



Back Propagation Recursive Least Squares Algorithm for Intrusion Detection Systems

خوارزمية الانتشار المرتد أصغر التربيعات التكرارية لأنظمة كشف التطفل

By

Omar Hisham Rasheed Al-sadoon

Supervisor: Prof. Dr. Reyadh Shaker Naoum

Co Supervisor: Dr. Sadeq Al-Hamouz

**A Thesis Submitted in Partial Fulfilment of the Requirement Master
Degree in Computer Science**

Department of Computer Science

Faculty of Information Technology

Middle East University

Amman- Jordan

(December-2014)

جامعة الشرق الأوسط

تفويض

أنا عمر هشام رشيد السعدون أعطي التفويضات لجامعتي, جامعة الشرق الأوسط لتكون قادرة على تزويد المؤسسات التعليمية الرسمية بنسخ من رسالتي حال طلبها.

الاسم: عمر هشام رشيد السعدون

التاريخ كانون الثاني 2014

التوقيع: 

Middle East University

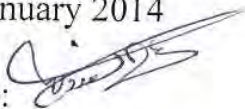
Authorization

I, Omar Hisham Rashid Al-Sadoon , I give the authorization for my University, Middle East University for being able to provide any formal educational institution of copies from my thesis upon request.

Name: Omar Hisham Rashid Al-Sadoon

Date: January 2014

Signature:



Middle East University
Examination Committee Decision

This is to certify that the thesis entitled "Back Propagation Recursive Least Squares Algorithm for Intrusion Detection Systems" was successfully defined and approved on April, 2015 .

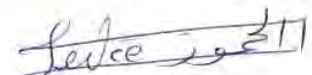
Examination Committee Members

Signature

Dr. Sadeq AlHamouz (supervisor & chairman)

Associate Professor Faculty of information technology

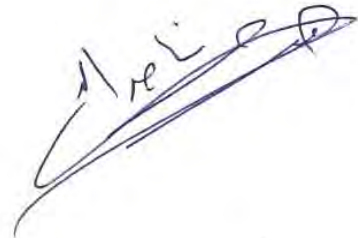
Middle East University (MEU)


6-4-2015

Dr. Hebah H.O. Nasereddin (Member)

Associate Professor Faculty of information technology

Middle East University (MEU)



Dr. Ashraf Bany Mohammad (external Member)

Associate Professor Faculty of business

University of Jordan



Dedication

This dissertation is dedicated to my family and my friends.

A special gratitude feeling to my parents for their encouragement words during my study and for my sisters and brothers who have never left my side. This dissertation is also dedicated to my many friends who gave me the support during the research. My extended appreciation for what they have done for me.

Omar-2014

Acknowledgment

*First and foremost, I would like to express my heartfelt gratitude and appreciation to my supervisor **Prof. Dr. Reyadh Naoum** who without his meticulous supervision, faithful guidance and continuous support, this work could never be accomplished.*

*My Extended thanks are also for **Dr. Sadeq Al Hamour** for his guidance and support during this research. Also, I wish to extend my profound gratitude and thanks to the **Dean and the head of the faculty of information technology in Middle East University**, and all the teaching staffs for every bit they have done to me.*

Genuine thanks to all my friends for their help and support.

Finally, I would like to express my love and gratitude to my family for their support and patience over this adventure. I don't think I would've been able to do this without them.

Omar – 2014

Table of Content

Back Propagation recursive Least Squares Algorithm for Intrusion Detection Systems...	i
التفويض.....	II
Authorization.....	III
Examination committee decision	IV
Dedication.....	V
Acknowledgment.....	VI
Table of Contents.....	VII
Table of Tables.....	XI
Table of figures.....	XIV
Abbreviations.....	XVI
الملخص.....	XVII
Abstract.....	XVIII
Chapter one Introduction.....	2
1.1 Preface.....	2
1.2 Problem Identification.....	4
1.3 Question.....	5
1.4 Study Objectives.....	5
1.5 Contribution.....	6
1.6 Significant.....	7
1.7 Limitations.....	7
1.8 Researches published.....	7
Chapter Two Theoretical Background and Literature Review	9
2.1 Overview.....	9

2.2 Theoretical Background.....	10
2.2.1 K-means Clustering Algorithm.....	10
2.2.2 Novel K-Nearest Neighbour Algorithm (K-NN).....	12
2.2.3 Back Propagation (BP) Algorithm.....	13
2.2.3.1 Back Propagation Recursive Least Squares (BPRLS) Algorithm.....	16
2.2.3.1.1 Normal Equations for the Linear Combiner.....	18
2.2.3.1.2 Inverse Function.....	19
2.3 Data Normalizations.....	23
2.4 Intrusion Detection Systems.....	24
2.4.1 Motivation.....	24
2.4.2 Intrusion Detection Systems Definitions.....	24
2.4.3 Intrusion Detection systems Functions.....	25
2.4.4 Intrusion Detection Systems Criteria.....	25
2.4.4.1 The Methods of Detection.....	25
2.4.4.1.1 Misuse Intrusion Detection Systems.....	26
2.4.4.1.2 Anomaly Intrusion Detection Systems.....	26
2.4.4.2 The Response upon Detection.....	26
2.4.4.3 The Information Exporter.....	27
2.4.5 Intrusion Detection Systems building Steps.....	27
2.4.6 Attacks Classification.....	28
2.4.7 Intrusion Detection Systems Evaluation.....	29
2.5 Artificial Neural Networks.....	32
2.5.1 Overview.....	32
2.5.2 Artificial Neural Networks Definitions.....	32
2.5.3 Artificial Neurons.....	33

2.5.4 Feed Forwards Neural Networks (FFNN).....	34
2.5.5 Learning Process and Supervised Learning.....	35
2.6 Literature Review.....	37
Chapter Three Proposed Intrusion Detection System Algorithms Implementation.....	43
3.1 Preface.....	44
3.2 Data Pre-Processing Phase.....	45
3.2.1 Training and Testing Data Set Collection Stage.....	45
3.2.2 Data Codification Stage.....	47
3.2.3 Data Normalization Stage.....	55
3.3 Intrusion Detection Systems Implementation phase.....	56
3.3.1 K-means Intrusion Detection System.....	56
3.3.1.1 K-means IDS Pseudo Code and Mat lab Implementation.....	57
3.3.2 Novel K-NN Intrusion Detection System.....	61
3.3.2.1 Novel K-NN IDS Pseudo Code and Mat lab Implementation.....	61
3.3.3 Multilayer Back Propagation Recursive Least Squares Based IDS.....	65
3.3.4 Proposed BPRLS IDS Training phase and Mat lab Implementation.....	67
3.3.5 Proposed BPRLS IDS testing Phase and Mat lab Implementation.....	71
Chapter Four Experimental Results and Systems Performance Analysis.....	74
4.1 Implementation Specification.....	75
4.2 Evaluation Criteria.....	75
4.3 Experiential Results of the Proposed K-means IDS Algorithm.....	76
4.3.1 Experiential Results of the Proposed K-means IDS Algorithm: Training Phase...77	
4.3.2 Experiential Results of the Proposed K-means IDS Algorithm: Testing Phase.....	79
4.4 Experiential Results of the Proposed Novel K-NN IDS Algorithm.....	82
4.5 Experiential Results of the Proposed BPRLS IDS.....	85

4.5.1 Experiential Results of the First Experiment of BPRLS IDS Algorithm Training Phase.....	85
4.5.2 Experiential Results of the First Experiment of BPRLS IDS Algorithm Testing Phase.....	88
4.5.3 Experiential Results of the Second Experiment of BPRLS IDS Algorithm Training Phase.....	92
4.5.4 Experiential Results of the Second Experiment of BPRLS IDS Algorithm Testing Phase.....	93
4.6 Experimental Results Comparative Analysis between Our Proposed Intrusion Detection Systems.....	96
4.7 Experimental Results Comparative between the Proposed System and other Systems.....	97
Chapter Five Conclusion, Recommendation and Future Work.....	99
5.1 Introduction.....	100
5.2 Conclusion and Results Discussion.....	100
5.3 Recommendation and Future Work.....	101
References.....	103

Table of Tables

Table 2-1 Type of Sub Attacks Occur Within Network.....	29
Table 2-2 Confusion Matrix.....	30
Table 3-1 the Numerical Representation of 2 th Feature (Protocol).....	48
Table 3-2 the Numerical Representation of 3 th Feature (Service).....	49
Table 3-3 the Numerical Representation of 4 th Feature (Flag).....	51
Table 3.4 the Numerical Representation of 42 th Feature (Records Label).....	51
Table 3-5 Full Description of Feature of a Connection Record of NSL-KDD Data Set	54
Table 4-1 the Number of Record Classes In 8% Training NSL-KDD Data Set.....	77
Table 4-2 Confusion Matrix of Training Stage of K-means IDS.....	78
Table 4-3 The Performance Evaluation Metrics of the K-means Based IDS In Training Phase.....	78
Table 4-4 Confusion Matrix (Total) of the K-means Based IDS Training Stage.....	78
Table 4-5 Total Performance of the K-means Based IDS Training Phase.....	79
Table 4-6 the Number of Record Class In 2% Testing NSL-KDD Data Set.....	79
Table 4-7 Confusion Matrix of Testing Stage of K-means IDS.....	80
Table 4.8 The Performance Evaluation Metrics of the K-means Based IDS In Testing Phase.....	81
Table 4-9 Confusion Matrix (Total) of the K-means Based IDS testing stage.....	81
Table 4-10 Total Performance of the K-means Based IDS Testing Phase.....	81
Table 4-11 Confusion Matrix of Testing Stage of Proposed Novel KNN IDS.....	83
Table 4-12 the Performance Evaluation Metrics of the Novel K-NN Based IDS.....	83
Table 4-13 Confusion Matrix (Total) of the Novel K-NN Based IDS.....	84

Table 4-14 Parameter and Topology of our Proposed Neural Network of first Experiment.....	86
Table 4-15 the Number of Record Class in Less Than 5% of Training NSL-KDD.....	86
Table 4-16 First Experiment Confusion Matrix of Training Stage of BPRLS IDS.....	87
Table 4-17 First Experiment Performance Evaluation Metrics of BPRLS IDS in Training Phase.....	87
Table 4-18 First Experiment Confusion Matrixes (Total) of the IDS BPRLS Training Phase.....	87
Table 4-19 Total Performance of the IDS BPRLS Training Phase.....	88
Table 4-20 the Number of Record Classes in Less Than 1% of Testing NSL-KDD Data Set.....	89
Table 4-21 First Experiment Confusion Matrix of Testing Stage of BPRLS IDS.....	89
Table 4-22 First Experiment Performance Evaluation Metrics of BPRLS IDS in Testing Phase.....	90
Table 4-23 First Experiment Confusion Matrixes (Total) of the IDS BPRLS Testing Phase.....	90
Table 4-24 Total Performance of the IDS BPRLS Testing Phase.....	90
Table 4-25 Parameter and Topology of our Proposed Neural Network of Second Experiment.....	92
Table 4-26 Second Experiment Confusion Matrix of Training Stage of BPRLS IDS.....	92
Table 4-27 Second Experiment Performance Evaluation Metrics of BPRLS IDS in Training Phase.....	93
Table 4-28 Second Experiment Confusion Matrixes (Total) of the IDS BPRLS Training Phase.....	93

Table 4-29 Total Performance of the IDS BPRLS Training Phase.....	93
Table 4-30 Second Experiment Confusion Matrix of Testing Stage of BPRLS IDS.....	94
Table 4-31 Second Experiment Performance Evaluation Metrics of BPRLS IDS in Testing Phase.....	94
Table 4-32 Second Experiment Confusion Matrixes (Total) of the IDS BPRLS Testing Phase.....	94
Table 4-33 Total Performance of the IDS BP R LS Testing Phase.....	95

Table of Figures

Figure 2-1 K-means Algorithm Flow Chart.....	10
Figure 2-2 Back Propagation Algorithm Flow Chart.....	14
Figure 2-3 Connection Neurons.....	15
Figure 2-4 (ML P) Structure with Linear Combiner.....	17
Figure 2-5 Intrusion Detection Systems.....	28
Figure 2-6 Artificial Neurons.....	33
Figure 2-7 Feed Forward Neural Network.....	34
Figure 2-8 Supervised Learning.....	35
Figure 3.1 the Stage of Data Pre-Processing Phase.....	46
Figure 3-2 Small Set of NSL-KDD Data Set Connection Records.....	47
Figure 3-3 Mat Lab Implementation of Protocol Feature Coding.....	48
Figure 3-4 Mat Lab Implementation of Service Feature Coding.....	50
Figure 3-5 Mat Lab Implementation of Flag Feature Coding.....	52
Figure 3-6 Mat Lab implementation of Label Feature coding.....	53
Figure 3-7 Mat Lab Implementation of Zscore Normalization.....	55
Figure 3.8 Proposed K-means IDS Block Diagram.....	57
Figure 3-9 Proposed Novel K-NN IDS Block Diagram.....	62
Figure 3-10 Proposed BPRLS IDS Module Block Diagram.....	66
Figure 4-1 The Statistics of record Classes in K-means IDS Training Stage.....	77
Figure 4-2 The Statistics of record Classes in K-means IDS Testing Stage.....	80
Figure 4-3 Graphical Comparison of K-means IDS Training and Testing Stage.....	82

Figure 4-4 Graphical Detection Rate Comparison Between K-means and Novel K-NN IDS.....	84
Figure 4-5 The Statistics of record Classes in BPRLS IDS Training Stage.....	86
Figure 4-6 The Statistics of record Classes in BPRLS IDS Testing Stage.....	89
Figure 4-7 Graphical Detection Rate Comparison between BPRLS IDS Training and Testing Stage of First Experiment.....	91
Figure 4-8 Graphical Detection Rate Comparison between BPRLS IDS Training and Testing Stage of Second Experiment.....	95
Figure 4-9 Graphical Detection Rate Comparison between our Intrusion Detection Systems.....	96
Figure 4-10 Graphical Accuracy Comparison between our Intrusion Detection Systems.....	97
Figure 4-11 Graphical Comparison between the Intrusion Detection System and other Intrusion Detection Systems.....	98

Abbreviations

ANN	Artificial Neural Network
AI	Artificial Intelligent
ACC	Accuracy Rate
BPRLS	Back Propagation Recursive Least Squares Algorithm
DoS	Denial Of Service
DR	Detection Rate
Es	Expert Systems
FL	Fuzzy Logic
FP	False Positive
FN	False Negative
FPR	False Positive Rate
FNR	False Negative Rate
GA	Genetic Algorithms
HIDS	Host-Based Intrusion Detection Systems
Ids	Intrusion Detection System
KDD	Knowledge Discovery and Data Mining
K-NN	K-Nearest Neighbor
MSE	Mean Square Error
MLP	Multi layer perceptron
NIDS	Network Based Intrusion Detection systems
Prob	Propping Attack
R2L	Remote To Local Attack
SIM	Cosine Similarity
TCP	Transmit ion Control Protocol
TP	True Positive
TN	True Negative
TPR	True Positive Rate
TNR	True negative rate
U2R	User To Root Attack

خوارزمية الانتشار المرتد أصغر التربيعة التكرارية لأنظمة كشف التطفل إعداد

عمر هشام رشيد السعدون
إشراف الاستاذ الدكتور رياض نعيم
ومشرف مشارك الدكتور صادق الحموز

الملخص Abstract

منذ التزايد في الهجمات الخبيثة على شبكات الحاسوب السلكية والأنظمة اللاسلكية وبأنواع مختلفة معروفة وغير معروفة جعل التقنيات الأمنية التقليدية مثل الجدر النارية (Firewalls) والبرامج المتاحة من مضادات الفيروسات غير كافية لتزويد شبكات حاسوب متكاملة، أمنة وموثوقة. أنظمة كشف التطفل تعتبر واحدة من التقنيات المجربة والفعالة والتي تستخدم في السيطرة على حركة المرور في الشبكات بحيث تميز أي نوع من سوء استخدام شبكة الحواسيب أو أي نوع من الاستخدام غير المسموح لها وهذا يجعل تطبيق نظام كشف التطفل أمراً حاسماً في شبكات الحاسوب.

تشكل أنظمة كشف التطفل جزءاً أساسياً من منظومة الدفاع في الشبكات. إلا أنه يعتبر تقنية غير ناضجة وغير مكتملة بعد. وهذه الحقيقة أعطت فرصة كبيرة أمام علم التنقيب عن البيانات (Data Mining) لخلق الكثير من الإسهامات في مجال علم كشف التطفل.

في هذه الرسالة، تم اقتراح ثلاث من تقنيات التنقيب عن البيانات لتحسين كشف التطفل:
خوارزمية (K -Means)، خوارزمية ($Novel K$ -nearest neighbours KNN)، والشبكة العصبية
Recursive Least Squares Back Propagation Neural Network (RLSBP NN)

تشترك الأنظمة المقترحة في ثلاث مراحل: أولاً، يتم معالجة قاعدة البيانات (NSL-KDD) ثم يستخدم (8%) من البيانات المعالجة لتدريب الخوارزميات. أخيراً، يستخدم الجزء الآخر من البيانات (2%) في سبيل تقييم الأداء. تم استخدام معدل كشف التطفل، معدل الدقة، FPR، TNR، FNR، كمقاييس للأداء.

