set outperforms the MLP with original 38 (the raw feature set) feature set based on main criteria. The MLP with raw feature set suffers computational and training overheads.

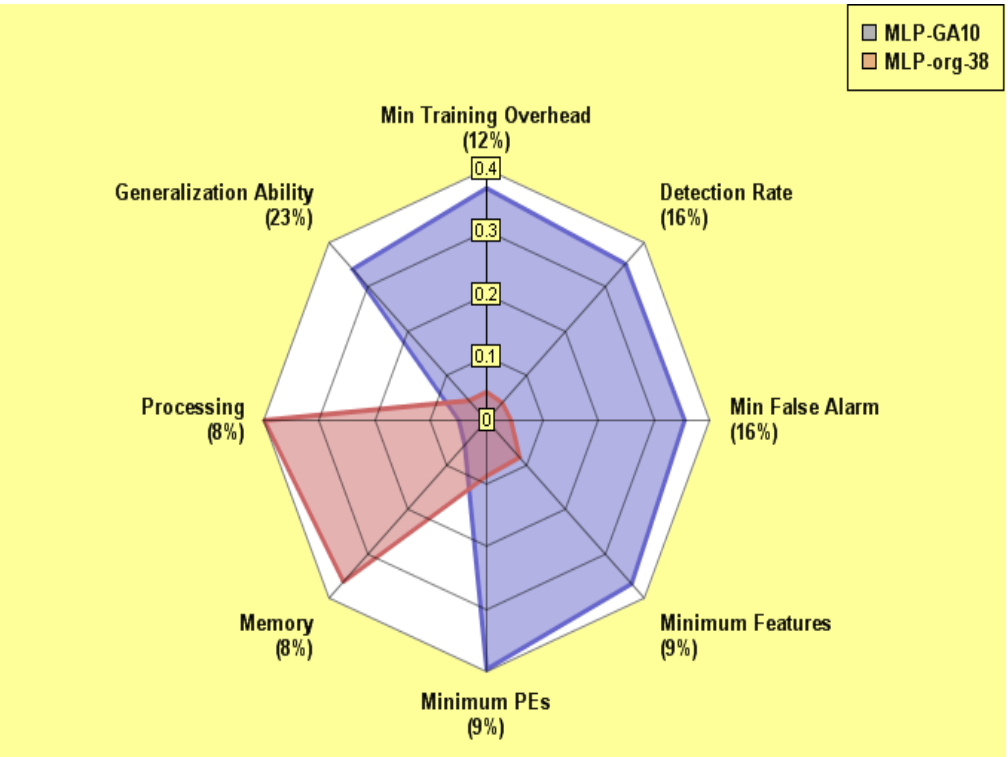


Figure 5.9 MLPGA10 vs. MLP-org-38 (sub-criteria)

Figure 5.9 demonstrate that the MLP with GA10 feature set outperforms the MLP with the raw feature set based on sub criteria. The MLP with raw feature set suffers processing, memory and training overheads that decrease on the whole performance of intrusion analysis engine.

Figure 5.10 demonstrates the MLPs results comparison with GA10, GA12, PCA22, TF38 and org38 (original raw feature) feature set. The MLP with GA10 and GA12 feature set present optimal results as compared to other approaches; PCA22, TF38 and org38 based on main criteria and sub criteria. The main criteria consist of accuracy, minimum training overhead, generalization ability, computational overhead and in architectural simplification. The sub criteria consist of detection rate, minimum false alarm, minimum features, minimum processing elements (PEs), usage of memory, processing time, generalization ability and minimum training overhead. The selection of feature set by searching the PCA space using GA technique results more

sensitive feature set that directly impact on performance of the intrusion detection classifier such as MLP.

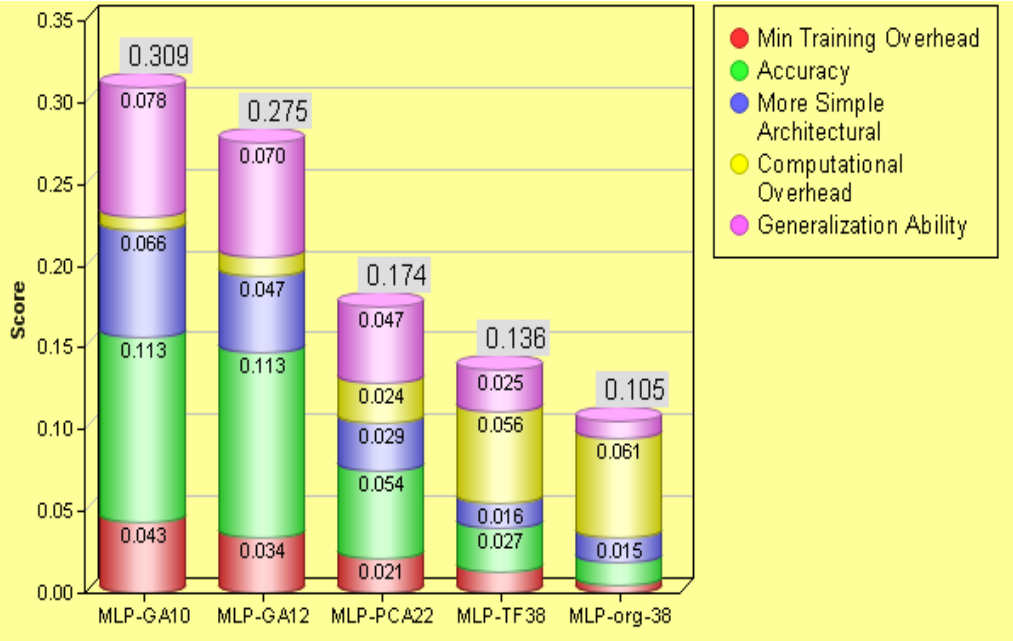


Figure 5.10 MLP Overall performance for intrusion detection

5.2.4.2 Comparative Analysis of applied approach based on SVM with other SVM approaches

The comparison of SVM approach based on pre-defined criteria as shown in Figure 5.1 with other approaches based on the achieved results as mentioned in Table 5.53. The detail comparative analysis of both approaches is depicted in graphs in Figure 5.11 – 5.18.

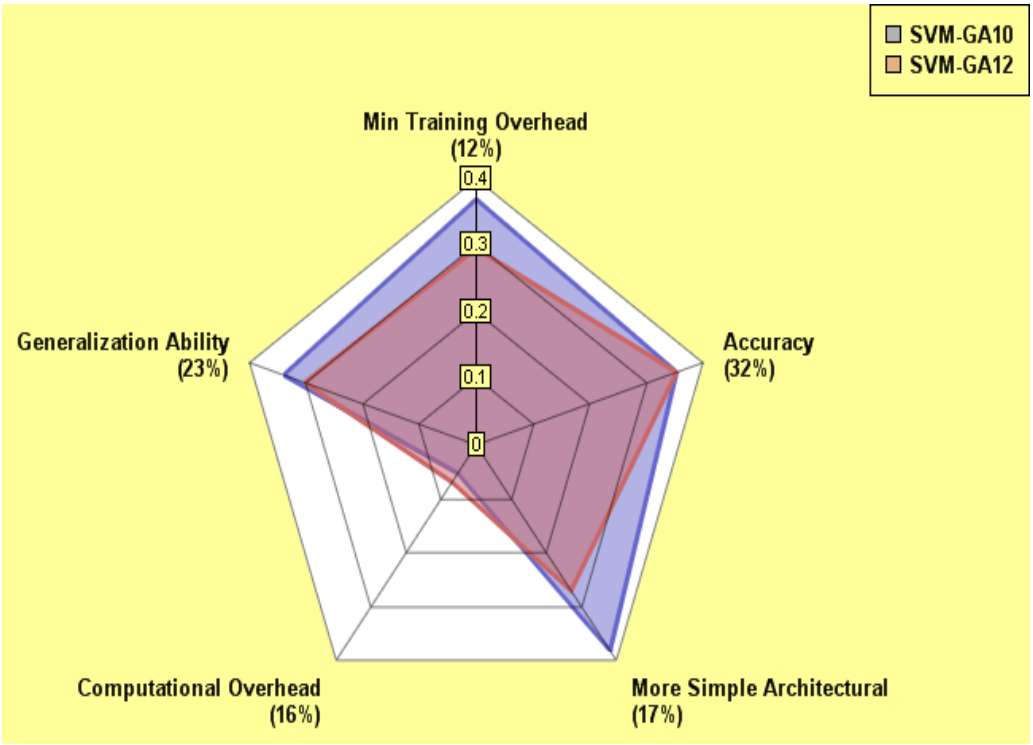


Figure 5.11 SVMGA10 vs. SVMGA12 (criteria)

Figure 5.11 represents the SVM results comparison between two datasets: GA 10 feature set and GA 12 feature set based on main criteria. The SVM shows enhanced performance with GA10 feature set as compared to GA12 feature set based on main criteria: accuracy, minimum training overhead, generalization ability, computational overhead and in architectural simplification.

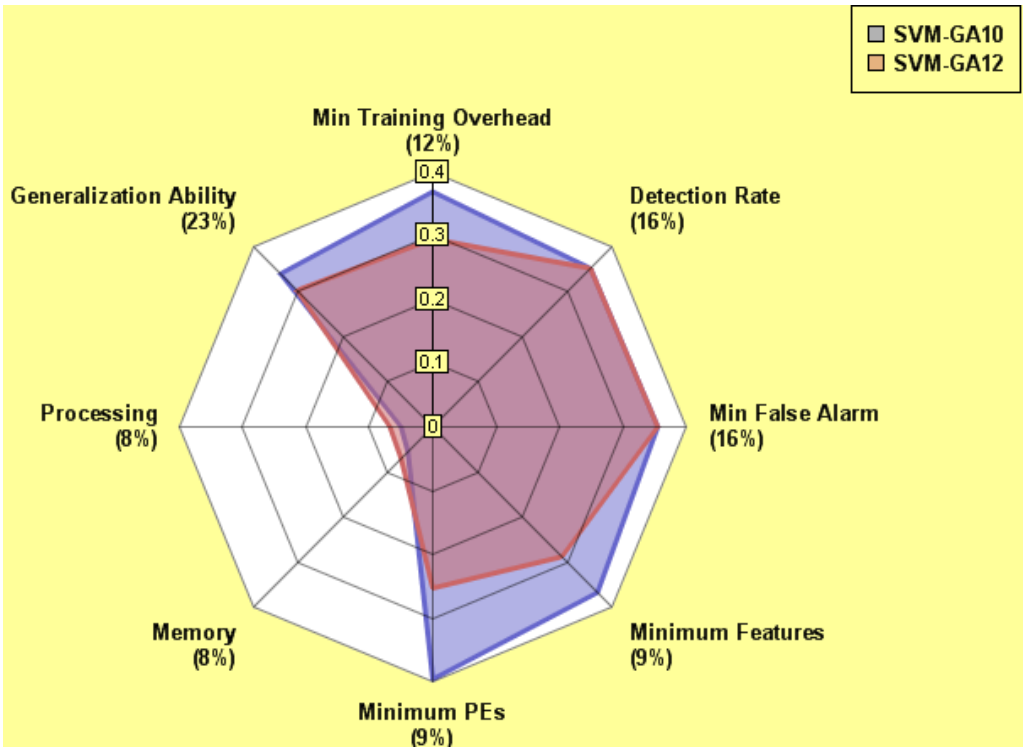


Figure 5.12 SVMGA10 vs. SVMGA12 (sub-criteria)

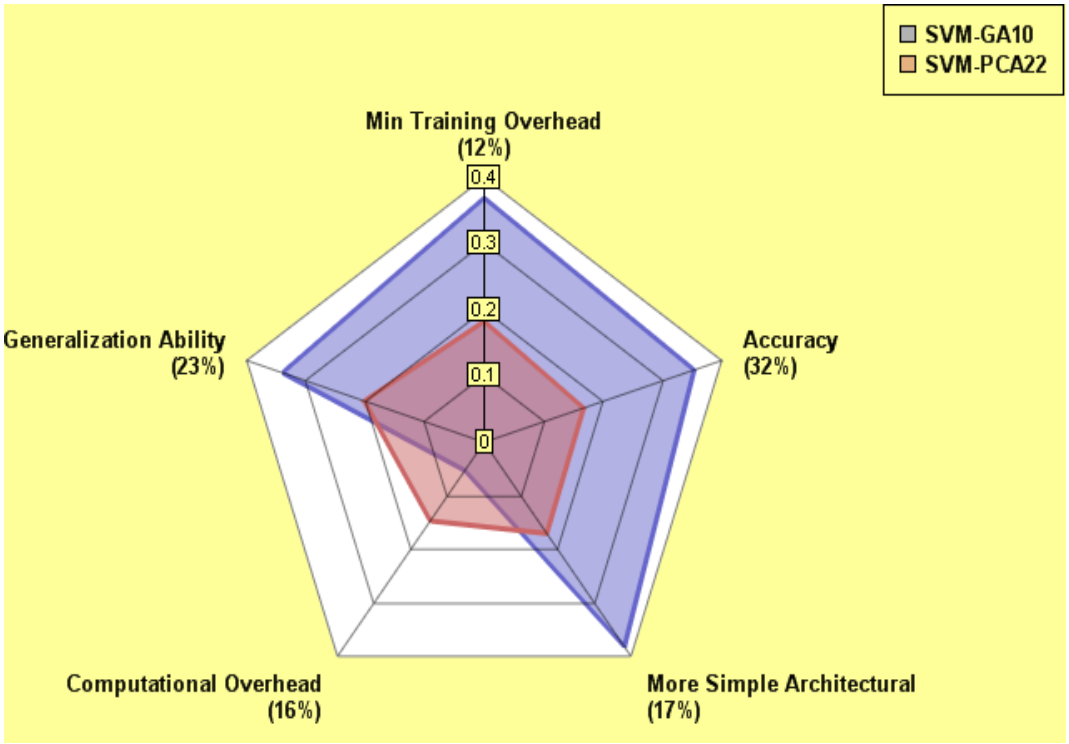


Figure 5.13 SVMGA10 vs. SVMPCA22 (criteria)

Figure 5.12 shows SVM comparative results between two datasets: GA 10 feature set and GA 12 feature set based on sub criteria: detection rate, minimum false alarm, minimum features, minimum processing elements (PEs), memory usage, processing time, generalization ability and minimum training overhead. The utilization of the GA12 feature set slightly increases training and computational overhead which is a contrary to the comparative result with GA10 feature set. Thus, the SVM with GA10 feature set demonstrates enhanced performance as compared to SVM with GA12 feature set based on aforementioned sub criteria.