تم تطبيق واختبار أنظمة كشف التطفل المقترحة في بيئة MATLAB 2012 حيث أظهرت النتائج تفوق الأنظمة المقترحة واستقرارها العالين من حيث معدل كشف التطفل، الدقة (Accuracy)، الحساسية (Sensitivity)، والخصوصية (Specificity) ولكل أنواع الهجمات. تم مقارنة نتائج الأنظمة المقترحة مع بعضها البعض وتم مقارنتها مع أنظمة أخرى. حيث أثبتت هذه المقارنات موثوقية وفعالية الأنظمة المقترحة.

Back Propagation Recursive Least Squares Algorithm for Intrusion Detection Systems

**Prepared by
Omar Hisham Rasheed Al-sadoon**

**Supervised by
Prof. Dr. Reyadh Shaker Naoum
Co Supervisor: Dr. Sadeq Al-Hamouz**

Abstract

Since malicious attacks have increased on wire computer networks and wireless systems, in various known and unknown types, conventional security techniques such as firewalls and available anti-viruses programs are not adequate to provide reliable, integrated and secure networks. Intrusion Detection Systems (IDSs) are considered one of the most reliable and tested technologies that used to control the network traffic in order to distinguish any kind of computer system network mishandling or identifying any kind of unauthorized usage, which make the implementation of intrusion detection system a critical issue in computer networks.

Intrusion detection systems form a principal part of the network system defence; however, intrusion detection is not yet a perfect and mature technology. This fact provided a great opportunity for data mining to create many leading contributions in the field of intrusion detection.

In this thesis, we have proposed three data mining techniques: *K-Means* algorithm, *Novel K- nearest neighbours (KNN)* algorithm and *Recursive Least Squares Back Propagation Neural Network (RLSBP NN)* for intrusion detection fortification.

The proposed techniques have three common major phases; in first NSL-KDD dataset, pre-processing is performed, and then part of processed dataset (8%) is used to train the proposed algorithms. Finally, the other part of dataset (2%) is used in sake of performance evaluation. Data set used is NSL-KDD dataset and we have used detection rate (recall), accuracy, FPR, TNR, TPR, and FNR as evaluation measurements.

Our proposed IDSs are implemented and tested in MATLAB 2012 environment, where the experimental results shows superiority and high stability of our proposed systems in terms of detection rate ,accuracy, sensitivity, and specificity for all types of intrusions attacks. The results of our proposed techniques are compared with each other and with other existing schemes. These comparisons proved the reliability and effectiveness of the proposed techniques.

Chapter one

Introduction

Chapter one

Introduction

1.1 preface

In recent years, numerous computers are hacked due to low precautions in protecting them against various attacks for the network. Furthermore; the companies and organizations are in Performance risk if there are problems in maintaining the security inside the whole system. Millions of dollars are usually paid as cost for single attack; thus the need to upgrade computers and networks security levels became so crucial to defend critical infrastructure against diverse threats. With the aggravation of electronic crimes; the design of information infrastructure that is safe-guarding, such as; “Intrusion Detection System (IDS)” that can be employed for the detection and prevention of incidents is now considered great challenge issue. (Jaiganes et al, 2013).

The security of a computer networks was defined by Bottino et al. (2006) as the level of users trust for the confidentiality and accuracy of the data exchanged inside the system. They stated that *interception, denial-of service, masquerade and modification* are considered the major threats for good policy of security policy. They also dedicated that the *confidentiality, availability, authentication and integrity* are considered the major parameters for a good policy of security. The property of authentication is used for the purposes of persons’ identity verification and enable them accessing the information in addition to preventing process is used to verify the identity of the person(s) with an access to information and defends against masquerade. The confidentiality property is used to ensure defending against the interception of information that is performed by unauthorized persons. The integrity property defends against data modification by verifying that the data has not been changed, destroyed or

lost in any manner. The availability property is recognized as the permissibility and authorization for to accessing and preventing the events of denial-of service (DoS).

It is reported by *Sharma et al.* (2012) that the attacks number aiming to steal secret and private information, such as; the number for credit card, passwords in addition to any critical financial information get rise day by day; it has increased from 9 million attacks in June 2004 to more than 33 million within less than one year . So the researchers concluded that the importance of the security within the network increases day by day due to the great growth that occurred recently within services that are based on networks, In addition to the sharing of highly sensitive information on these networks.

The currently available software's and applications are keeping our data, system, and networks secure through the use of various security techniques including detection systems, prevention systems, firewalls and anti-viruses (Singh & Chandra, 2014).

From safety management perspective, *Tian et al.* (2009) mentioned that traditional security techniques such as static technologies, authentication systems, and firewall are playing certain role in preventing the illegal intrusion, but the defense strategy alone is not enough.

IDS can be defined as the system that is designed for attacks detection; that occurs in spite of the precautions of security. IDS is now considered as one of the main technologies that are employed for managing the traffic of the network in addition to identification of the network intrusions, such as; DoS, Probe, "User to Root (U2R)" and "Remote to User attacks (R2L)" (Al-Sharafat & Naoum 2009).

1.2 Problem Identification

According to *Norouzian et al.* (2011), the most current approaches for detecting intrusions are utilizing some forms of predefined rule-based analysis which are generated by the system or supplied by the administrator. “Expert systems (ES)” are considered widespread type of intrusion detection systems of rule-based type. ES includes set of rules that employed by the system in order to build conclusions regarding to the security-related data from IDS mimicking the encoding knowledge of a human “expert”.

ES unfortunately needs regular updates to stay current; this consequently results in inflexible IDS that cannot detect the attacks within the system; once the event sequence is slightly diverse from the predefined profile. Furthermore; increasing the rule-base abstraction level provides only a partial solution to this weakness point. So the actual problem is that the hacker or the intruder is performing as a flexible and intelligent agent compared to the rule-based IDSs which act by obeying fixed rules. So the expert systems are keeping suffering from updating, searching and matching the rule sets (Norouzian & Merati, 2011). In order to overcome these defects, *Moradi & Zulkernine* (2004) suggested using the applications for soft computing methods in IDSs, such as “*fuzzy logic (FL)*”, “*artificial neural networks (ANNs)*”, “*probabilistic reasoning (PR)*”, and “*genetic algorithms (GAs)*”.

As the neural networks are able to be generalized from the learned data, in addition of being broadminded of inexact and uncertain information, thus, they seemed to be an suitable approach to IDS (Jing, et al., 2010). Hence, the “Artificial Neural Networks (ANNs)” were applied effectively for IDs development they have the benefit of easier representation for nonlinear relationship among the output and input along with their inherent speed of computation. Even if the data were imperfect or distorted,

ANNs are proven to be still able to analyze the data from the network (Poojitha, et al., 2010).

1.3 Questions

- Can we classify the connection records of the NSL-KDD'99 dataset according to their types *Normal*, *DoS*, *R2L*, *U2R*, and *Prob* via applying (*K-means*, *Novel K-NN*, and *BPRLS*) Algorithms?
- Can we minimizing the “False Positive Rate (FPR)” and “False Negative Rate (FNR)”, with maximizing the detection rate (DR)’ when when we apply (K-means, Novel K-NN, and BPRLS) Algorithms for intrusion detection?
- Can we reduce the “mean square error (MSE)”, when we use (BPRLS) Algorithm?

1.4 Study Objectives

- 1- To apply k-means machine learning algorithm for intrusion detection.
- 2- To apply Novel k-nearest neighbor machine learning algorithm for intrusion detection.
- 3- To use multilayer neural network, and training it by using Back Propagation Recursive Least Squares Algorithm for intrusion detection.
- 4- To distinguish and classify attack connection records via applying:
 - a. k- Means machine learning algorithm.
 - b. Novel k-nearest neighbor machine learning algorithm.
 - c. Back Propagation Recursive Least Squares Algorithm.
- 5- To compare results obtained from the following algorithms:
 - a. k-means machine learning algorithm,

- b. Novel k-nearest neighbor machine learning algorithm,
- c. Back Propagation Recursive Least Squares Algorithm.

1.5 Contribution

The contribution of this thesis is summarized in the following points:

- ❖ Select the best centers” *Normal, DoS, R2L, U2R, Prob*” from NSL-KDD99 data set for each cluster when applying k-means clustering algorithm in training phase to enhance the results for this algorithm in training phase and obtain the optimal centers” *Normal, DoS, R2L, U2R, Prob*” to used it later in testing phase and improve the results for this algorithm totally with intrusion detection.
- ❖ Use Euclidian distance with cosine similarity when applying Novel K-nearest neighbor as measure distance between the test records and trained records to improve the results for this algorithm with intrusion detection.
- ❖ Use multilayer neural network and training it via (BPRLS) hybrid algorithm with deferent numbers of neurons for each internal layer to improve the results for this algorithm.
- ❖ Use the optimal parameters” steps size, forgetting factor” when applying (BPRLS) hybrid algorithm to improve the results for this algorithm.
- ❖ Obtain convergence when applying (BPRLS) hybrid algorithm between the real Input and the demand output from only 30 epochs.
- ❖ Use Z-score normalization instead of min-max normalization to improve the results for all algorithms.

1.6 Significant

In a line with the internet revolution that was occurred recently, the needs for security within computer systems and networks increased day by day. The intrusion detection systems are considered one of the most essential and critical security aspects that must be considered. This research introduces new scheme for intrusion detection purposes; it is mainly designed in order to detect and then classify the most common intrusion types that may be occurred within computer systems and networks. This detection is considered essential in order to overcome the security vulnerabilities. Furthermore; there will be ability for avoiding these intrusions via enhancing the security and privacy levels within the system.

1.7 Limitations

- The proposed system deals only with the detection of intrusion and it must be developed to include the prevention.
- The proposed system deals only with single CPU.
- The (BPRLS) system includes only two hidden layers.

1.8 Researches published

- Al-Hamouze,S,Naoum,R,&Al-Sadoon,O(2015). The Novel K-Nearest Neighbor and Back Propagation Recursive Least Squares Algorithms for Intrusion Detection Systems. *International journal of scientific Research* ,Vol.4
- Naoum,R, Alsadoon,O& Abughazleh,A(2015) A network Intrusion Detection System Using Recursive Least Squares Multilayer Neural networks, *European Journal of scientific Research*, Vol 129 No 2 PP(131-141).

Chapter Two
Theoretical Background and
Literature Review

Chapter two

Theoretical Background and Literature Review

2.1 Overview

Intrusion detection systems have attained a great deal of importance during the last years; several researches and studies were introduced and investigated using several methods. The three main techniques that will be used to implement the (IDS) will be introduced and investigated during this chapter. Two of these techniques are based on learning machine, which are; k-means and novel k-nearest neighbor algorithms. The third algorithm is “Back Propagation Recursive Least Square (BPRLS)” that is based on Neural Networks (NNs). The chapter also includes an investigation for the normalization process including its main equations. The (IDS) will be then introduced in terms of their definition, functions, the criteria that each (IDS) should subject, the ways and the architectures in addition to the things that must be considered during the implementation of any (IDS).

The chapter will also address the four possible attacks for any computer network, and the criteria that can be used to evaluate the effectiveness of (IDS). The “Artificial Neural Networks (ANNs)” will be then introduced and investigated including the process of biology cell in human brain. The process of Multilayer (NN) will be then investigated and the supervised learning for (ANNs). The second part of this chapter is the literature review that reviews some of the recent works about the (IDSs) using the three considered algorithms in addition to the Genetic Algorithm.

2.2 Theoretical Background

2.2.1 K-mean Clustering Algorithm

The type of learning for this algorithm is unsupervised learning, this algorithm group any source of objects depend on less distance to the centre for each cluster, The process of learning is depend on Training paradigms that we submit to this algorithm to group our objects depend on attributes in to number of clusters, this process is done by reducing the sum of squares of distance between the objects and cluster centres, (Teknomo, 2006).

Figure 2-1 represents the flow chart of k-means algorithm

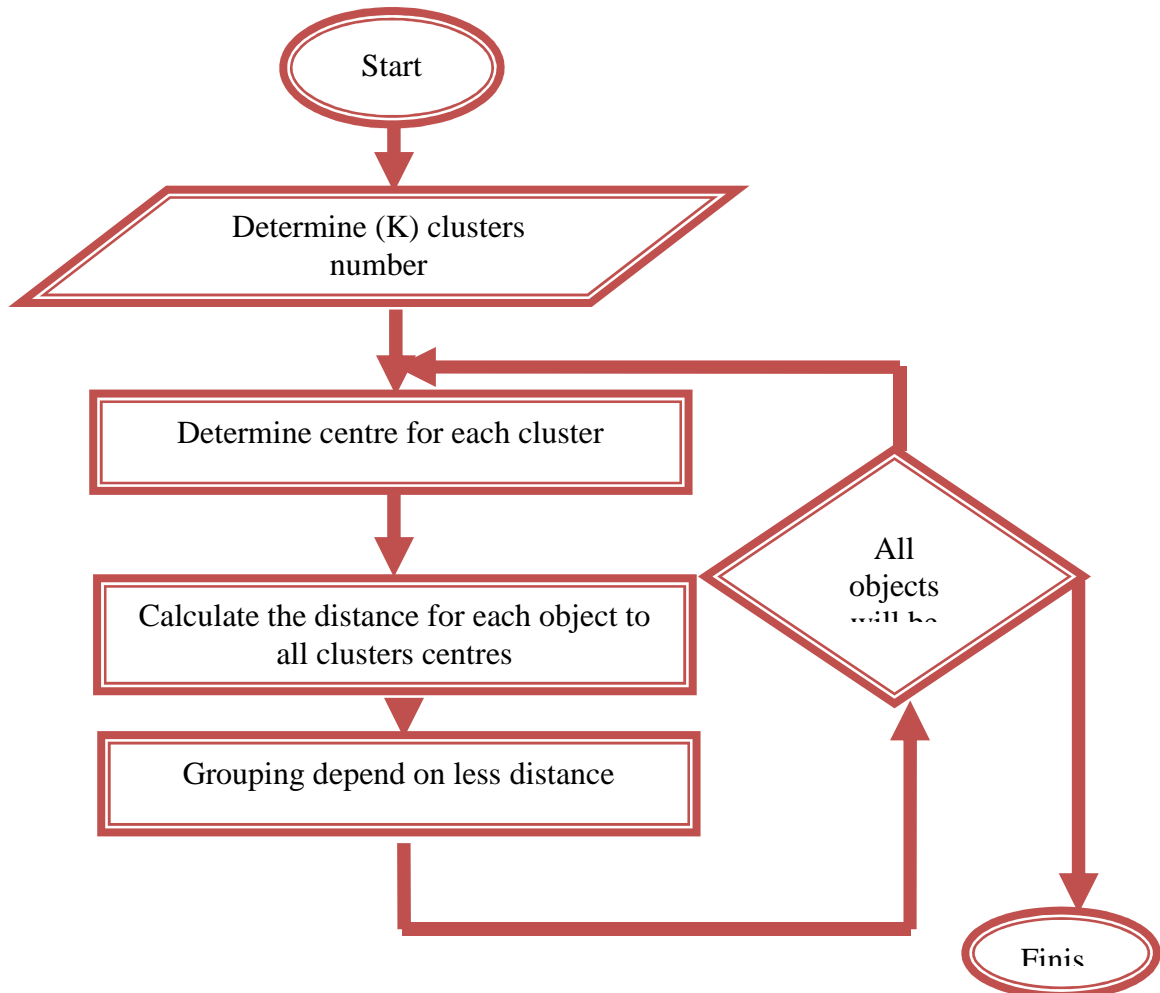


Figure 2-1 K-means algorithm flowchart, source (Teknomo, 2006)

K- Means clustering algorithm steps (Nazeer&Sebastian,2009)

Inputs

$\mathbf{r} = \{r_1, r_2, \dots, r_n\}$ // set of \mathbf{n} records from NSLKDD99 data set.

\mathbf{K} // clusters number depend on normal records and four main attacks records founded in the data set.

Out put

Groups of \mathbf{K} clusters

- 1- Randomly determine the initial center for each cluster from \mathbf{r} connection records.
- 2- Iterate
 - a. Put each record \mathbf{r}_i in cluster that has the nearest center after calculate the distance measure

$$d(X, Y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \dots + (x_n - y_n)^2}$$

- b. Calculate new center that belong for each cluster.
- c. Go to step-2

Till the centres for all clusters not change (which means the items stables).

The complexity of this algorithm is $\mathbf{o(tkn)}$, where \mathbf{t} clarify the iterative of time for this algorithm, \mathbf{k} is cluster numbers, \mathbf{n} is the number of connection records in the dataset, the difficult process of this algorithm is how to fixing of the initial centres for each clusters ,(Jianliang et al,2009).

2.2.2. Novel K-Nearest neighbour Algorithm (K-NN)

The classifier of K-nearest neighbour (K-NN) type is considered as learning method that is based on instance; it is mainly depend on the similarity function or distance, including; the Euclidean and Cosine. During the research ;(K- NN) of a training data is calculated at the beginning. Then one sample similarities from testing data are tested for the classifiers exactness (Jivani, 2013).

(Al Sultani, 2012) summarizes an ordinary (K-NN) algorithm as following steps:

- The NSL-KDD99 training dataset are stored with its equivalent label.
- The distance is then calculated between each one of the connections within testing data set and those connection used for training.
- The calculated distances are then arranged in ascending order. The firs “Minimum Nearest Neighbor” is then selected considering $k=1$.
- The label for nearest neighbor is then selected; which is considered the test example prediction.
- The above procedure is repeated for all the connections within the considered testing dataset.