Figure 5.13 shows SVM results comparison between two datasets: GA10 feature set and PCA22 feature set based on main criteria. The SVM shows better performance with GA10 feature set as compared to PCA22 feature set.

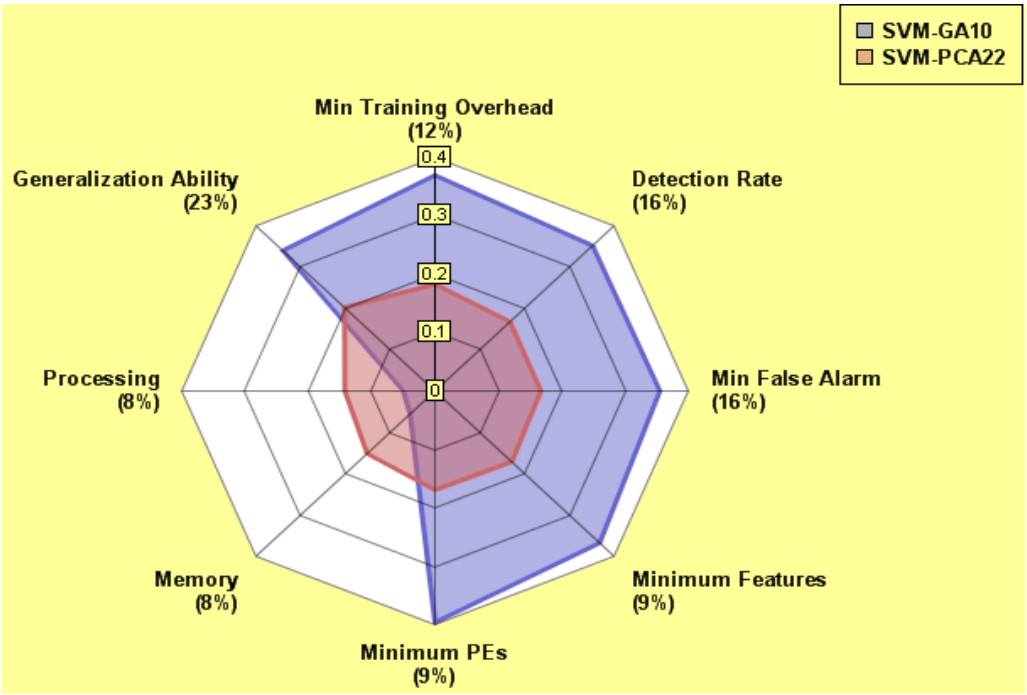


Figure 5. 14 SVMGA10 vs. SVMPCA22 (sub-criteria)

Figure 5.14 shows SVM comparative results between two datasets: GA 10 feature set and PCA22 feature set based on sub criteria: detection rate, minimum false alarm, minimum features, minimum processing elements (PEs), memory usage, processing time, generalization ability and minimum training overhead.

Hence, the use of PCA22 feature set improvises the training and computational overhead as compared to GA10 feature set. Thus, the SVM with GA10 feature set demonstrates better performance than SVM with PCA22 feature set based on aforementioned sub criteria.

Figure 5.15 shows SVM comparative results between the two datasets: GA 10 feature set and TF38 (Transformed features from raw dataset using PCA) feature set based on main criteria. The computational overhead amplifies, as used TF38 instead of GA10 feature set. Hence, the SVM shows better performance with GA10 feature set than the TF38 feature set based on main criteria: accuracy, minimum training overhead, generalization ability, and in architectural simplification.

Figure 5.16 illustrates comparative analysis between SVMs with GA10 and TF38 feature set which bases on the supra-mentioned sub criteria. The SVM with GA10 outperforms the SVM with TF38 feature set.

Figure 5.17 demonstrates that the SVM with GA10 feature set outdoes the SVM with the raw feature set based on sub criteria. The SVM with raw feature set suffers processing, memory and training overheads traits, which results in the deterioration of the whole performance of intrusion analysis engine.

Figure 5.18 demonstrates that the SVM with GA10 feature set supersedes than the SVM with original 38 (the raw feature set) feature set based on main criteria. The SVM with raw feature set suffers computational and training overheads.

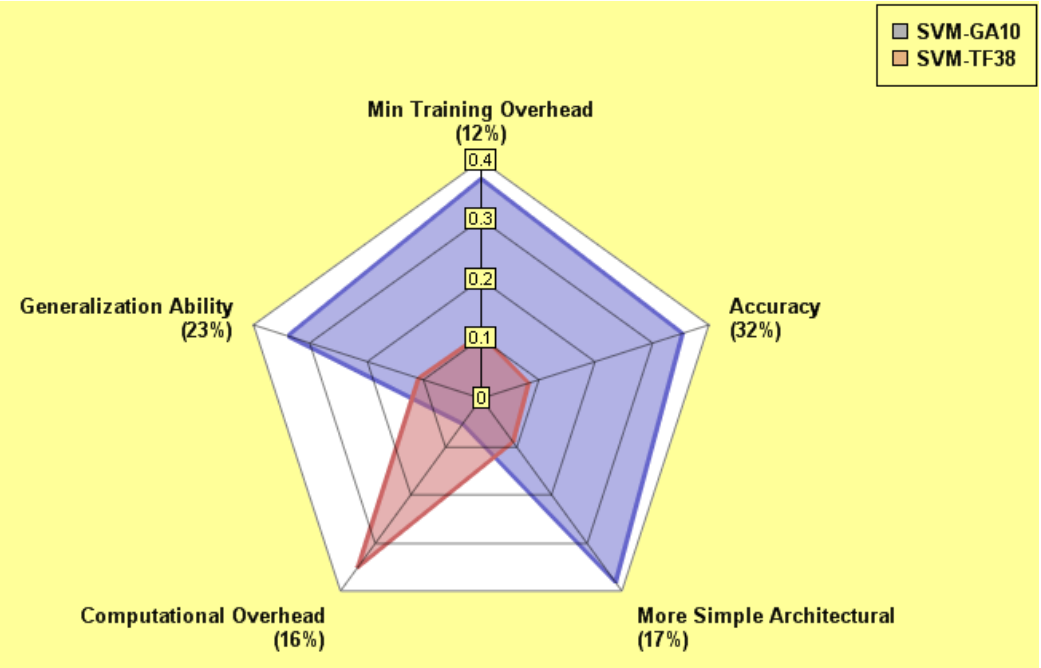


Figure 5.15 SVMGA10 vs. SVMTF38 (criteria)

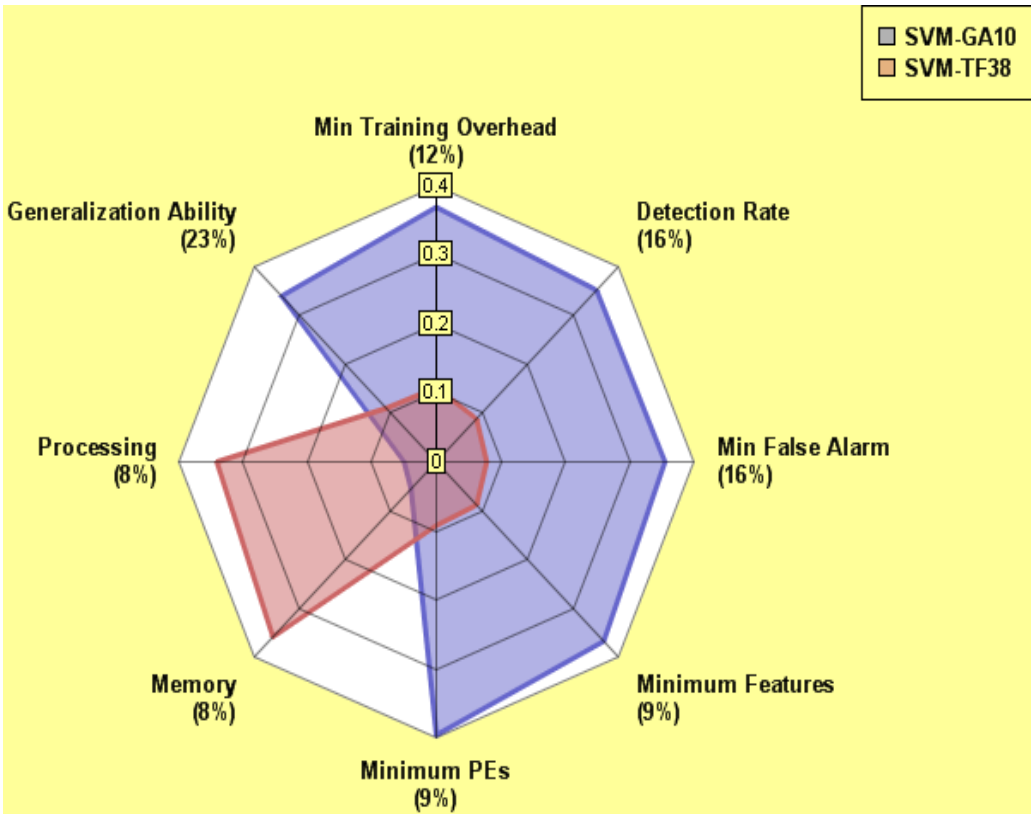


Figure 5.16 SVMGA10 vs. SVM-TF38 (sub-criteria)

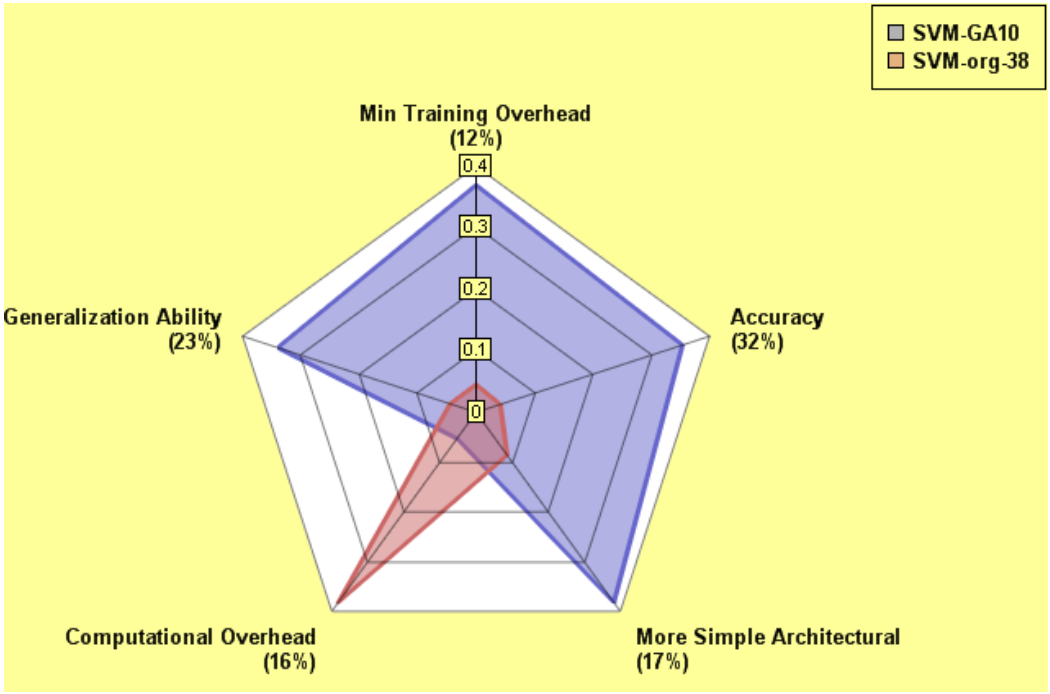


Figure 5.17 SVMGA10 vs. ML-org-38 (criteria)

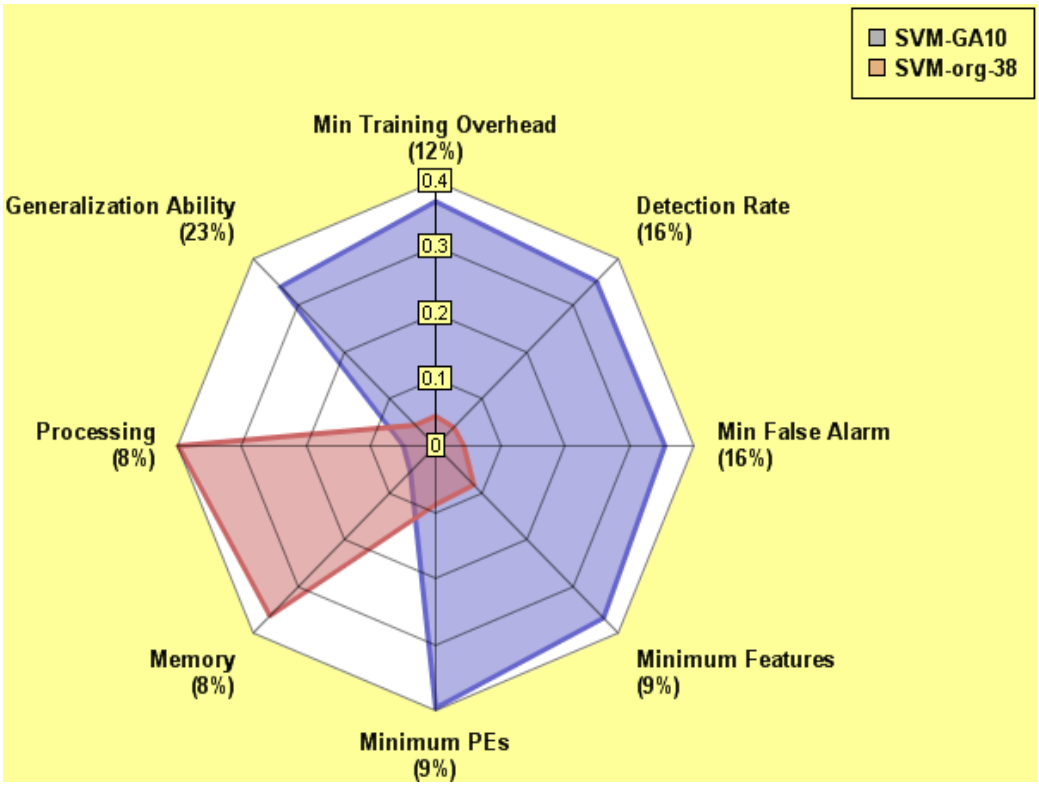


Figure 5.18 SVMGA10 vs. SVM-org-38 (sub-criteria)