Jivani (2013) mentioned that if the number of training records for some classes is much more than the rest then there are chances that these records may get selected in the k nearest neighbour, and the test records would automatically get classified to the majority class instead of the actual class it belongs to. In this proposed algorithm the selection of k nearest records depends on the parameter n . This parameter is taken as input from the user and it depends on the size of the smallest class.

Novel k- nearest neighbour algorithm steps

1. First select the **n** nearest neighbors of connection record NSLKDD99 x_j from each class founded in the training set, the value of **n** should not be greater than the size of the smallest class.
2. Sort these **n** nearest neighbors in descending order of the similarity- $\text{sim}(r_i, x_j)$.

$$\text{Sim}(r_1, r_2) = \frac{\vec{v}(r_1) \cdot \vec{v}(r_2)}{\|\vec{v}(r_1)\| \cdot \|\vec{v}(r_2)\|} \dots\dots\dots (2.1)$$

3. Select now the top k nearest neighbours from the list prepared in step 2. These are the final **k** nearest neighbours of the records.

4. Using the decision rules given in (2.2) or (2.3), now find the class to which record r_i is most similar to.

$$Y(r_i) = \max_k \sum x_j \square \text{KNN } y(x_j, c_k) \dots\dots\dots (2.2)$$

$$Y(r_i) = \max_k \sum x_j \square \text{KNN Sim}(r_i, x_j) y(x_j, c_k) \dots\dots\dots (2.3)$$

Where r_i is the record to be classified, x_j is one of the neighbors of r_i and $y(x_j, c_k) \square \{0, 1\}$ indicates whether Record x_j belong to class c_k or not, $\text{sim}(r_i, x_j)$ is the similarity measure between the test record r_i and its neighboring record x_j . This is generally the cosine similarity in (2.1).

2.2.3 Back Propagation (BP) Algorithm

This algorithm is a recursively algorithm originate to reduce the “mean square error (MSE) “between the real output of “feed forward neural networks (FFNNs)” and the desired output that we need.

The process of learning for this algorithm is done by paradigms, we submit these paradigms to the neural network and it understand the mission by modify its weight for

each layer , when the learning process complete , the net will produce the demand out for these paradigms ,Jaiganesh et al ,(2013) .

Figure 2.2 represent the flowchart of back propagation algorithm

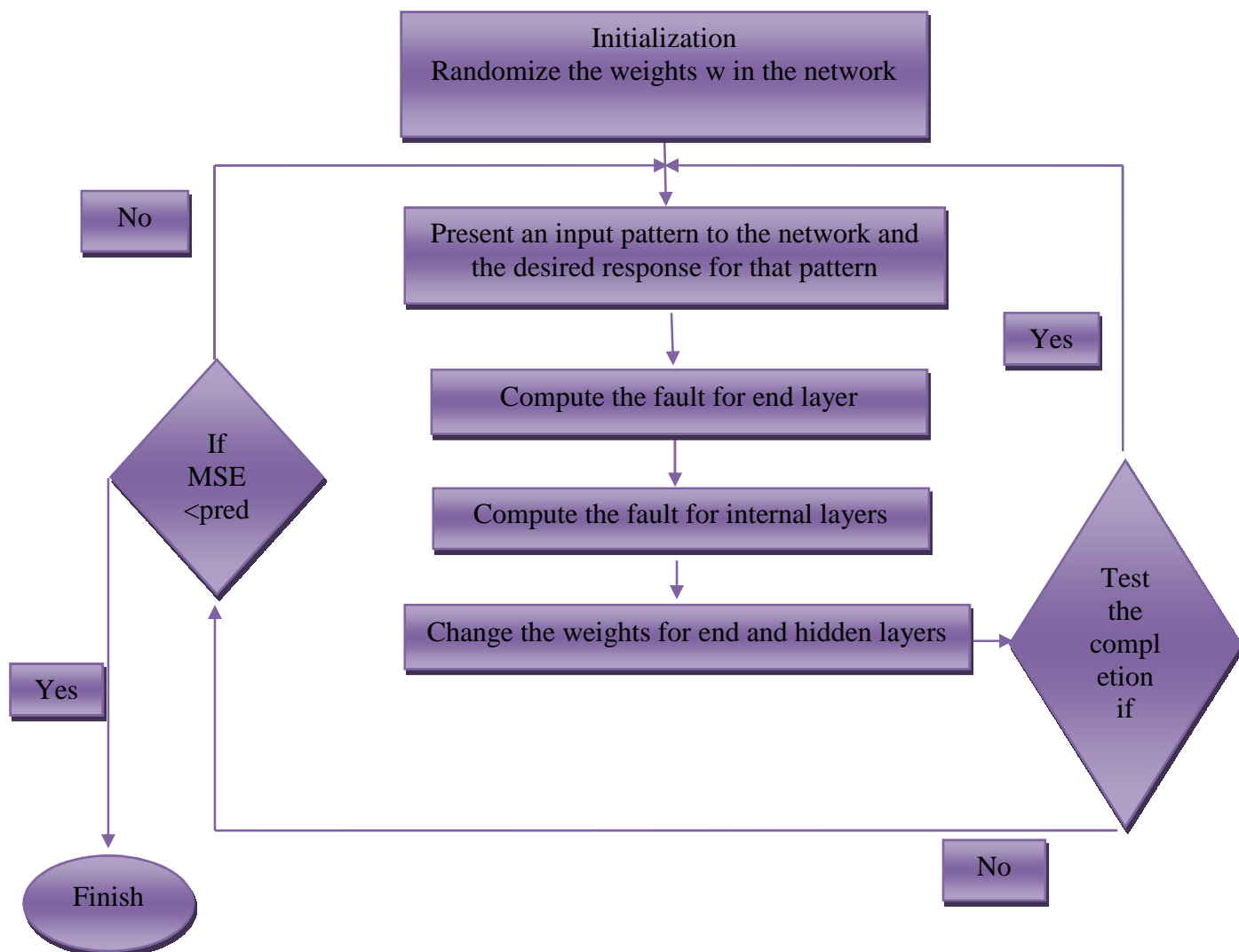


Figure 2.2 back propagation algorithm flowchart, source (alsadoon,O, 2014)

Jaiganesh et al, (2013) summarize the basic steps of back propagation algorithm with respect to figure 2.3 bellow

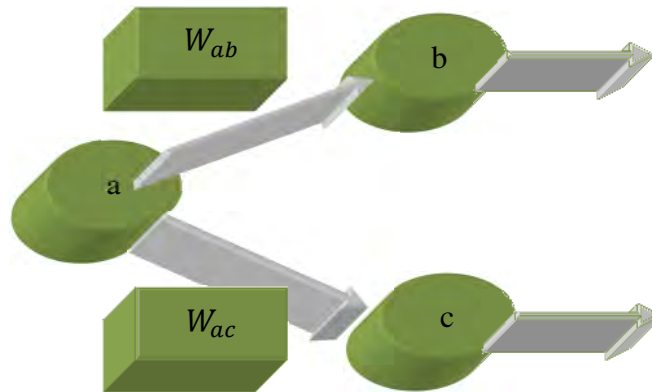


Figure 2.3 connection neurons; source (Jaiganesh et al, (2013))

figure 2.3 shows the relation between neuron **a** that belong to the internal layer and neurons **b** that belong to the end layer has the weight W_{ab} , also shows the relation between neuron **a** and neurons **c** has the weight W_{ac} .

Step (1): precede the paradigm to the net.

Step (2): run that paradigm throw the net till reach the output layer.

Step (3): Calculate the fault for end neurons **b**, **c**, and this fault can be calculated as the following formulas:

$$\text{Fault } \mathbf{b} = \text{produce } \mathbf{b} (1 - \text{produced } \mathbf{b}) (\text{goal } \mathbf{b} - \text{produced } \mathbf{b})$$

$$\text{Fault } \mathbf{c} = \text{produce } \mathbf{c} (1 - \text{produced } \mathbf{c}) (\text{goal } \mathbf{c} - \text{produced } \mathbf{c}).$$

Step (4): Calculate the fault for internal neuron **a**, by taking the fault for neurons **b**, **c**, and back propagate it through the weight to obtain the internal layers faults, from this process came the name for this algorithm.

$$\text{Fault } \mathbf{a} = \text{produce } \mathbf{a} (1 - \text{produced } \mathbf{a}) (\text{Fault } \mathbf{b} * W_{ab} + \text{Fault } \mathbf{c} * W_{ac}).$$

Step (5): modify the weights, suppose W_{+ab} be the trained weight, and W_{ab} be the old weight.

$$W_{+ab} = W_{ab} + (\text{Fault } \mathbf{b}^* \text{ produced } \mathbf{a})$$

$$W_{+ac} = W_{ac} + (\text{Fault } \mathbf{c}^* \text{ produced } \mathbf{a})$$

Step (6): repeat by going to step (3) till the “mean square error (MSE) “approaching to zero.

2.2.3.1 Back Propagation recursive Least Squares (BPRLS) Algorithm

Recent researches and applications in artificial neural networks uses conventional back-propagation that described by McClelland and Rumelhart (1986). This algorithm works well for simple systems or small nets, however it converges slowly and depending on the initial weights used in training process especially for complex and large systems where thousand, may be millions of patterns run through the network. Back propagation shows poor convergence for complex systems even if the used dataset is small (Waible et al., 1989).

Scalero and Tepedelenlioglu (1992) proposed to minimize the “mean square error (MSE) “between the actual summation output and the desired summation output of the network rather than to minimize the (MSE) between the actual output and desired output of the network. by this approach the problem is transferred from the nonlinear world of the network (consist of the nonlinear activation function and it’s inputs and outputs) to the linear world of the network (consists of the summer and it’s inputs and outputs) where the output of the summer is the actual summation output and $f^{-1}(\text{Desired Output})$ is the desired summation output, and the error is the difference between them.

Figure 2.4 represent (MLP) structure with linear combiner, (Ham&Kostanic,2000)

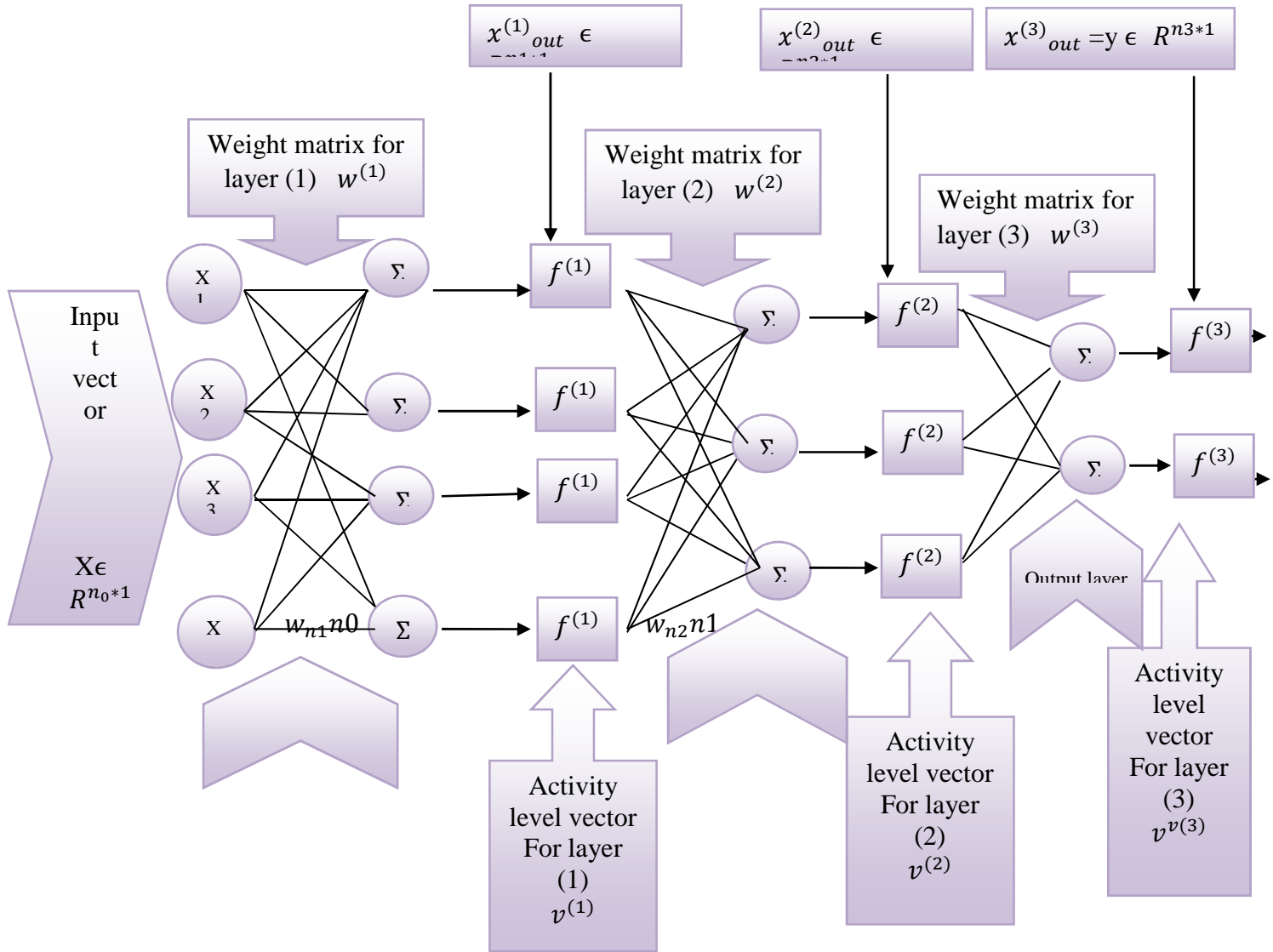


Figure 2.4 (MLP) structure with linear combiner, source (Ham&Kostanic,2000)

2.2.3.1.1 Normal Equations for the Linear Combiner

Referencing to figure 2.4 (Ham& Kostanic,2000) i^{th} linear combiner in the s^{th} layer of the (MLP NN) , when the q^{th} input pattern is presented to the network ,the output of combiner is calculated as an inner product between the combiner weights $w_i^{(s)}$ and the input vector to the particular layer $x_{out,q}^{(s-1)}$ that is,

$$v_i^{(s)} = w_i^{(s)t} \cdot x_{out,q}^{(s-1)} \dots\dots\dots (2.1)$$

Suppose the desired output for the particular combiner is $d_{i,q}^{(s)}$ is known for every pattern in the training set ,training the (MLP NN) effectively assumes training all its linear combiners, the goal of the learning algorithm is to minimize the squared error cost function, given by

$$j_i^{(s)} = \frac{1}{2} \sum_{q=1}^m (d_{i,q}^{(s)} - v_i^{(s)})^2 \dots\dots\dots (2.2)$$

Where m represents the whole number of vectors founded training data set, (2.2) can be writing

$$j_i^{(s)} = \frac{1}{2} \sum_{q=1}^m (d_{i,q}^{(s)} - w_i^{(s)t} \cdot x_{out,q}^{(s-1)})^2 \dots\dots\dots (2.3)$$

To find the weight vector which minimizes the cost function given in (2.3), we take the partial derivative with respect to $w_i^{(s)}$ and equate it to zero, that is

$$\frac{\partial j_i^{(s)}}{\partial w_i^{(s)}} = \sum_{q=1}^m (-d_{i,q}^{(s)} x_{out,q}^{(s-1)} + x_{out,q}^{(s-1)} x_{out,q}^{(s-1)t} w_i^{(s)}) = 0 \dots\dots\dots (2.4)$$

Defining

$$c_i^{(s)} = \sum_{q=1}^m x_{out,q}^{(s-1)} x_{out,q}^{(s-1)t} \dots\dots\dots (2.5)$$

And

$$p_i^{(s)} = d_{i,q}^{(s)} x_{out,q}^{(s-1)} \dots\dots\dots (5.6)$$

Equation (2.4) can be rearranged in vector matrix form as

$$c_i^{(s)} w_i^{(s)} = p_i^{(s)} \dots\dots\dots (2.7)$$

Where $c^{(s)}$ can be interpreted as an estimate of the covariance matrix of the input to the s^{th} layer, $p_i^{(s)}$ is the cross-correlation vector between input to the s^{th} layer and the demand output of the i^{th} linear combiner in the s^{th} layer, $w_{i(s)}$ is the weight vector to the i^{th} linear combiner in the s^{th} layer, equation (2.7) is referred to as deterministic normal equation in the context of adaptive filtering, if the covariance matrix $c^{(s)}$ and cross-correlation vector $p_i^{(s)}$ are known's the appropriate weight vector can be solved by using one of the standard technique for solving a system of linear equation

$$w_{i(s)} = [c^{(s)}]^{-1} p_i^{(s)} \dots\dots\dots (2.8)$$

2.2.3.1.2 Inverse Function

The demand output of the particular node in the output layer is known, the desired summation for those nodes in the output layers can be computed according to the inverse function in formula (2.9).

$$v^{\wedge} i^{(s)} = f^{(-1)}(d_{i,q}^{(s)}) \dots\dots\dots (2.9)$$

We have the desired summation for the nodes in the output layers as

$$v^{\wedge} i^{(s)} = \frac{1}{\sigma} \ln \frac{1+d_{i,q}^{(s)}}{1-d_{i,q}^{(s)}} \dots\dots\dots (2.10)$$

Return to equation (2.8) we do not have explicate knowledge of either the covariance matrices or cross correlation vector, and over the course of training of the network they have to be estimated, the estimate of the correlation matrix for the s^{th} layer can be written as

$$c_i^{(s)}(k+1) = b c_i^{(s)}(k) + x_{out,q^{(s-1)}} x_{out,q^{(s-1)}}^t \dots\dots\dots (2.11)$$

The cross correlation vector for each the linear combiner can be estimated as

$$p_i^{(s)}(k+1) = p_i^{(s)}(k) + v^{\wedge} i^{(s)} x_{out,q^{(s-1)}} \dots\dots\dots (5.12)$$

Where, b coefficient in (2.11) is called the forgetting factor in the range 0.9 to 0.99. Equation, (2.11), is in recurrent form, what one need in recursive equation for the inverse autocorrelation matrix $[c^{(s)}]^{-1}$ this can be achieved by using matrix inversion lemma or the kalman filter.

$$k^{(s)}(k) = \frac{[c^{(s)}(k-1)]^{-1} x_{out(s-1)}(k)}{b + x_{out(s-1)}^T(k) [c^{(s)}(k-1)]^{-1} x_{out(s-1)}(k)} \dots\dots\dots (2.13)$$

And the update of the inverse matrix

$$[c^{(s)}(k)]^{-1} = b^{(-1)} \{ [c^{(s)}(k-1)]^{-1} - k^{(s)}(k) x_{out(s-1)}^T(k) [c^{(s)}(k-1)]^{-1} \} \dots\dots\dots (2.14)$$

(BPRLS) Algorithm steps

The proposed algorithm that used to train the recursive least squares back propagation neural network (Scalero et al.1992) that consist the heart of the proposed system is as following:

Step (1): initialize

- Randomize all weight in the network.
- Set the forgetting factor between 0.9-0.99.
- Set the back propagation step size from 10 to 400.
- Set the initial value of the inverse matrix $[R^{(j)}]^{-1}$ to I, I represents an identity matrix
- Set the slop of sigmoid activation function to 0.2

Step (2): select training pattern

Select a training pattern randomly. The input is the connection record \mathbf{X}_0 and the output is the label of the record (attack type) denoted by $\mathbf{0}$.

Step (3): Run the selected input/desired output pattern through the network and evaluate the summation output for each layer from the hidden layer j through L

$$y_{jk} = \sum_{i=0}^N (x_{j-1}, w_{jki})$$

And the activation function output is:

$$x_{jk} = f(y_{jk}) = \frac{1 - \exp(-ay_{jk})}{1 + \exp(-ay_{jk})}$$

Step (4): Calculate the Kalman gain $K_j(t)$ for each jth layer to update R_j^{-1} correlation matrix:

$$K_j(t) = [R_j^{-1}(t-1)x_{j-1}(t)]/[b_j + x_{j-1}^T(t)R_j^{-1}(t-1)x_{j-1}(t)]$$

$$R_j^{-1}(t) = [R_j^{-1}(t-1) - K_j(t)x_{j-1}^T(t)R_j^{-1}]/b$$

Step 5: Back -propagate the error signals through the network

$$e_L = f'(y)(O - x) \quad \text{For output layer L}$$

And the error signal for the interior hidden layer (j) is:

$$e_j = f'(y_j) \sum_j (e_{j+1} * w_{j+1})$$

Where $f'(y_j)$ is the first derivative of the activation function $f(y_j)$ and in our case the derivative is given by:

$$f'(y_{jk}) = 2a[\exp(-ay_{jk})]/[1 + \exp(-ay_{jk})]^2$$

Step 6: Update the weights vector in each layer j up to the output layer L

$$\mathbf{w}_{Lk}(\mathbf{t}) = \mathbf{w}_{Lk}(\mathbf{t} - 1) + \mathbf{K}_L(\mathbf{t}) (\mathbf{d}_k - \mathbf{y}_{Lk}) \quad \text{For output layer}$$

$$\mathbf{w}_{jk}(\mathbf{t}) = \mathbf{w}_{jk}(\mathbf{t} - 1) + \mathbf{K}_j(\mathbf{t}) (\mu \mathbf{e}_{jk}(\mathbf{t})) \quad \text{For } j^{\text{th}} \text{ hidden layer}$$

Where, \mathbf{d}_k is the desired summation output of k^{th} node of the L^{th} layer, and is given by:

$$d_k = \frac{1}{a} \ln \frac{1+o_k}{1-o_k}$$

Step (8): test the completion

Use the MSE of the network output as convergence test.

2.3 Data Normalizations

Large values input variables to the neural network, make the neurons reaching the saturated condition, which means that the effective of weights associated with each neuron does not change the input values or very small change to produce the demand output and effect to the training process (Poojitha et al (2010)).

The data set must be normalized before entering the training phase based on the mean and (i.e average) and standard deviation. As the following formulas:

- 1- Calculate the mean for respective column

$$\mu = \frac{1}{n} \sum_{i=0}^n xi \dots\dots\dots (\text{Hu et al (2006)}).$$

The formula above can be done using MATLAB function (mean).

- 2- Calculate the standard deviation for respective column

$$\sigma = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (xi - \mu)^2} \dots\dots\dots (\text{Hu et al (2006)}).$$

The above formula can be done using mat lab function (std)

- 3- Normalize the attributes

$$x = \frac{xi - \mu}{\sigma} \dots\dots\dots (\text{Hu et al (2006)}).$$

2.4 Intrusion Detection Systems

2.4.1 Motivation

When the 21st century introduced networks, specifically networks internet, the world benefited from its unlimited chances but along with infinite challenges, for instance, a normal running of a network was able to bring great business, health and educational services developments in addition to new route of financial incomes to the society by initiating a genuine electronic businesses and labour market, yet, the same network was also able to cause unexpected disasters due to insufficient security. The relation between computer security threats and computer safety process insurance can be portrayed as the connection in the midst the bayonet and the armour, thus, in order to deal properly with various unrestrained changing attacks, users employed divers anti-attack tools, where they have an intrusion detection systems (IDSs).

2.4.2 Intrusion Detection Systems definitions

An intrusion can be defined as an unauthorized entry to another's property or area, while from computer sciences perspective it is known as the activities settle the basic computer network security objectives versus to *confidentiality, integrity, and availability*. (Chakraborty, 2013)

Intrusion Detection Systems (IDSs) is the gate that beyond the firewall that can be exploited to discover an assortment of intrusion and also to activate the intercept and the dynamic reactor to the vicious intrusion before any jeopardize operation occurs to the network system, (Yichun et al. 2012).

2.4.3 Intrusion Detection Systems functions

Since a long time ago firewalls were adapted by network security systems; however, many techniques were developed to deceive and elude firewalls. These unsecured performing conditions introduced (IDSs) as a much more advanced security tool comparing to firewalls. (*Bulajoul et al, 2013*).

The main functions of intrusion detection systems were illustrated by (Scarf one & Mell, 2007) as following:

- Recording the information of the observed actions. Usually, information is recorded to be sent into separated systems such as centralized logging servers.
- Notifying the administrator network about important observed actions. Notifying process known as an alert and can occur through several methods, including *e-mails*, *pages* and *messages* on the *user's interface*.
- Producing summery reports regarding the detected events, or providing entire descriptive details on a particular event of concern.

2.4.4 Intrusion Detection Systems criteria

Intrusion detection systems (IDSs) can be categorized into three main criteria including the method of detection, the response upon detection, and the information exporter.

2.4.4.1 The methods of detection

The Methods of detection are either a behaviour-based method “anomaly” or a knowledge-based method “misuse”.

2.4.4.1.1 Misuse Intrusion Detection Systems

Misuse detection (IDS) acts by first designing a pattern for the malicious behaviours and later it will identify intrusion by referring to that designed pattern. The main benefit of misuse detection system is its high level of accuracy in detecting all known attacks; however its deficiency lies in its selectivity toward only detecting intrusions that follow predefined patterns (Wankhade et al., 2013).

2.4.4.1.2 Anomaly Intrusion Detection Systems

Anomaly detection defined a profile for the anticipated behaviour of the network, the profiles of either users or hosts which are representing the normal behaviour should be constructed first. The construction of these profiles is done through the employment of data which were collected at the past and over a period of normal operations; any significant deviations from a previously defined anticipated behaviour are instantly reported as a possible attack despite the fact that not all such deviations would actually be attacks. The main advantage of this approach is its capability of examining unknown and intricate intrusions (Khan et al., 2010).

2.4.4.2 The response upon detection

The responses which are generated by the systems after detecting an intrusion are divided into two types; active and passive responses, in first case the system permeating mechanism relays on automatically stop the intrusion without any kind of human involvement. Once an intrusion is revealed, this type of systems called intrusion prevention systems (IPSs). in second case the system technicality depends on warning the administrator about the intrusion by sending a series of triggering alarms so the

administrator can decide to take the proper action in order to stop the intrusion this type of systems called intrusion detection systems (IDSs), (Khan et al, 2010).

2.4.4. 3 the information exporter

The information exporter is referring to the type of the used information by the system to detect attacks; there are two types for these systems, Host-based intrusion detection system (HIDS), and Network based intrusion detection system (NIDS).

In Host-based intrusion detection system (HIDS) software are installed on each computer in the network in order to superintend all the events which are taking place within that individual host alone, in the case of Network based intrusion detection system (NIDS) the system were designed to monitor the network's traffic in an aim of anatomizing this network protocol activity to identify any dubious activity. Usually, (NIDS) prevailed at the boundaries between networks such as routers, firewalls, virtual private networks, (Chakraborty, 2013).

2.4.5 Intrusion detection systems building steps

When building (IDS) many issues should be taken in consideration, such as:

- Data gathering.
- Data pre-processing
- intrusion detection
- the act of reporting and responding

. Figure 2.5 describes the regulation of (IDS), where sidelines point represents data-control flow and dashed lines point out the rejoinder to intrusive activities (ELSHOUSH and OSMAN, 2011).

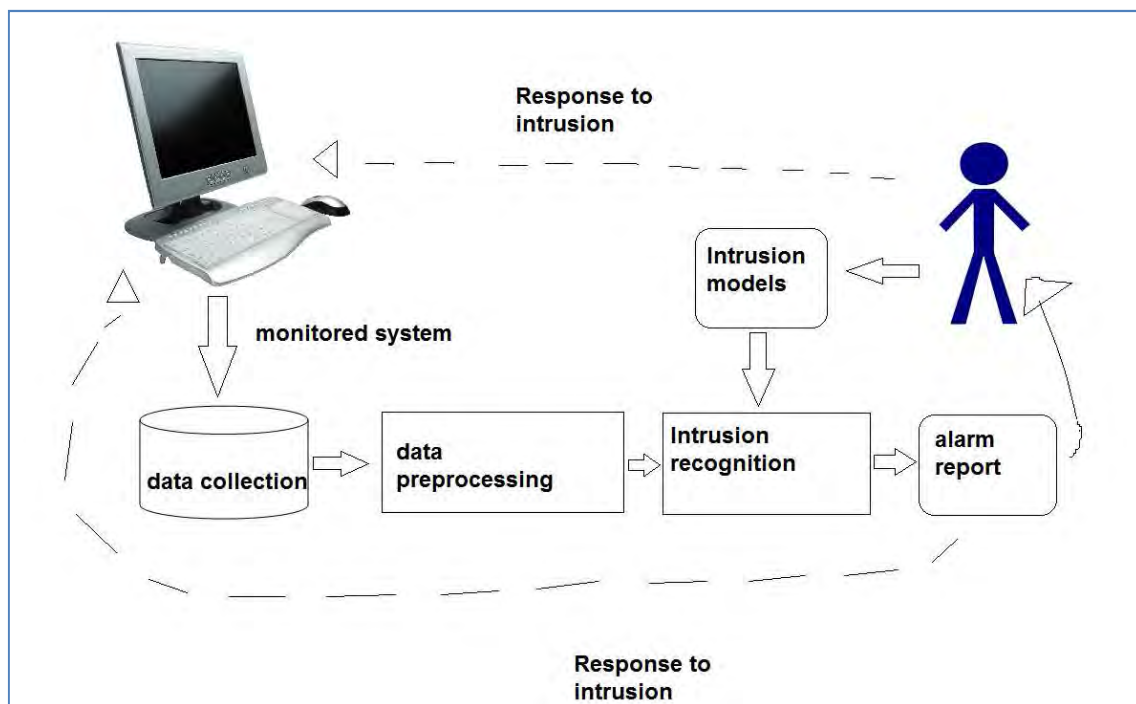


Figure 2.5 Intrusion Detection Systems; source (Elshoush&Osman, 2011).

2.4.6 Attacks classification

Generally, the type of attacks appears in the network are classified into four main groups, (Kezih&Taibi, 2013).

- **Denial of Service Attack (DOS):** in this kind of attacks the attacker will carry out some computation or memory resource unavailable in order to prevent valid requests, or reject valid users from using the service.
- **User to Root Attack (U2R):** The attackers have account on the system, and they will try to tap some weakness to get super user privileges within the system.
- **Remote to Local Attack (R2L):** in this group of attacks, the attacker having an ability to send packets into targeted devices, but the attacker here does not own an account on that device, and they try to obtain it.

- **Probing Attack:** this attack is a trial to gather any information about network of computers.

Table 3.1 elucidating different types of sub attacks that belong to the main attacks above along with their popular name (Kezih&Taibi, 2013).

Attack name	Attack type	Attack name	Attack type
Back	DOS	Per1	U2R
Buffer_ overflow	U2R	Phf	U2L
Ftp_ write	R2L	Pod	DOS
Guess_ passwd	R2L	PortswEEP	Prob
Imap	R2L	Rootkit	U2R
Ipsweep	Prob	Satan	Prob2
Land	DOS	Smurf	DOS
Loadmodule	U2R	Spy	R2L
Multihop	R2L	WareZclient	R2L
Neptune	DOS	WareZmaster	R2L
Nmap	Prob		

Table 2.1 types of sub attacks occur within a network, source (Kezih&Taibi, 2013).

2.4.7 Intrusion Detection Systems Evaluation

(Chandrasekhar & Raghuveer, 2013) described that only four final potential outcomes are expected when evaluate (IDS), and were called the *confusion matrix*, including the true positive (TP), true negative (TN), false positive (FP) and false negative (FN). The following Table 3.2 illustrates the confusion matrix

Class	Predicted negative class (normal)	Predicted positive class(attack)
Actual negative class(normal)	True negative (T N)	False positive (FP)
Actual positive class (attack)	False negative (FN)	True positive (T P)

Table 2.2 confusion matrix, source Chandrasekhar & Raghuveer, 2013)

We notice from table 2.2 the following:

- **True positive (T P):** indicates the summation of attacks connection records that were observed.
- **True negative (T N):** indicates the summation of normal connection records that were observed.
- **False positive (F P):** indicates the normal connection records that were observed as attacks records.
- **False negative (F N):** indicates the attacks records that were observed as normal activity.

Due to the fact that the summation of of U2R and R2L attacks records in the training and testing data set is very low, the above definitions are not adequate as standard performance measure, it could be a biased, so the metrics bellow solve this problem which is independence of the size of data set samples, (Chandrasekhar & Raghuveer, 2013), (Elshoush&Osman, 2010), (Al-Rshdan, 2011):

1. **Detection rate (DR) or sensitively or or true positive rate (T P R):** can be calculating by dividing the number of observed attacks records by the sum true positive and false negative.

$$(\text{TP R}), (\text{DR}) \text{ or sensitivity} = \frac{TP}{TP+FN} \dots\dots\dots (2.1)$$

- 2. Accuracy rate(AR):** can be calculating by dividing the summation of observed records (normal or attacks) to the sum of true negative, true positive, false negative, false positive

$$\text{Accuracy (AC)} = \frac{TN+TP}{TN+TP+FN+FP} \dots\dots\dots (2.2)$$

- 3. Precision rate (PP V):** is the proportion of observed attacks to the sum of true and false positive.

$$\text{Precision} = \frac{TP}{TP+FP} \dots\dots\dots (2.3)$$

- 4. False positive rate (FP R):** is the proportion unobserved normal records to the sum of true negative and false positive.

$$(\text{FP R}) = \frac{FP}{TN+FP} = 1 - \text{specificity (TN R)} \dots\dots\dots (2.4)$$

- 5. False negative rate (FN R):** is the proportion unobserved attacks, to the sum of true positive and false negative.

$$(\text{FN R}) = \frac{FN}{TP+FN} = 1 - \text{sensitivity (DR)} \dots\dots\dots (2.5)$$

- 6. True Negative Rate (Specificity):** is the proportion of observed normal connection records to the sum of these normal connection records and false positive.

$$(\text{TN R}) = \frac{TN}{TN+FP} \dots\dots\dots (2.6)$$

2.5 Artificial Neural Networks

2.5.1 Overview

Recently, computational alterations have brought a major outgrowth to modern techniques, such as the artificial neural networks (ANNs). Throughout the previous years of technological progression, (ANNs) offered a wide range of solutions for business, industry, developing and designing smart systems became essential tool for enhanced services to the users. A good example is the application of an artificial life to solve the interrogatives that a linear system alone is unable to resolve.

2.5.2 Artificial Neural Networks definitions

According to (Al-Janabi&Saeed, 2011) an artificial neural networks(ANNs) technique stands for a numerical model designed to cope with data in an aim to mimicking the biological nervous system processing methodology, such as the brain performance in handling all sorts of information, while its structure is considered as the corner stone of this model. The system by itself is adaptive and consists of numerous consistent processing elements named *neurons* acting to solve particular or diverse troubles.

Poojitha et al, (2010) stated that (ANNs) are categorized into parallel and distributive processing systems that consist of numerous connected processors.

Debar et al, (1992) confirmed that a neural networks is consisting of a group of simple units named *neurons*, that realizes a weighted sum of several inputs , these neurons are interconnected according to a given topology.