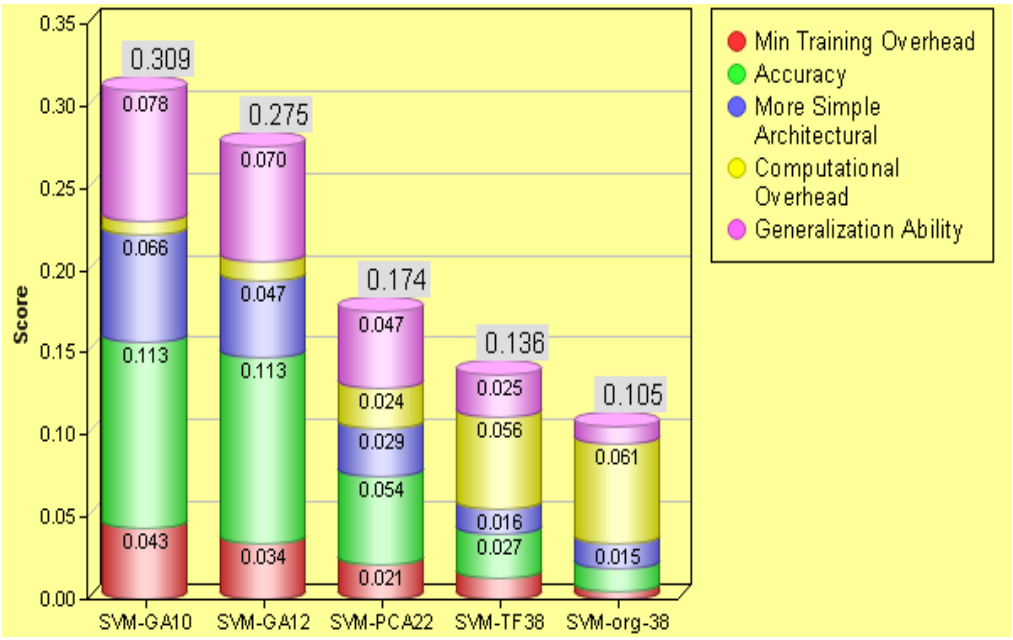


Figure 5.19 SVM Overall performance for intrusion detection

Figure 5.19 demonstrates the SVMs comparative results with GA10, GA12, PCA22, TF38 and org38 (original raw feature) feature sets. The SVM with GA10 and GA12 feature set present optimal results as compared to other approaches; PCA22, TF38 and org38 based on main criteria and sub criteria. The main criteria consist of accuracy, minimum training overhead, generalization ability, computational overhead and architectural simplification. The sub criteria consist of detection rate, minimum false alarm, minimum features, minimum processing elements (PEs), memory usage, processing time, generalization ability and minimum training overhead. The selection of the feature set by searching the PCA space using GA technique offers more sensitive feature set that directly has an impact on the overall performance of the intrusion detection classifier such as SVM.

5.3 Research Contributions

This section presents research contributions in intrusion detection using soft computing techniques; MLP, SVM, GA with PCA. The main objective was to induce an intrusion detection mechanism that produces optimal detection rate and minimize

features that makes architecture simple and reduces training and computational overheads. A List of the contributions mentioned below:

- *Performance optimization in case of detection rate and false alarm:* The applied approach based on PCA and then application of GA for optimal features subset selection positively affects the accuracy of the proposed model based on MLP and SVM.
- *Minimize training overhead:* The adopted mechanism demonstrates less training time as compared to other approaches. Minimum number of features and features with higher discriminatory power reduces the training overheads.
- *Simplified Architectural framework:* The application of PCA and GA for features transformation and selection, made the intrusion analysis engine simple and more efficient.
- *Minimize computational overhead:* This approach considerably reduces memory and computational overheads during training and testing process. The smaller the number of features, the reduced is the memory requirement as well as processing overheads.
- *Contribution in the existing approaches:* The applied approach performed accurately in detection rates, simplification in architecture, and reduced memory and processing requirements.
- *Aides and guides in network security.* The applied mechanism provides help and guides security implementers and researchers in the field of intrusion detection analysis by using the concepts introduced and applied in my work.

5.4 Summary

This chapter describes the following: (1) the results obtained through MLP and SVM in different scenarios during testing and verification phases. (2) The overall performance of both the intrusion analysis engines. Further, the comparison of the performance of MLP along with GA's ten features to other scenarios of MLP and similarly with SVM based on, criteria and sub-criteria as mentioned in this chapter.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Introduction

This chapter emphasizes the conclusion of my work in the vertical of intrusion detection. This is followed by the explanation of the achievements accomplished during this research work. Lastly, the chapter discusses limitations of the work and recommendations/suggestions for the future work.

6.2 Conclusion

Presently, Intrusions on computer and network systems are main security issues. Therefore, it is very important to adhere to such tribulations. The prevention of such intrusions is entirely dependent on their detection which is a key part of many security tools such as: Intrusion Detection System (IDS), Intrusion Prevention System (IPS), Adaptive Security Alliance (ASA), checkpoints and firewalls. Further, accurate detection is another important issue in these days. A number of intrusion detection approaches are available but the main problem is their performance and efficiency, which has been enhanced by increasing the detection rates (99.96% in case of SVM and 99.98% in case of MLP) and reducing false alarms (0.04% in case of SVM and 0.02% in case of MLP).

There are some other drawbacks in the existing intrusion detection approaches such as: usage of raw dataset, bad features extraction, bad features selection, complex classifier architecture, training overhead, and memory and computational overheads. To overcome these issues, this research work presented an optimized intrusion

detection mechanism using techniques; PCA, Genetic Algorithm (GA), Multilayer Perceptron (MLP), and Support Vector machine (SVM). One of the main contributions is the application of GA for optimal feature subset selection that positively affects the whole performance of the applied intrusion detection mechanism. Firstly, a standard dataset; KDD cup is selected and then discard three symbolic features from it. After this, the dataset is further processed in a system acceptable format. For this, the pre-processing process is divided into two steps; feature organization and feature selection. In the first step, the features are organized and arranged to increase their visual discrimination using Principal Component Analysis. Secondly, the GA is applied in order to select a subset of principal components from the principal components space, which offers a subset of features with optimal sensitivity and highest discriminatory power. Then, the selected principal components or PCA features are fed to the proposed model for intrusion analysis. Two classifiers; MLP and SVM are used for intrusion analysis. Firstly, MLP classifier is trained and tested in different scenarios and the results arising out of it are evaluated and compared. Secondly, SVM classifier is trained and tested the same way. Results proved that this research mechanism provides an optimal intrusion detection mechanism that outperforms the existing approaches and has the capability to minimize the number of features and maximize the detection rates.