2.5.3 Artificial Neurons

The basic unit of artificial neural network is an artificial neuron which is representing the work of a living neuron in a human brain, Figure 2.6 represents the usual schema of an artificial neuron operation, (Kukielka&Kotulski,2008)

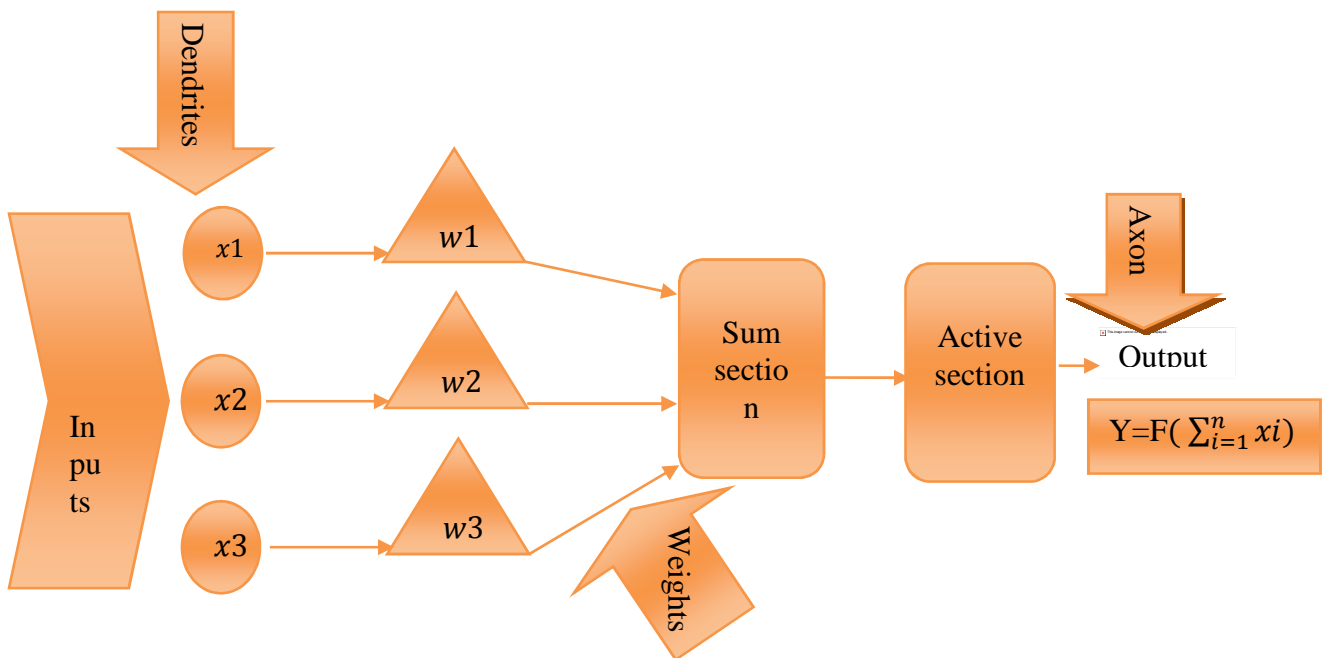


Figure 2.6: artificial neuron, source (Kukielka&Kotulski,2008)

Artificial neurons are consisting of a set of inputs that is imitating a biological neuron's *dendrites*, and a summation part, and an activation part, and a single output that simulates the biological neuron's *axon*. The input signals are multiplied by the weight values and subsequently the final result will be added in the summation section, which will be forwarded to the activation section to be processed by the activation function, and as a result, we will obtain a neurons output for every input 'X' signal

Based on *Naoum* (2013) artificial neural networks resembling the brain in two aspects:

- Gaining the knowledge via network throughout a learning process
- An interconnection benefits which called the synaptic weights are utilized for saving knowledge.

2.5.4 Feed forwards neural networks (FFNN)

The main (ANN) structure includes an *input*, *hidden*, and *output layers*. Neurons in adjacent layers are completely associated with weights, while within the same layer they are not linked to each other (GU et al, 2013). As shown in figure 2.7.

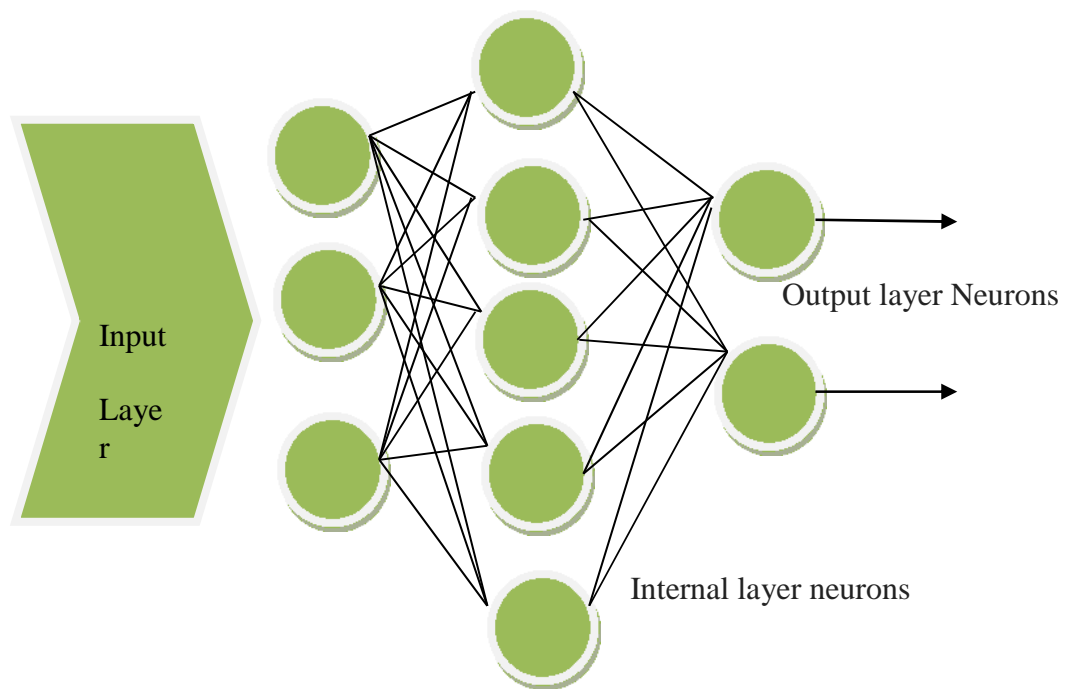


Figure 2.7: feed forward neural network, source (GU et al, 2013).

2.5.5 Learning process and supervised learning

Practically, learning represents improving or acquiring knowledge. The (ANN) learning is mainly divided into three types; unsupervised, supervised and reinforcement. This process happens in (ANN) based on modifying network attributes, (Sathya&Abraham,2013).

Sathya and Abraham (2013) stated that supervised learning supposes the accessibility of a supervisor. It mainly learns via controlling its interconnection weight groups with an aid from error signals. This process when occurring it's known as error back-propagation method. This method trains the network depending on the input/output samples. In addition, it discovers error signal that represents the difference among the computed and required outputs and controls the neurons synaptic weights.

Figure 15 below represents a supervised learning schema (Aboshosha,n d,2014)

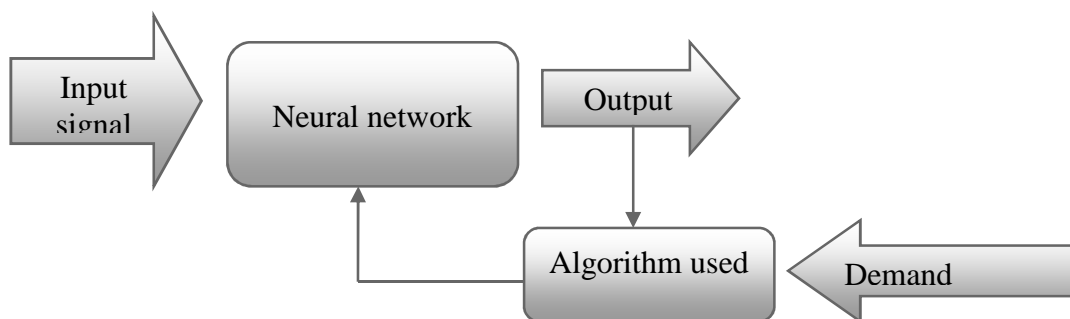


Figure 2.8: supervised learning, source (Sathya & Abraham, 2013)

Naoum (2013) mentioned that (ANN) learn from its fault and consists of three parts:

- Compute output.
- Compute output with answers.
- Modified the weight W_i and re processing.

The process start by putting random value for weight and compute the difference between the real outputs assume (y) and the demand output assume it (z) and change the weights till $\|z-y\|$ converge to zero at this stage we have $Z= y$, (nauoum ,2013). The training procedure aims to discover the weight values which can offer the output from (NN) to equal the real one .

2.6 Literature Review

An intrusion detection systems and computer networks traffic have been extensively studied and investigated by several researchers during the last years. A description of the previous work done in this field is explained briefly in the following paragraphs.

Riad et al (2013) evaluated the unsupervised learning technique that is employed for the detection of anomaly. K-means algorithm with the KDD Cup 1999 network dataset was used during the evaluation. There were 22 types of sub attacks have been founded in the dataset, and were classified into four major groups (*R2L Probe*, *U2R* and *Dos*). The final results showed a detection rate of 0.9766% for type (DoS) attacks, with false alarm rate of about 0.003%, on the other hand the detection rate of type (Prob) attacks was 0.0659 % with false alarm of 0.013%, and type (U2R) attacks detection rate was 0.00061 % with false alarm about 0.0004%, finally the detection rate of type (R2L) attacks was 0.381 % with false alarm about 0.0022%.

The number of features based traffic as well as the host-based traffic features as an input to the neural network for anomaly intrusion detection system was reduced by Wang and Ma (2009). The system performance was evaluated using *KDD99* by employing (Resilient Back Propagation) neural networks with different input numbers. (LogSig) was also used as transfer function, and their results showed that upon using 9 basic features as an input with architecture of (ANN) as 9-17-1 the detection rate was 85.7% with a training time of 126.753 seconds, but upon using 18 basic features+ time based as an input with architecture of (ANN) as 18-36-1, the detection rate was 88.4% with a training time of 202.562 seconds, yet upon using 22 basic features+ content as an input with architecture of ANN as 22-45-1 the detection rate was 85.3% with a training

time of 278.684 seconds, whereby using 28 basic features+ time based+ host-based as an input with architecture of (ANN) as 28-60-1 the detection rate was 86.1% with a training time of 369.542 seconds, while using 31 basic features+ content+ time based as an input with architecture of (ANN) as 31-65-1 yielded a detection rate of 86.9% with a training time of 409.919 seconds, and upon using 41 basic features+ content+ time based +host-based as an input with architecture of ANN as 41-85-1 the detection rate was 89.6% with a training time of 537.843 seconds. From all the above we conclude that the best result achieved when using an input number 18 to the neural network where the detection rate was 88% with a training time of 200 seconds and the mean square error about 0.000573.

In order to classify the type of attacks which were found in the KDD99 data set, *Mukhopadhyay et al.* (2011) designed an anomaly based IDS. They have learnt the NNs including 5000 samples (record) and later they took 41 attributes as an input to the NN. Experiment parameters were, the number of the input layers nodes which was 41, and the hidden layers number was 1 with 12 neurons, as well as a training algorithm that was based on back propagation the Results showed that the rate of detection was 95.6% with 4.4% rate of false alarm rate and the mean squares error of 0.0088598 at epoch 237.

Further, *Norouzzian et al.* (2011) was able to design a network intrusion detection system based on ANNs using “*Multi-Layer Perceptron (MLP)*”, in order to detects the attacks and classify them in 6 different groups (Smurf, Teardrop, Satan, Guest, Warezclient ,Buffer overflow). They employed the 10% of the KDD99 data set to evaluate the performance of their system, and also considered 13 features from each record, while (MLP) trained was trained using the technique of back propagation with three hidden layers. The results showed a detection rate of 90.78%.

Naoum et al. (2012) introduced a hybrid IDS using KNN and ERBP-NN. An optimal learning factor was derived to enhance the performance and speed up the convergence of the ERBP. First Norm was used in the k-nearest neighbour was implementation instead of Euclidean distance, in fact they have used the first nearest neighbor (k) equals 1. The enhanced resilient back propagation neural network was trained using an optimal neurons hidden layers number; therefore it was trained only with one hidden layer and 34 hidden neurons. The evaluation was performed on the NSL-KDD99 anomaly intrusion detection dataset. The proposed system has 97.2% rate of classification rate (5 classes) and 1% rate of false negative.

Hoque et al (2012) dedicated that maintaining high level of security is very essential in order to ensure that the information is being exchanged between the organizations trusty and safely. Achieving the security within internet environment is a critical issue due to the fact that these networks may be supposed to attack or any intrusion threat. So, there is increasing needs for applying IDSs in order to obtain better level of security via detecting intrusions and miss use. The researchers have been comprehensively studied and investigated the IDSs; no one of the designed and implemented IDS can be considered perfect. During their investigation, GA was employed in order to attain efficient intrusion detection approach. Two main phases were included within the proposed IDS, which are; pre-calculation stage and then applying the stage of detection. KDD was employed as a data set during the implementation and evaluation. The equation of standard deviation was also employed to measure the chromosome fitness. The results confirmed the effectiveness of the proposed IDS system in terms of the rates of false positive with value of 0.3046.

Ojugo et al (2012) also confirmed the importance of the data and network security in internet environment. The attacks detection or the users' intrusion for the network

were examined and investigated during their study. They dedicated that there are several conventional methods, such as; cryptography and password, can be employed by the users to achieve the security for the data. IDS approach that is recognized as a device, software or driver that can be employed to permit only the legal and secure access for the network system was used as a base for their implementation evaluation. Several IDSs are not able to detect new attacks because the implementation of these systems was performed through the systems of rule-based type. They introduced an IDS system based-GA and used classification rule set that has been derived based on the data audition of network in addition to the support confidence structure. Furthermore; an efficient, simplex and flexible fitness function was employed to estimate each rule's goodness. The aim from the implemented system is to improve the security of network system such that the integrity, availability and confidentiality for the resources of the system are allowed. The results confirmed the effectiveness of the created rule in detecting and classifying the network's intrusion.

Cleetus and Dhanya (2014) dedicated that the IDS are mainly used to prevent the outside and inside attacks for the system. The Evolutionary algorithm plays an essential role in the process of detecting the intrusion within the system; since it responses highly to the space reduction of features. The IDS performance can be enhanced through using minimal features number. A novel IDS was proposed during their study; this system includes several methods for feature selection, such as; features cardinality, mutual correlation and information gain. GA was applied into changeable subset of features in order to perform the detection of anomaly within the system. The results illustrated that the rate of detection can be improved by applying the proposed IDS method. The obtained model accuracy was 87.54% achieved at one hundred iterations considering 350 populations.

A novel IDS that can detect several network intrusion types was proposed by Benaicha et al (2014). The system based on using the GA considering enhanced selection operator and initial population in order to effectively perform the detection. GA was employed in order to enhance the search process about the scenario of the attack within the audit files as much as possible. This algorithm affords the presented potential attack subsets within the audit files with reasonable time of processing. NSL-KDD99 data set was employed during the evaluation of system performance for misuse activities detection. The results illustrated that combining the GA with IDS will result in enhancing the performance by increasing the rate to 99% of detection and reducing the rate of false positive to 3%.

Sandhya and Julian (2014) dedicated that the communication system security is considered critical and essential issue. The attacks' recognition and preventing with network is not easy task; so the IDS are needed in order to ensure the security of the network. A technique for pattern analysis is usually included and employed in IDS in order to find out the useful pattern features of the system. These discovered patterns are employed in determining the behavior of the user. The computation of anomalies is performed using the relevant features of the system. The unknown patterns attack can be effectively detected using clustering techniques, such as; k-mean algorithm.

Furthermore; insignificant features elimination is also effective and importance for simplification and obtaining more precise and faster attacks' detection. So; combining the GA and k-mean algorithm for clustering will provide the recognition for reduced and only the significant input features. The system performance was evaluated in terms of the rate of detection and the rate of false positive, the performance was compared to other existing algorithm for intrusion detection. The results confirmed the ability of the proposed genetic-k-mean IDS for detecting the intrusion with rate of 90% and rate of

false positive with value less than 4%. They concluded that the proposed IDS is effective in intelligently detect and analyzing the intrusion and it can be also employed for dynamic environment.

Chapter three
Proposed Intrusion Detection System
Algorithms Implementation

Chapter Three

Proposed Intrusion Detection System Algorithms Implementation

3.1 preface

An Intrusion Detection Systems (IDS) is a network security technology originally built for detecting vulnerability exploits against a target application or computer. IDS can build based on different techniques; the proposed IDS based on three different approaches:

- The first algorithm is K-means clustering technique to build the intrusion detection system.
- The second algorithm is Novel K-NN clustering technique.
- The third algorithm is Recursive Least Squares Multilayer Back Propagation Neural Network.

Each of the above mentioned techniques will be detailed in the following sections as a separate unit.

The common building block in these intrusion detection systems is the pre-processing unit, which will prepare the training and testing data in appropriate format to be used in the three algorithms mentioned above. Our proposed system consists of two main phases:

- The former is the data pre-processing phase.
- The latter is the phase of IDS implementation.

3.2 Data Pre-Processing phase

The data pre-processing phase is processed in three main phases:

- 1) Training and Testing Data Sets Collection stage,
- 2) Data Codification stage, and,
- 3) Data Normalization stage.

The above mentioned stages are best illustrated in figure (3.1).

3.2.1 Training and Testing Data Sets Collection stage

The first stage of building any intrusion detection system is to collect the data that will be used in both, training and testing phases. The databases are subsets of NSL-KDD'99 dataset. This dataset is available for researchers whose work in the field of intrusion detection or prevention via <http://nsl.cs.unb.ca/NSL-KDD> sit. This dataset proposed in order to solve the problems of the KDD'99 data set (Tavallae and et.al, 2009).

NSL-KDD can be considered as an update version of the KDD data set, where it has many pros over the KDD data set where it doesn't contain any record redundancy in the training set, so when we use the classifiers (K-means, Novel K-NN, and RLS ANN) it will not be shifted towards records that are more duplicated. In addition, there is no records duplication in the testing set, so the action of the classifiers ((K-means, Novel K-NN, and RLS ANN) will not be shifted by the techniques which have higher detection rates on the duplicated t records.

NSL-KDD can be applied to this research to compare different intrusion detection techniques since it can be considered an effective benchmark dataset.

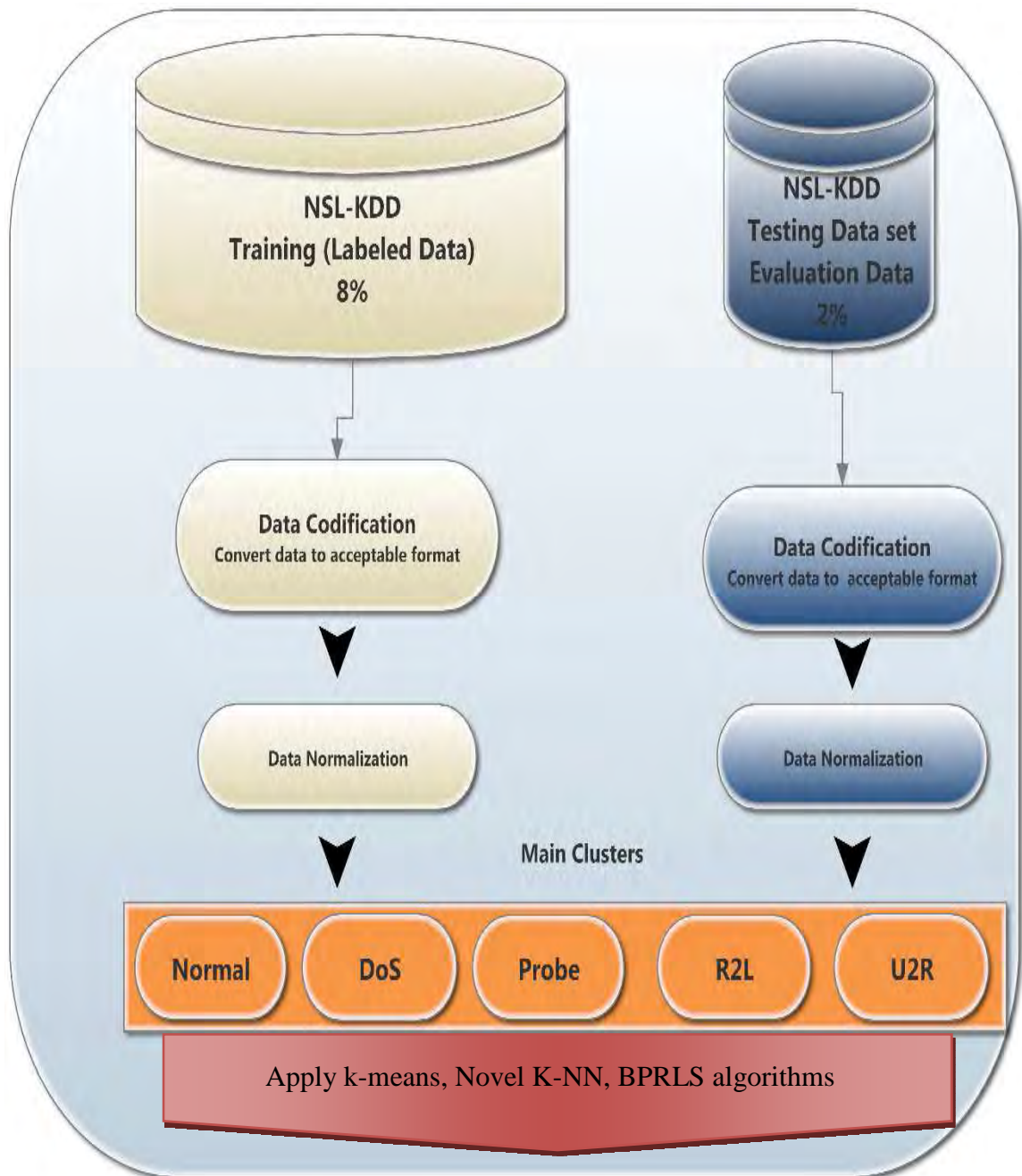


Figure 3.1: The stages of Data pre-processing phase

3.2.2 Data Codification Stage

Coding is a principal and sometime can be considered the most difficult stage in the analysing qualitative data; this subsection has been dedicated to the coding phase of the NSL_KDD dataset. Several record examples provided to elaborate how coding process is done.

Coding is defined as the process in which the raw qualitative data is examined in away assign codes or labels to any piece of data that come in the form of words, phrases, sentences or paragraphs.

NSL_KDD connection records contain fields (features) that need to be transformed to numerical values in order to be in appropriate format suitable to our classification techniques. Figure (3.2) shows a small set of connection records of the raw NSL_KDD dataset where the fields (attributes) that needed to be coded is represented in black.

```
0,tcp,http,SF,288,300,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,4,4,0.25,0.25,0.00,0.00,1.0
0,0.00,0.00,4,255,1.00,0.00,0.25,0.03,0.25,0.01,0.00,0.00,normal,19

0,tcp,private,REJ,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,121,19,0.00,0.00,1.00,1.00,
0.16,0.06,0.00,255,19,0.07,0.07,0.00,0.00,0.00,0.00,1.00,1.00,neptune,21

6,tcp,gopher,SF,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,
0.00,0.00,51,2,0.02,1.00,0.02,1.00,0.00,0.00,0.92,0.50,ipsweep,9

0,icmp,eco_i,SF,20,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,65,0.00,0.00,0.00,0.00,1.0
0,0.00,1.00,3,57,1.00,0.00,1.00,0.28,0.00,0.00,0.00,0.00,saint,15

0,tcp,ftp_data,SF,0,5828,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,
1.00,0.00,0.00,2,2,1.00,0.00,1.00,0.00,0.00,0.00,0.00,0.00,buffer_overflow,12
```

Figure 3.2: Small set of NSL-KDD dataset connection records source (mat lab program)

As figure (3.2) shows; the second, third, fourth and 42th attribute need to be coded into numerical values, so we obtain a full numerical record.

Second attribute represents the type of the connection protocol, which can be one of three protocols (**TCP**, **UDP**, **ICMP**), table (3.1) shows the numerical numbers that used to code this feature and figure (3.3) shows the MATLAB code of this coding.

Table (3.1) The numerical representation of 2th feature (protocol).

Protocol Type	Numerical Representation
TCP	2
UDP	3
ICMP	4

```

=====
% Processing column 2 of NSL-KDD Dataset: Protocol
=====

for i=1:length(Data(:,2))
    if(strcmp(Data{i,2},'tcp'))
        Data{i,2}=2;
    elseif(strcmp(Data{i,2},'udp'))
        Data{i,2}=3;
    end
    if(strcmp(Data{i,2},'icmp'))
        Data{i,2}=4;
    end
end
end

```

Figure 3.3: MATLAB implementation of protocol feature coding

The third attribute represent the destination service, it can be one of variety of available destination services. In coding process of this attribute, the services in connection records must be collected first and comparing the training and testing dataset records to this collected set , then assign a numerical value to each one .This is best illustrated via table (3.2) and figure(3.4).

Table (3.2) The numerical representation of 3th feature (Service).

Service	Numerical Representation	Service	Numerical Representation
'http'	16	'login'	60
'private'	17	'kshell'	61
'gopher'	18	'sql_net'	62
'telnet'	19	'time'	63
'ftp_data'	20	'hostnames'	64
'other'	21	'exec'	65
'remote_job'	22	'ntp_u'	66
'eco_i'	23	'nntp'	67
'smtp'	24	'ctf'	68
'ftp'	25	'daytime'	69
'ldap'	26	'shell'	70
'pop_3'	27	'IRC'	71
'courier'	28	'pop_2'	72
'discard'	29	'printer'	73
'ecr_i'	30	'tim_i'	74
'imap4'	31	'pm_dump'	75
'domain_u'	32	'red_i'	76
'mtp'	33	'netbios_ssn'	77
'sysstat'	34	'rje'	78
'iso_tsap'	35	'X11'	79
'csnet_ns'	36		
'finger'	37		
'uucp'	38		
'whois'	39		
'nnsf'	40		
'netbios_ns'	41		
'domain'	42		
'ssh'	43		
'netstat'	44		
'name'	45		
'supdup'	46		
'uucp_path'	47		
'Z39_50'	48		
'netbios_dgm'	49		
'urp_i'	50		
'auth'	51		
'bgp'	52		
'vmnet'	53		
'http_443'	54		
'efs'	55		
'echo'	56		
'klogin'	57		
'link'	58		
'sunrpc'	59		

```

=====
% Processing column 3 of NSL-KDD Dataset : Service.
=====

service_array = {'aol'; 'http_443'; 'http_8001'; 'http_2784';...
                'domain_u'; 'ftp_data'; 'auth'; 'bgp'; 'courier';...
                'tftp_u'; 'uucp_path'; 'csnet_ns'; 'ctf';...
                'daytime'; 'time'; 'discard'; 'domain'; 'echo';...
                'eco_i'; 'ecr_i'; 'efs'; 'exec'; 'finger'; 'gopher';...
                'harvest'; 'hostnames'; 'http'; 'imap4'; 'IRC';...
                'iso_tsap'; 'klogin'; 'kshell'; 'ldap'; 'link';...
                'login'; 'smtp'; 'mtp'; 'name';...
                'netbios_dgm'; 'netbios_ns'; 'netbios_ssn'; 'netstat';...
                'nntp'; 'nntp_u'; 'ntp_u'; 'other'; 'pm_dump'; 'pop_2';...
                'pop_3'; 'printer'; 'private'; 'red_i'; 'remote_job'; ...
                'rje'; 'shell'; 'sql_net'; 'ssh'; 'sunrpc';...
                'supdup'; 'systat'; 'telnet'; 'tim_i';...
                'urh_i'; 'urp_i'; 'uucp'; 'ftp'; 'vmnet';...
                'whois'; 'X11'; 'Z39_50'};

L = length(service_array);
arr = cell(L,1); % these are the 66 services that exist in the third
column of data base
Service = cell(L,2);
arr(1,:) = Data(1,3); % this is the first value of the Service column of
the NSLKDD database .
p=2;

for i=2:length(Data(:,3)) % Looping on the whole records in the
third(service ) column
    flag=true;
    for j=1:length(arr)
        if(strcmp(Data(i,3),arr(j)))
            flag = false;
            break;
        end
    end

    if(flag==true)
        arr(p,:) = Data(i,3);
        p = p+1;
    end
end

for i=1:length(Data(:,3))
    for j=1:length(arr)

        if(strcmp(Data{i,3},arr(j) ))
            Service(i,1)= Data{i,3};
            Data{i,3} = j+15;
            Service(i,2) = Data{i,3};
        end
    end
end
end

```

Figure 3.4: MATLAB implementation of service feature coding.

The fourth attribute of the connection records of NSL-KDD is status flag of the established connection, or flag feature. The coding process of this attribute exactly in

the same manner in case of service attributes coding and this is shown in table (3.3) and figure 3.5.

The 42th attribute of connection records represent the label of the record, or it classify the connection record whether it is normal or attack one. The label of record can be either “normal” or “attack”.

To prepare the actual labels for propose of being the targets of the ANN later on, the first step is to transform the attack labels to its major type and then substitute number indicates the value of attack type as shown in table (3.4) and figure(3.6).

Table (3.3) Numerical representation of 4th feature (flag).

Flag	Numerical Representation
'SF'	4
'REJ'	5
'S0'	6
'RSTO'	7
'RSTR'	8
'SH'	9
'S3'	10
'S1'	11
'RSTOS0'	12
'S2'	13
'OTH'	14

Table (3.4) Numerical representation of 42th feature (record label).

Record Label	Numerical Transformation
Normal	1
DoS	2
Prob	3
R2L	4
U2R	5


```

=====
% Processing coloumn4 of NSL-KDD Dataset : Flag.
=====
flag =
{'OTH'; 'REJ'; 'RSTO'; 'RSTOS0'; 'RSTR'; 'RSTRH'; 'S0'; 'S1'; 'S2'; 'S3'; 'SF'; 'SH'; 'SHR'
''};
L = length(flag);
arr = cell(L,1);
Flag = cell(L,2);
arr(1,:) = Data(1,4);
p=2;
for i=2:length(Data(:,4))
    flag=true;
    for j=1:length(arr)
        if(strcmp(Data(i,4),arr(j)))
            flag = false;
            break;
        end
    end
    if(flag==true)
        arr(p,:) = Data(i,4);
        p=p+1;
    end
end

% transform the flag string into numeric values and collect them in one
% array
for i=1:length(Data(:,4))
    for j=1:length(arr)
        if(strcmp(Data(i,4),arr(j)))
            Flag(i,1)= Data(i,4);
            Data(i,4) = j+3;
            Flag(i,2)= Data(i,4);
        end
    end
end
end

```

Figure 3.5: MATLAB implementation of flag feature coding.

Figure (3.6) shows the record label coding operation in MATLAB. The deleted 43th feature column since it is not a part of the dataset but added by the creators of NSL-KDD for evaluation purposes, this attribute is called the difficulty level, and it does not consider as a part of the dataset. This removal via MATLAB is as following:

```

% Remove the coloumn of 43
Data(:,43)= [];

```

```

%=====
% Processing column 42 (Attacks Labels )column of NSL-KDD Dataset and take the
Labels
%=====
DoS
=['back';'land';'neptune';'pod';'smurf';'teardrop';'mailbomb';'processtable';'udpstorm';'apache2';'worm'];
Probe =['satan';'ipsweep';'nmap';'portsweep';'mscan';'saint'];
R2L
=['guess_password';'guess_passwd';'ftp_write';'imap';'phf';'multihop';'warezmaster';'xlock';'xsnoop';'snmpguess';'snmpgetattack';'httptunnel';'sendmail';'named';'warezclient';'spy'];
U2R =['buffer_overflow';'loadmodule';'rootkit';'perl';'sqlattack';'xterm';'ps'];

for i=1:length(Data(:,42)) % Begin the Processing

    if(strcmp(Data{i,42},'normal'))

        Normals = Normals + 1 ;
    end

    for j=1:length(DoS)
        if(strcmp(Data{i,42},DoS(j)))
            Data{i,42}='DoS';
            DoSs = DoSs + 1 ;
        end
    end

    for j=1:length(Probe)
        if(strcmp(Data{i,42},Probe(j)))
            Data{i,42}='Probe';
            Probes = Probes +1 ;
        end
    end

    for j=1:length(R2L)
        if(strcmp(Data{i,42},R2L(j)))
            Data{i,42}='R2L';
            R2Ls = R2Ls +1;
        end
    end

    for j=1:length(U2R)
        if(strcmp(Data{i,42},U2R(j)))
            Data{i,42}='U2R';
            U2Rs = U2Rs +1;
        end
    end

end % End of Process

```

Figure 3.6: MATLAB implementation of label feature coding.

For the rest of other attributes, from the 5th field up to the 41th feature is listed in the table (3.5) .Since these attributes have numerical values, and then it will not undergo a codification process.

Table (3.5) Full description of features of a connection records of NSL-KDD, source

No.	Feature	Description	No.	Feature	Description
1	Duration	duration of the connection	22	Is guest login	1 if the login is a guest login,0 otherwise
2	Protocol type	Connection otocol(e.g.tcp,udp)	23	Count	Number of connections to the same host as the current connection in the past two second
3	Service	Destination service (eg.ftp,telnet)	24	Srv count	Number of connections to the same service as the current connection in the past tow second
4	Flag	Status flag of the connection	25	Error rate	% of connections that have 'syn' error
5	Source bytes	Bytes send from source to destination	26	Srv error rate	%of connection that have 'syn' errors
6	Destination bytes	Bytes sent from destination to source	27	Rerror rate	%of connection that have 'rej'errors
7	Land	1 if connection is from/to the same host/port,0 otherwise	28	Srv error rate	%of connections that have 'rej'error
8	Wrong fragment	Number of wrong fragment	29	Same srv rate	%of connections to the same service
9	Urgent	Number of urgent packets	30	Diff srv rate	%of connections to different service
10	Hot	Number of hot indicators	31	Srv diff host	%of connection to different host
11	Failed logins	Number of failed logins	32	Dst host count	Count of connections having the same destination host
12	Logged in	1 if successfully logged in,0 otherwise	33	Dst host srv count	Count of connections having the same destination host and using the same service
13	# compromised	Number of compromised condition	34	Dst host same srv rate	%of connections having the same destination host and using the same service
14	Root shell	1 if root shell is obtained,0 other wise	35	Dst host diff srv rate	%of different services on the current host
15	Su attempted	1 if 'su root ' command attempt ,0 otherwise	36	Dst host same port rate	%of connections to the current host having the same scr port
16	#root	Number of root accesses	37	Dst host srv diff host rate	%of connections to the same service coming from different host
17	#file	Number of file creation operation	38	Dst host error rate	%of connections to the current host that have an s0 error
18	#shell	Number of shell prompt	39	Dst host srv error rate	%of connections to the current host and specified service that have an s0 error
19	# access file	Number of operation on access control files	40	Dst host error	%of connections to the current host that have rst error
20	#outbound cmds	Number of outbound commands in an ftp session	41	Dst host srv error	%of connections to the current host and specified service that have an rst error
21	Is hot login	1 if the login belongs to the hot list,0 otherwise	42	Difficulty level	Measure the difficulty level

source, (kayaci et al, 2005).