6.3 Achievements

The main achievements of this research work are as follows:

- *Accuracy improvement:* The applied approach based on PCA and GA for features subset selection improves substantially the accuracy of the proposed intrusion model based on MLP and SVM.
- *Reduces training overhead:* This intrusion detection mechanism reduced training overheads as compared to other approaches. Smaller number of features with higher discriminatory power decreases the training overhead.
- *Architectural framework Improvement:* The application of PCA and GA for features transformation and selection made the intrusion analysis engine more simple and efficient.

- *Reduces computational overhead:* This approach considerably reduces memory and computational overheads during training and testing process. The smaller the number of features decreases memory requirement as well as processing overhead.
- *Contribution in existing approaches:* The applied approach performed better in accuracy, and provided a simple architecture that reduced memory and processing requirements.
- *Help and guide in network security:* The applied mechanism provides help and guidance for security implementers and researchers in the field of intrusion detection by using the concepts introduced and applied in this work.

6.4 Limitations and Future Work

This research work identifies some of the limitations of the applied approach as follows:

- The proposed system works on two classes; normal and intrusive. This generates alarm about the network activity, which is analysed as to whether it is normal or an attack. Therefore, the research can be further extended in the future to classify network activity based on the categories.
- Principal component analysis is used for features organization and arrangement in this research. There are some other alternative techniques; K-dimensional scaling, K-means clustering, self-organizing map, and Kernel PCA. This can further be explored and applied in the future work.
- The features selection is performed using GA in this work. This selection process can be further investigated and deployed using some other techniques; greedy search, back elimination, and Memetic Algorithm (MA) etc.
- This system used two classifiers as intrusive analysis engines; MLP and SVM. This model can further be testified using some other techniques as modular neural network, Jordan/Elman network, recurrent network and fuzzy techniques.

REFERENCES

Ahmad I, Abdullah AB, and Alghamdi AS. 2009a. Application of Artificial Neural Network in Detection of DOS Attacks. ACM International Conference on Security of Information and Networks (SIN). Gazimagusa, North Cyprus, Turkey. p 229-234.

Ahmad I, Abdullah AB, and Alghamdi AS. 2009b. Application of Artificial Neural Network in Detection of Probing Attacks. IEEE Symposium on Industrial Electronics and Applications (ISIEA). Kuala Lumpur, Malaysia. p 557 - 562

Ahmad I, Abdullah AB, and Alghamdi AS. 2009c. Artificial Neural Network Approaches to Intrusion Detection: A Review. Telecommunications and Informatics conference. Istanbul, Turkey. p 200-205.

Ahmad I, Abdullah AB, and Alghamdi AS. 2010. Towards the selection of best neural network system for intrusion detection. International Journal of the Physical Sciences 5(12):1830-1839.

Ahmad I, Ansari MA, and Mohsin S. 2008. Performance Comparison between Backpropagation Algorithms Applied to Intrusion Detection in Computer Network Systems. Recent Advances in Systems, Communications & Computers. p 47-52.

Ahmad I, Swati SU, and Mohsin S. 2007. Intrusion Detection Mechanism by Resilient Back Propagation (RPROP). European Journal of Scientific Research (EJSR). 17(4):523-530.

Amini M, and Jalili R. 2004. Network Based Intrusion Detection using Unsupervised Adaptive Resonance Theory. Fourth International ICSC Symposium on Engineering of Intelligent Systems (EIS), Portugal. p 250-258.

Amini M, Jalili R, and Shahriari HR. 2006. RT-UNNID: A practical solution to real-time network-based intrusion detection using unsupervised neural networks. Computers & Security Elsevier Inc 25(6):459-468.

Anderson JP. 1980. Computer Security Threat Monitoring and Surveillance. Technical Report. J.P. Anderson Company, Fort Washington, Pennsylvania. p 55-66.

Bace R, and Mell P. 2001. Intrusion Detection Systems. National Institute of Standards and Technology (*NIST*) Special Publication. p1-51. [Online].[Avaialable]: http://csrc.nist.gov/.

Bäck T. 1996. Evolutionary Algorithms in Theory and Practice: Genetic Algorithms, Evolution Strategies, Evolutionary Programming. Oxford University Press. New York. p 55-65.

Bankovic Z, Moya JM, Araujo Á, Bojanic S, and Nieto-Taladriz O. 2009. A Genetic Algorithm-based Solution for Intrusion Detection. Journal of Information Assurance and Security 4:192-199.

Bankovic Z, Stepanovic D, Bojanic S, and Nieto-Taladriz O. 2007. Improving Network Security Using Genetic Algorithm Approach, Computers & Electrical Engineering. Security of Computers & Networks 33(5-6):438-451.

Bebis G, Louis S, Varol Y, and Yfantis A. 2002. Genetic Object Recognition Using Combinations of Views. IEEE Trans Evol Comput 6(2):132–146. .

Bebis G, Uthiram S, and Georgiopoulos M. 2000. Face Detection and Verification Using Genetic Search. Int. J. Artif. Intell. Tools. 4(1):225–246.

Bivens A, Palagiri C, Smith R, Szymanski B, and Emrechts M. 2002. Network-Based Intrusion Detection Using Neural Networks. 12th Proc. Intelligent Engineering Systems through Artificial Neural Networks (ANNIE), ASME press. New York, NY. p 579-584.

Browne A. 2000. Neural Network Analysis, Architectures and Applications, Institute of Physics Publishing (IOP) Press, London. p 98-101.

Burges C. 1998. Tutorial On Support Vector Machines for Pattern Recognition. Data Mining Knowledge Discovery. 2(2): 955–974.

Cannady J. 1998 Artificial Neural Networks for Misuse Detection.National Information Systems Security Conference (NISSC'98). Arlington, VA. p443-455.

Cortes C, and Vapnik V. 1995. Support Vector Networks, Journal of Machine Learning. Springer Netherlands 20(3):273-297.

Cristianini N, Campbell C, Burges C, and 2002. Kernel methods: Current Research and Future Directions, Journal of Machine Learning, Springer Netherlands. 46(2): 5-9.

Cristianini N, and Shawe J. 2000. An Introduction to Support Vector Machines and Other Kernel-based Learning Methods: Cambridge University Press. p55-60. [Online].[Available]:http://www.support-vector.net/.

Denault M, Gritzalis D, Karagiannis D, and Spirakis. 1994. Intrusion Detection: Approach and Performance Issues of the SECURENET System. Computers and Security. p 495-507.

Denning D.1987. An Intrusion-Detection Model. IEEE Transactions on Software Engineering. 13(2):222-232.

Dutta M, Chatterjee A, and Rakshit A. 2006. Intelligent Phase Correction in Automatic Digital AC Bridges by Resilient Backpropagation Neural Network. Measurement 39(10): 884–891.

Eiben AE, and Smith JE. 2003. Introduction to Evolutionary Computing. Natural Computing Series. Springer-Verlag Berlin Heidelberg. p 222-301.

Este A, Gringoli F, and Salgarelli L. 2009. Support Vector Machines for TCP traffic classification, Computer Networks. p 2476-2490.