3.2.3 Data Normalization Stage

The final stage of the data pre-processing is the data normalization. In this stage, the codified data sets (both Training dataset and testing dataset) undergo normalization operation, which could be one of two types: zscore normalization or min-max normalization that explained in details in the chapter two sections 3.2. In the proposed system implementation zscore normalization approach are adapted, since it proved it's efficient in feature classification which enhances the operation of the proposed intrusion detection systems .

Our MATLAB implementation is explained below in figure (3.7).

```

%=====
% Do Normalization
%=====
if(option == 1)
    Normalized_Data = zscoreNormalization(Data);
else
    Normalized_Data = minmaxNormalization(Data);
end
%=====
% Calling zscore normalization function
%=====
function [Normalized_Data] = zscoreNormalization(Data)

    mu = zeros(1, size(Data, 2));
    sigma = zeros(1, size(Data, 2));

    mu = mean (Data);
    for i=1:size(Data, 2)
        Data(:,i) = Data(:,i) - mu(i);
    end;

    sigma = std (Data)+0.5;
    for i=1:size(Data, 2)
        Data(:,i) = Data(:,i) ./ sigma(i);
    end;

    Normalized_Data = Data;
end

```

Figure 3.7: MATLAB implementation of zscore normalization.

3.3 Intrusion Detection Systems Implementation Phase

In this section, three different intrusion detection system algorithms are implemented, which are K-means IDS, Novel K-NN IDS and Recursive Least Squares ANN IDS algorithms.

3.3.1 K-Means Intrusion Detection System

K-Means, suggested by James McQueen, is considered one of the clustering techniques in machine learning, which can be used to recognize groups of similar records in the data set.

The algorithm classifies the connection records to a pre-defined number of clusters that is pre-determined by the user, and in our case is five clusters, one cluster for each class (**Normal, DoS, Probe, U2R and R2L**), so $k = 5$ or (**5 Means**) in our proposed system.

Randomly, the first step is to select the centres of the clusters. Then, the algorithm continues to read each record from the dataset and assigns it to the nearest cluster.

The Euclidian distance is the method that used to measure the distance between record and the centre and it is considered the most popular one. However, there are many other methods and can be suggested for further performance improvement.

The cluster centres' are always recomputed after every records insertion, and this process is iterated until our algorithm reach to saturation, which means that no more changes are made to the centres'. The proposed K-Means Intrusion Detection System is explained in the following subsection as pseudo code where each one is followed by its MATLAB implementation.

3.3.1.1 K-means IDS Pseudo Code and MATLAB Implementation

In this subsection, K-Means IDS algorithm will be explain that implemented in MATLAB.

Figure 3.8 depicts the basic steps of k-means algorithm with intrusion detection and explains in steps bellow it with mat lab implementation.

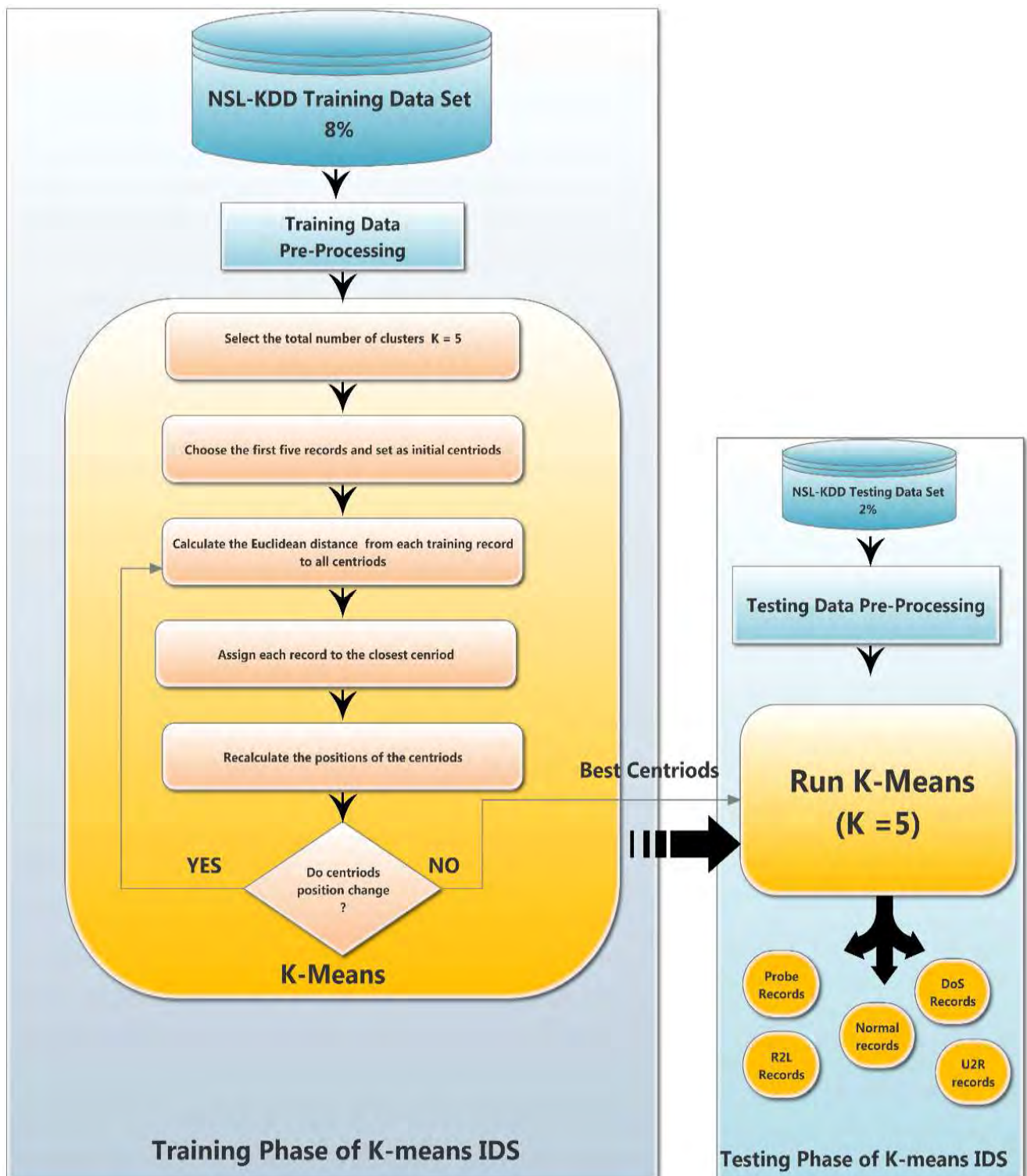


Figure 3.8: Proposed K-Means IDS block

1. Load Training Data.(8% of Database)
2. Run Data Pre-Processing function.
3. Select the total number of clusters, and our case, the total number of cluster is five due to fact that five labels (Normal, DoS, Probe, U2R, and R2L) were exists.

```
k =5; % five clusters corresponding to "normal " and four attacks.
```

4. Choose the first five records and set it as initial centers.

```
function [class,centroid,counter]=MyKMeans(data,k,option,init_centroid)

% Option1- First Five records
% Option2- Centroid passes as parameter

centroid=[];% initialize the centriods matrix
class=[];
[n,m]=size(data);

%get the initial centroids
if (option==1)
    centroid(1:5,:)=data(1:5,:);% Option1- in training phase

elseif (option==2)
    centroid = init_centroid; %Option2- in Testing phase

end

flag=0;
count=0;
counter=[];
```


5. Compute the Euclidean distance from each record to all centers and assign each record to the closest center.

```
%Calculate the distance and assign the class
function [dist,classify]=MyDistance(centroid,data)

[n,m]=size(data);
[k,l]=size(centroid);
dist=[];
classify=[];

for i=1:k
    for j=1:n
        %calculate the distance from each centroid to the data
        sum=0;
        for p=1:m
            sum=sum+(data(j,p)-centroid(i,p))^2;
        end
        dist(i,j)=sum^0.5;
    end
end

for j=1:n
    [minv,minindex]=min(dist(:,j));
    classify(j,1)=minindex;
end
```

6. Recomputed the location of the centers.

```
%Calculate new Centroid
function [newCentroid,counter]=CalculateCentroid(centroid,class,data)

[n,m]=size(data);
[k,l]=size(centroid);
newCentroid=zeros(k,l);
counter=zeros(k,l);
for j=1:k
    for i=1:n
        if(class(i,1)==j)
            for p=1:m
                newCentroid(j,p)=newCentroid(j,p)+data(i,p);
            end
            counter(j,1)=counter(j,1)+1;
        end
    end
end
for j=1:k
    for p=1:m
        newCentroid(j,p)= newCentroid(j,p)/counter(j,1);
    end
end
```

7. Repeat steps (5) and (6) until the centers no long change.

```

flag=0;
count=0;
counter=[];
%classify the data

while(flag==0)
    [dist,class]=MyDistance(centroid,data);

    if(count~=0)
        temp=(class==prevclass);
        if( max(max(temp))==1 && min(min(temp))==1)
            flag=1;
            counter;
            break;
        end
    end

    prevclass=class;
    [centroid,counter]=CalculateCentroid(centroid,class,data);
    count=count+1;
end

```

8. Re-do the steps (2) and (3) for Testing data set.
9. Run K-Means algorithms, but using the second option, which means, using the best centroids that obtained in the training phase, and run the confusion matrix function to measure the algorithm performance.

```

k =5; % five clusters corresponging to "normal " and four subattacks .
%Option is one of the following:
%1- First Five records
%2- Centroid passes as parameter
Option = 2 ;%in the training phase , use option 1.
init_centroid = Centriods;%Centriods that you got in the training phase.
[testPredictedLabels,Centriods,testcounter]=
MyKMeans(Normalized_testData, k,Option,init_centroid);

%=====
% Confusion Matrix .%
%=====

stats = confusionmatStats(ActualLabels,testPredictedLabels);
disp(stats.confusionMat);

```

3.3.2 Novel KNN Intrusion Detection System

Novel KNN is a clustering algorithm similar to K-Means, which means it depends on a distance or similarity functions, such as Euclidean distance function. This classifier is first employed to classify the documents in (Jivani, 2013).

This algorithm transferred to implement the second proposed intrusion detection system to classify the connection records into normal and attacks and then further to classify the attacks into its major classes.

3.3.2.1 Novel KNN IDS Pseudo Code and MATLAB Implementation

In this subsection, Novel K-NN IDS algorithm will be explain and implemented in MATLAB.

Figure 3.9 depict the basic steps of Novel K-NN for intrusion detection and explains in steps bellow it with mat lab implementation.

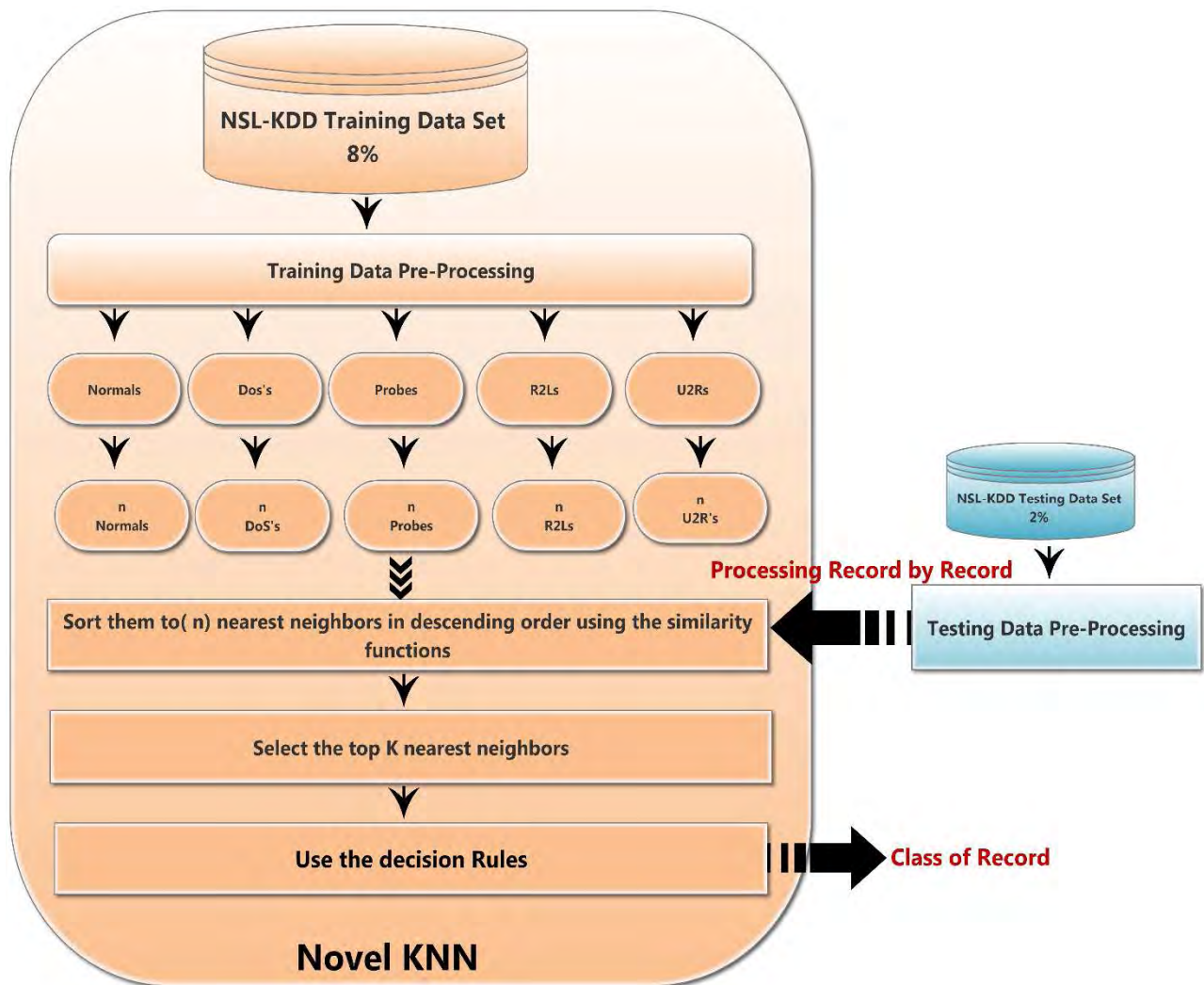


Figure 3.9: Proposed Novel K-NN IDS block diagram.

1. Load Training Data.(8% of Database)
2. Run Data Pre-Processing function and get the statistics of classes in the Training data set, so you can determine the smallest class (class with less frequency).

```
input_Data = 'NSLKDDTrain.txt';
%if option is 1 , then you choose zscore Normalization , otherwise
,minimax Normalization.
NormalizationOption = 1 ;

[trLabelsStatistics,trData ]=
DataPreprocessing(input_Data,NormalizationOption);

disp('The Statistics of Normals and subattacks in Training Data');
disp(trLabelsStatistics);
```

3. In this algorithm, the selection of (**k**) nearest connection records depends on a parameter (**n**) that is taken as input from the user and at the same time, it depends on the size of the smallest class.

```
n = input('Enter n nearest neighbors of test record ?');
```

4. Select the (**n**) nearest neighbors of connection record (**R_i**) from each class. The value of (**n**) should not be greater than the size of the smallest class. For example, in our implementation of this algorithm, the smallest class was the U2R class with size = **27**, thus (**n**) can have a value that is less than or equals to (**n**), namely, (**n** <= **27**).

```
% 1.First,select the n nearest neibors of record Ri,from each class.The
value
% value of n should not be greater than the size of the smallest class.
% For example,the smallest training class contains 27 records then , n
can
% have a value that is less than or equals to n;i.e.;n <= 27.

for R=1:size(testData,1)

    [NormalNeighbors] = kNearestNeighbors(Normals,testData,n);
    [DoSNeighbors]    = kNearestNeighbors(DoSs,testData,n);
    [ProbeNeighbors]  = kNearestNeighbors(Probes,testData,n);
    [R2LNeighbors]    = kNearestNeighbors(R2Ls,testData,n);
    [U2RNeighbors]    = kNearestNeighbors(U2Rs,testData,n);
```

```

function Knearestneighbors = kNearestNeighbors(dataMatrix, queryVector,
k)

numDataVectors = size(dataMatrix,1);

dist = sqrt( sum(( repmat(queryVector(1,:),numDataVectors,1)-
dataMatrix).^2,2));
[sortval sortpos] = sort(dist,'descend');
neighborIds      = sortpos(1:k);
Knearestneighbors = dataMatrix(neighborIds,: );

end

```

5. Sort these (**n**) nearest neighbors in descending order of the similarity. We use the following cosine similarity function :

$$\text{Sim}(r_1, r_2) = \frac{\vec{v}(r_1) \cdot \vec{v}(r_2)}{\|\vec{v}(r_1)\| \cdot \|\vec{v}(r_2)\|}$$

```

% 2.Sort these n nearest neighbors in descending order of the similarity
Neighbors_list =
[NormalNeighbors;DoSNeighbors;ProbeNeighbors;R2LNeighbors;U2RNeighbors];
sim = zeros();%For initialization .
for i= 1: (5*n)
    sim(i) =(
dot(testData(R,:),Neighbors_list(i,:))./(norm(testData(R,:))*norm(Neighbors_list(i,:)));.....(eq.3)
end
sim = sim';%to see it as a vector .
[val pos] = sort(sim,'descend');

```

6. Now, select the top (k) nearest neighbors from the list prepared in previous step and these are considered the final (k) nearest neighbors of the record R_j

```

% 3.Select now the top k nearest neighbors from the list prepared in
% step2 .These are the final k nearest neighbors of the records Ri now.
k = 15;

NeighborsID = pos(1:k);
Final_KNeighbors = Neighbors_list(NeighborsID,:);

```

7. Use the decision rules given by the following classification formulas:

$$y(R_j) = \underset{k}{\operatorname{argmax}} \sum_{x_j \in KNN} y(x_j, C_j)$$

Where, R_j is the record to be classified, x_j is one of the neighbors of R_i and $y(x_j, C_j)$ belongs to $\{0, 1\}$ which indicates whether the connection record x_j belongs to class C_k or not.