Fausett LV. 2009. Fundamentals of neural networks: architectures, algorithms, and applications. Prentice-Hall, Prentice-Hall, Inc. Upper Saddle River, NJ, USA. p 33-44.

Folino G, Pizzuti C, and Spezzano G. 2005. GP ensemble for distributed intrusion detection systems. Pattern Recognition and Data Mining. Lecture Notes in Computer Science (LNCS), Springer Berlin, 3686 (1). p 54-62.

Fox KL, Henning RR, Reed JH, Simonian RP. 1990. In Proc. 13th National Computer Security Conference. Information Systems Security. p 124-134.

Fu L. 1992. A Neural Network Model for Learning Rule-Based Systems. In Proceedings of the International Joint Conference on Neural Networks. p 343-348.

Ghosh A, and Schwartzbard A. 1999. A Study in Using Neural Networks for Anomaly and Misuse Detection. USENIX Security Symp. Washington, D.C, USA. p 213-217.

Ghosh AK, Schwartzbard A, and Schatz M. 1999. Learning Program Behavior Profiles for Intrusion Detection. SANS Workshop on Intrusion Detection and Network Monitoring. p 51-62.

Gong S. 2001. Dynamic Vision: From Images to Face Recognition: Imperial College Press.London,UK. p 43-50.

Hammerstrom D. 1993. Neural Networks At Work. IEEE Spectrum. p 26-53.

Lewis JP. 2004. Tutorial on SVM. CGIT Lab, USC. p 14-20. [Online].[Available]: http://www.dataminingtools.net/.

James C. 2000a. Artificial Neural Networks for Misuse Detection. National Information Systems Security Conference. p 368–381.

James C. 2000b. Artificial Neural Networks for Anamoly Detection. National Information Systems Security Conference. p 281–288.

James H. 1997. Adaptation in Natural and Artificial System. MIT press Cambridge,MA,USA. p 55-62.

Jang JR, Sun CT, and Mizutani E. 1997. Neuro-Fuzzy and Soft Computing. Prentice Hall. p 212-220.

Jing-Xin W, Zhi-ying W, and Kui D. 2004. A Network Intrusion Detection System Based on the Artificial Neural Networks. ACM Transaction on information security 85:166-170.

Jirapummin C, Wattanapongsakorn N. 2002. Hybrid Neural Networks for Intrusion Detection System. The International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC). Phuket, Thailand. p 16-19.

Jonsson E, Almgren M, and Alfonso. 2004. Recent Advances in Intrusion Detection. 7th International Symposium, RAID, Sophia Antipolis. p 102-108.

Khan L, Awad M, and Thuraisingham B. 2007. A New Intrusion Detection System Using Support Vector Machines And Hierarchical Clustering. The International Journal on Very Large Data Bases 16(4):507–521.

Kim DS, Nguyen HN, Ohn SY, and Park JS. 2005. Fusions of GA and SVM for Anomaly Detection in Intrusion Detection System. Advances in Neural Networks Lecture Notes in Computer Science 3498(3):415-420.

Kuchimanchi GK, Phoha VV, Balagami KS, and Gaddam SR. 2004. Dimension reduction using feature extraction methods for Real-time misuse detection systems. IEEE Workshop on Information Assurance and Security. p 195–202.

Labib K, and Vemuri VR. 2004. Detecting and Visualizing Denial-of-Service and Network Probe Attacks Using Principal Component Analysis. Third Conference on Security and Network Architectures. La Londe, France. p 45-55.

Lakhina S, Joseph S, and Verma B. 2010. Feature Reduction using Principal Component Analysis for Effective Anomaly–Based Intrusion Detection on NSL-KDD. International Journal of Engineering Science and Technology 2(6):1790-1799.

Lee SC, and Heinbuch DV. 2001. Training a Neural-Network Based Intrusion Detector to Recognize Novel Attacks. IEEE Trans on Systems, Man, and Cybernetics. p 294-299.

Li RZ, and Guan X. 2007. Accurate Classification of the Internet Traffic Based on the Svm Method. 42th IEEE International Conference on Communications (ICC 2007).p455-462.

Lilia de sa Silva, Santos AD, Silva JD, and Montes A. 2004. A Neural Network Application for Attack Detection in Computer Networks. IEEE Transactions. p 1569-1574.

Lippmann RP, and Cunningham RK. 2000. Improving Intrusion Detection Performance Using Keyword Selection and Neural Networks. Computer Networks 34(8):597-603.

Liu G, and Yi Z. 2006. Intrusion Detection Using PCASOM Neural Networks. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer. p 240–245.

Liu G, Yi Z, and Yang S. 2007. A Hierarchical Intrusion Detection Model Based on the PCA Neural Networks, Neurocomputing. 14th European Symposium on Artificial Neural Networks. p 1561-1568.

Lunt TF. 1989. Real-Time Intrusion Detection. Proceedings from IEEE COMPCON. p 455-461.

Michalewicz Z. 1996. Genetic Algorithms, Evolution Programs. Springer Verlag. Berlin. p 231-236.

Min L, and Wang D. 2009. Anomaly Intrusion Detection Based on SOM. WASE International Conference on Information Engineering. p 40-43.

Miranda AA, Borgne YB, and Bontempi G. 2008. New Routes from Minimal Approximation Error to Principal Components , Springer Verlag. Berlin. p 344-351.

Mitchell T, and Learning M. 1997. Computer Science Series. McGraw-Hill. p 201-212.

Mukherjee B, Heberlein LT, and Levitt KN.1994. Network Intrusion Detection. IEEE Network. p 26-41.

Smith N, and Gales M. 2002. Advances in Neural Information Processing Systems. Cambridge, MA. MIT Press. p 33-38.

Oja E. 1992. Principal Components, Minor Components, and Linear Neural Networks. Neural Networks 5(6):927–935.

Osareh A, and Shadgar B. 2008. Intrusion Detection in Computer Networks based on Machine Learning Algorithms. International Journal of Computer Science and Network Security (IJCSNS), 3 (4):15-23.

Pearson K. 1901. On Lines and Planes of Closest Fit to Systems of Points in Space. Philosophical Magazine. p 559–572.

Pervez S, Ahmad I, Akram A, and Swati SU. 2007. A Comparative Analysis of Artificial Neural Network Technologies in Intrusion Detection Systems. WSEAS Transaction on Computers. p 175-180.

Principe JC, Euliano NR, and Lefebvre WC. 2000. Neural and Adaptive Systems: Fundamentals through Simulations. New Yor, NY, John Wiley. p 213-218.

Rhodes BC, James AM, and James DC. 2000. Multiple Self-Organizing Maps for Intrusion Detection. NIST National Information Systems Security (NISS) Conference. p 344-350.

Ryan J, Lin MJ. 1998. Intrusion Detection with Neural Networks, Advances in Neural Information Processing Systems. 10 (2):943-949.

Saad A. 2008. An Overview of Hybrid Soft Computing Techniques for Classifier Design and Feature Selection, Hybrid Intelligent Systems. Eighth International Conference . p 579-583.

Sandhya S. 2009. Neural Networks for Applied Sciences and Engineering: From Fundamentals to Complex Pattern. p71-78.

Shyu M, Chen S, Sarinnapakorn K, and Chang L. 2003. A Novel Anomaly Detection Scheme Based on Principal Component Classifier. ICDM'03. p 172–179.

Smith LI. 2002. A Tutorial on Principal Components Analysis. Cornell University, USA. p12-16. [Online]. [Available]:http://www.cs.otago.ac.nz/.

Sun Z, Bebis G, and Miller R. 2004. Object Detection Using Feature Subset Selection, Pattern Recognition. 37(11):2165-2176.

Verikas A, Kalsyte Z, Bacauskiene M, and Gelzinis A. 2010. Hybrid and Ensemble-Based Soft Computing Techniques In Bankruptcy Prediction: A Survey, A Fusion of Foundations, Methodologies and Applications. p 995-1010.

Yatim A, and Utomo W. 2006. Efficiency Optimization of Variable Speed Induction Motor Drive Using Online Backpropagation. IEEE International Conference on Power and Energy. p 441-446.

Yu J, Lee H, Kim MS, and Park D. 2008. Traffic Flooding Attack Detection with SNMP MIB using SVM, Computer Communications. 31(17):4212-4219.

Yu Y, Ge Y, and Fu-xiang G. 2005. A Neural Network Approach for Misuse and Anomaly Intrusion Detection. Journal of Natural Sciences. Wuhan University Journals Press. 10 (1) :115-118.

Zargar GR, and Kabiri P. 2010. Selection of Effective Network Parameters in Attacks for Intrusion Detection, Advances in Data Mining. Applications and Theoretical Aspects. Lecture Notes in Computer Science (LNCS), 6171(2):643-652.

Zhang Z, Li J, Jorgenson J, and Hide UJ. 2001. A Hierarchical Network Intrusion Detection System Using Statistical Preprocessing and Neural Network Classification. IEEE Workshop on Information Assurance and Security. West Point. p 85-90.

APPENDIX A

PUBLICATIONS

1 Ahmad I, Abdullah AB, and Alghamdi AS. 2009a. Application of Artificial Neural Network in Detection of DOS Attacks. ACM International Conference on Security of Information and Networks (SIN). Gazimagusa, North Cyprus, Turkey. p 229-234.

2 Ahmad I, Abdullah AB, and Alghamdi AS. 2009b. Application of Artificial Neural Network in Detection of Probing Attacks. IEEE Symposium on Industrial Electronics and Applications (ISIEA).Kuala Lumpur, Malaysia. p 557 - 562

3 Ahmad I, Abdullah AB, and Alghamdi AS. 2009c. Artificial Neural Network Approaches to Intrusion Detection: A Review. Telecommunications and Informatics conference. Istanbul, Turkey. p 200-205.

4 Ahmad I, Abdullah AB, and Alghamdi AS. 2010a. Applying neural network to U2R attacks. IEEE Symposium on Industrial Electronics and Applications (ISIEA 2010). Penang, Malaysia. p 1-6.

5 Ahmad I, Abdullah AB, and Alghamdi AS. 2010b. Comparative Analysis of Intrusion Detection Approaches. IEEE UKSIM. Cambridge University (Emmanuel College), England. p 586 - 591

6 Ahmad I, Abdullah AB, and Alghamdi AS. 2010c. Evaluating Intrusion detection Approaches using Analytic Hierarchy process. In: Abstract, editor. IEEE ITSIM. Kuala Lumpur, Malaysia. p 885 - 890.

7 Ahmad I, Abdullah AB, and Alghamdi AS. 2010d. Evaluating Intrusion Detection Approaches Using Multi-criteria Decision Making Technique, Information Sciences and Computer Engineering. International Journal of Information Sciences & Computer Engineering (IJISCE) 1(1):60-67.

8 Ahmad I, Abdullah AB, and Alghamdi AS. 2010e. Investigating Supervised Neural Networks to Intrusion Detection. International Journal of Research And Surveys (IJRS-ICIC-EL) Japan 14(3)2133-2138.

9 Ahmad I, Abdullah AB, and Alghamdi AS. 2010f. Remote to Local Attack (R2L) Detection Using Supervised Neural Network. IEEE International Conference for Internet Technology and Secured Transactions (ICITST) [IN PRESS].

10 Ahmad I, Abdullah AB, and Alghamdi AS. 2010g. Towards the designing of robust IDS through an optimized advancement of neural networks: Lecture Notes in Computer Science, Springer, Berlin.

11 Ahmad I, Abdullah AB, and Alghamdi AS. 2010h. Towards the Selection of Best Neural Network System for Intrusion Detection. International Journal of the Physical Sciences (IJPS)[ISI/SCI] 5(12):1830-1839.

12 Ahmad I, Abdullah AB, and Alghamdi AS. 2011a. Distributed Denial of Service Attacks Detection Using Support Vector Machine. INFORMATION: An International Interdisciplinary Journal (ISI/SCI), [IN PRESS].

13 Ahmad I, Abdullah AB, and Alghamdi AS. 2011b. Optimized Intrusion Detection Mechanism Using Soft Computing Techniques Telecommunication System (ISI/SCI) [IN PRESS].

14 Ahmad I, Abdullah AB, and Alghamdi AS. 2011c. Features Subset Selection for Network Intrusion Detection Mechanism Using Genetic Eigen Vectors, Proceedings of 2011 International Conference on Telecommunication Technology and Applications (ICTTA 2011), Australia, Sydney, May 2-4, 2011 [IN PRESS].

APPENDIX B

DEFINATION OF TERMINOLOGIES

Attack: The act of attempts to bypass one or more computer security control to achieve unauthorized access.

Penetration: The intentional passage through an equipment or computer system by illegal way.

Anomaly Detection: Activities which vary from established patterns for users, or groups of users.

Misuse Detection: Comparison of a user's activities with the known behaviors of attackers.

ANN: A network of highly interconnected processing elements called neurons that transform a set of inputs to a desired output.

Layer: A component of ANN containing neurons.

Synapse: A component of ANN used as connection between layers.

Intruder: An illegal user that can access network/system resources and play some thing havoc.

IDS: A system that detect unauthorized access to a computer or network.

Kddcup: An attack database that is standard for the evaluation of IDS.

DOS: A type of attack on a network that is designed to bring the network by flooding it with useless traffic.

Probing: It involves discovering the algorithms and parameters of the recommender system itself. It may be necessary for an intruder to acquire this knowledge through interaction with the system itself.

R2U: It involves unauthorized access from a remote machine.

U2R: It involves unauthorized access to local super user privileges by a local unprivileged user.

Packet: A basic communication unit.

False Positive: When the system classifies an action as intrusion when it is a legal action.

False Negative: Intrusion occurred but passed a normal by IDS.

Subversion: The processing of changing behavior of IDS to false negative occurs.

Support Vector Machines (**SVMs):** SVMs are a set of related supervised learning methods that analyze data and recognize patterns, used for classification and regression analysis.

Principal Component Analysis (PCA): PCA is a mathematical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of uncorrelated variables called principal components.

Genetic Algorithms (GAs): GAs are adaptive heuristic search algorithm based on the ideas of natural selection and genetic. The basic concept of GAs is designed to simulate processes in natural system essential for evolution, particularly those that follow the principles first laid down by Charles Darwin of survival of the fittest.