Alternatively, you can use the following classification formula:

$$y(R_j) = \underset{k}{\operatorname{argmax}} \sum_{x_j \in KNN} y(R_j, x_j) \operatorname{sim}(x_j, C_k)$$

Where, $\operatorname{sim}(x_j, C_k)$ is similarity measure between the test connection record R_j and its neighboring record x_j and this is generally will be the cosine similarity.

Then find the class to which R_j is most similar to and would belong.

```
% 4.Using the decision rules given,now find the class to
% which Ri is most similar to and would belong.
KNeighbors_labels = Final_KNeighbors(:,42);
PredictedLabels(R)= mode(KNeighbors_labels);%voting process.
```

3.3.3 Multilayer Back Propagation Recursive Least Square based IDS

In this section, BPRLS IDS will be elaborate, the first step is to present the whole framework of the novel approach depicted in figure (3.10). Then discuss the training and testing phases of the proposed model associated with its MATLAB implementation.

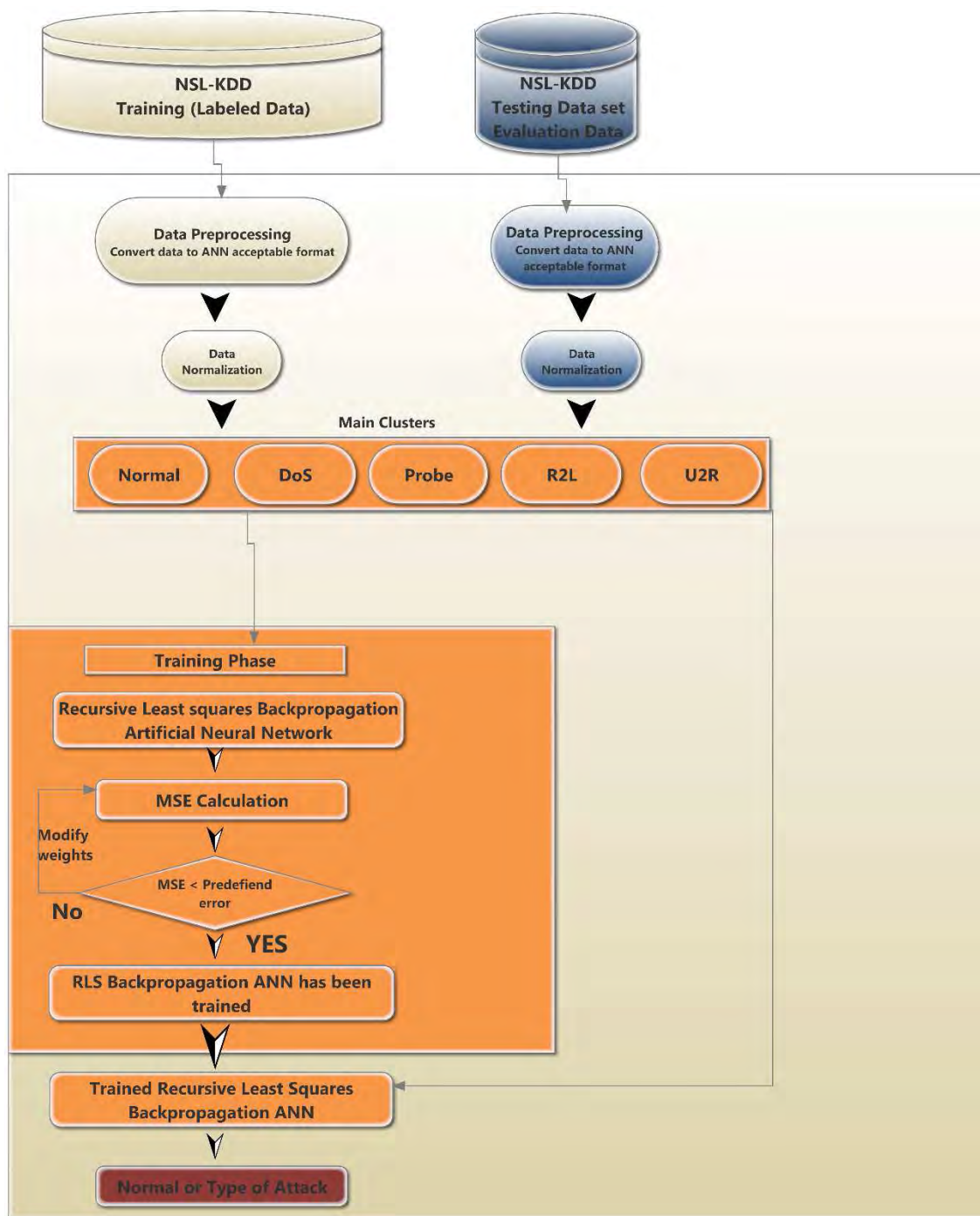


Figure 3.10: proposed BPRLS IDS Model block diagram.

3.3.4 Proposed BPRLS IDS Training Phase and MATLAB Implementation

In this section the training of recursive least squares back propagation neural network algorithm (Scalero et al.1992) that consists the heart of the proposed system. In this section, BPRLS algorithm will be used to build IDS. First, the BPRLS must be trained via the training dataset until it reaches to optimal MSE, which in turn leads to optimal weights. Then the optimal weights that obtained in training phase will be used in the testing phase of BPRLS network.

1. Load Training Dataset.(5% of NSL-KDD Database)
2. Run Data Pre-Processing function
3. Run the BPRLS algorithm training as explained in the following steps :

Step 1: Initialization phase: in this phase, the weight will be initialized to random values and the correlation matrix \mathbf{R} is initialized too. We initialize the weights randomly between -0.000005 and

+ 0.000005, and we initialize \mathbf{R} matrix to $\zeta^{-1}\mathbf{I}$, where \mathbf{I} is the identity matrix and ζ is a small number ($0.001 < \zeta < 0.01$)

```
%initialize the weights,Randomize all weights in the network
weights = cell(layersLength-1,1); % a weight matrix between each
layer .
for i=1:layersLength-2
weights{i} = [0.000005 + 0.000005.* rand(layers(i+1),layers(i)+1);
zeros(1,layers(i)+1)];
end
weights{end}= 0.000005 + 0.000005.* rand(layers(end),layers(end-
1)+1);

R = cell(layersLength-1,1);
R_inv = cell(layersLength-1,1);
for i=1:layersLength-1
x{i} = ones(layers(i)+1,1); % inner layers include a bias node
(trainingDataCount-by-Nodes+1
R{i} = 0.1*eye(length(x{i}));
R_inv{i} = inv(R{i});
end
```

Step 2: Select a training pattern randomly. The input is the connection record x and the output is the label of the record (attack type) denoted as O .

Step3: Run the selected input/desired output pattern through the network and evaluate the summation output for each layer from the hidden layer j through L

$$y_{jk} = \sum_{i=0}^N (x_{j-1}, w_{jki})$$

The output of activation function is:

$$x_{jk} = f(y_{jk}) = \frac{1 - \exp(-ay_{jk})}{1 + \exp(-ay_{jk})}$$

N : is the number of node inputs except the offsets (bias), a : slope of sigmoid function.

```
%=====
%Step #3 : Run selected pattern through the network .
%=====

%Compute the inputs and outputs to each layer
for i=1:layersLength-1
y{i} = weights{i}*x{i}; % compute 'net' inputs to current layer

if i < layersLength-1 % inner layers(hidden layers)
    x{i+1} = [(1-exp(-a.*y{i}(1:end-1)));1]./[(1+exp(-a.*y{i}(1:end-1)));1];%Scalero paper.
else % output layer
    x{i+1} = ( 1 - exp(-a.*y{i}) )./(1 + exp(-a.*y{i}));%Scalero paper.
    if( x{i+1} <= 0.5 && x{i+1}>= 0.1 )
        x{i+1} = round2dp(x{i+1},1);
    else
        x{i+1} = 0.1;
    end
end

end

end
PredictedOutputs(p,1)= x{end};
```


Step 4: Calculate the Kalman gain $K_j(t)$ for each j th layer to update R_j^{-1} correlation matrix:

$$K_j(t) = [R_j^{-1}(t-1)x_{j-1}(t)]/[b_j + x_{j-1}^T(t)R_j^{-1}(t-1)x_{j-1}(t)]$$

```
%=====
% Step #4 :Invoke Kalamn Filter equations
%=====
for i=1:layersLength-1 % hidden layers

K{i} = (R_inv{i}*x{i})*inv(b + x{i}'*R_inv{i}*x{i});%Scalero paper.

R_inv{i} = ( R_inv{i} - K{i}*x{i}'*R_inv{i})/b ;%Scalero paper.

end
```

$$R_j^{-1}(t) = [[R_j^{-1}(t-1) - K_j(t)x_{j-1}^T(t)R_j^{-1}]/b$$

Step 5: Back -propagate the error signals through the network

$$e_L = f'(y)(O - x) \quad \text{for output layer L}$$

The error signal for the interior hidden layer (j) is:

$$e_j = f'(y_j) \sum_j (e_{j+1} * w_{j+1})$$

Where $f'(y_j)$ is the first derivative of the activation function $f(y_j)$ and in our case the derivative is given by:

$$f'(y_{jk}) = 2a[\exp(-ay_{jk})]/[1 + \exp(-ay_{jk})]^2$$

Step 6: Update the weights vector in each layer j up to the output layer L

$$w_{Lk}(t) = w_{Lk}(t-1) + K_L(t) (d_k - y_{Lk}) \quad \text{for output layer}$$

$$w_{jk}(t) = w_{jk}(t-1) + K_j(t) (\mu e_{jk}(t)) \quad \text{for } j^{\text{th}} \text{ hidden layer}$$

```
%=====
% Step #5: Backpropagate error signals.
%=====
diff = O-x{end};
% Calculatæ the error (difference between actual and target and
accumulate the error .
err{end} = (((2*a*(exp(-a*y{end}))) ./ ((1 + exp(-a*y{end})) .^2 ))) .*(O-
x{end}) ;%Scalero paper
%Calculate error for hidden layers
for i=layersLength-1:-1:1
if i > 1 % don't compute delta for input layers .
err{i-1} = (((2*a*(exp(-a*y{i-1}))) ./ ((1 + exp(-a*y{i-1})) .^2
))) .*(err{i}'*weights{i}); %Scalero paper.
end
end
```


where d_k is the desired summation output of k^{th} node of the L^{th} layer, and is given by:

```
%=====
% Step #6 : Find the desired summation output
%=====
d = (1/a).*log( (1+O)./(1-O) ); %Scalero paper

%=====
% Step #7 :Update the weights
%=====

%For Output layer
weights{end} = weights{end} + (d - y{end})*K{end}'; % Scalero paper

%For hidden layers
for i=leayersLength-1:-1:1
if i > 1 % don't compute weights for input layers .
weights{i-1} = weights{i-1} + (mu*err{i-1})*K{i-1}';%Scalero Paper
end
end
```

$$d_k = \left(\frac{1}{a}\right) \ln\left(\frac{1 + O_k}{1 - O_k}\right)$$

Step 7: check if MSE reach its desired value, if ‘YES’ then quit the algorithm and if ‘No’, then return back to step 2 of this algorithm.

```
%=====
% Step #8: Test for completion
%=====
sse = sse + sum((diff).^2);
end %end of for loop.
IterationsCount = IterationsCount +1;
MSE = sse/(IterationsCount*targetNodesCount);% MSE = 1/trainingDataCount
* 1/targetNodesCount * summed squared error
end %end of while loop
```

3.3.5 Proposed BPRLS IDS Testing Phase and MATLAB Implementation

In this section we recursive least squares back propagation neural network algorithm will be tested or plug it into the IDS model. The optimal weights that obtained in training phase will be used in the testing phase of BPRLS network.

1. Load Training Dataset.(1% of NSL-KDD Database)
2. Run Data Pre-Processing function
3. Call the Testing function of BPRLS network .MATLAB implementation of it is shown below.

```
function [Results,PredictedOutputs] =
KalmanBackpropagationTesting(inputs,targets,layers, weights)
% This function implement Scalero algorithm
% Inputs : inputs : input data
%          targets: Desired response.
%          Layers : the structure of the network .
%          weights of the trainig phase .
% Outputs :Results
%          PredictedOutputs
%**
%*****
%% Phase #1 : Input Validation
%*****

[trainingDataCount,inputNodesCount] = size(inputs);
[targetDataCount,targetNodesCount] = size(targets);

if trainingDataCount ~= targetDataCount
error('BackpropagationTraining:TrainingAndTargetDataLengthMismatch', 'The
number of input vectors and desired ouput vectors do not match');
end
%=====
%% Phase 2 : Initialization
%=====
layersLength = length(layers);

% Initialize the activation vectors, all layers include biase except the
% output layer .and %initialize the the inverse matrix  $R^{-1}$ 
inputs = [inputs ones(trainingDataCount,1)];% affix the biase coloumn of
activation to the input layer ,namely,input + 1 for the bias node activation
x = cell(layersLength,1);
for i=1:layersLength-1
x{i} = ones(layers(i)+1,1); % inner layers include a bias node
(trainingDataCount-by-Nodes+1
end
x(end) = ones(layers(end),1); % no bias node at output layer
```

```

% Initialize the net input of layers ,one for each node of that layer , but
% there is no net for the input layer .
y = cell(layersLength-1,1); % one net matrix for each layer exclusive input
for i=1:layersLength-2;
y(i) = ones(layers(i+1)+1,1); % affix bias node
end
y(end) = ones(layers(end),1);

a =0.2;
PredictedOutputs = ones(size(targets));
sse =0;
=====
%Step #3 : Run selected pattern through the network .
=====
%Compute the inputs and outputs to each layer
for p=1:trainingDataCount
    %get the current input vector and it's desired output .
    x(1) = inputs (p,:)';
    O    = targets(p,:)';

    for i=1:layersLength-1
        y(i) = weights(i)*x(i); % compute 'net' inputs to current layer

        if i < layersLength-1 % inner layers(hidden layers)
            x(i+1) = [(1-exp(-a.*y(i)(1:end-1)));1]./[(1+exp(-a.*y(i)(1:end-1)))]);%Scalero paper.

        else % output layer
            x(i+1) = ( 1 - exp(-a.*y(i)) )./(1 + exp(-a.*y(i)));%Scalero
            paper.

            if( x(i+1) <= 0.5 && x(i+1)>= 0.1 )

                x(i+1) = round2dp(x(i+1),1);

            else
                x(i+1) = 0.1;
            end

        end
    end
    PredictedOutputs(p,1) = x(end);
    diff = (O-x(end));
    sse = sse + sum((diff).^2);% sum of the error for all samples, and all nodes
end

MSE = sse/(trainingDataCount*targetNodesCount); % MSE = 1/trainingDataCount *
1/targetNodesCount + summed squared error
% return the trained network
Results.MSE = MSE;
Results.sse = (sse/trainingDataCount)*100;
Results.PredictedOutputs = PredictedOutputs;

```

4. Finally, run Confusion Matrix function to evaluate the performance of IDS ,as shown below :

```
function stats = confusionmatStats(group,grouphat)
% INPUT
% group = true class labels
% grouphat = predicted class labels
%
% OR INPUT
% stats = confusionmatStats(group);
% group = confusion matrix from matlab function (confusionmat)
%
% OUTPUT
% stats is a structure array
field1 = 'confusionMat';
if nargin < 2
    value1 = group;
else
    value1 = confusionmat(group,grouphat);
end

numOfClasses = size(value1,1);
totalSamples = sum(sum(value1));

field2 = 'accuracy'; value2 =
(2*trace(value1)+sum(sum(2*value1)))/(numOfClasses*totalSamples);

[TP,TN,FP,FN,sensitivity,specificity,precision,f_score] =
deal(zeros(numOfClasses,1));
for class = 1:numOfClasses
    TP(class) = value1(class,class);
    tempMat = value1;
    tempMat(:,class) = []; % remove column
    tempMat(class,:) = []; % remove row
    TN(class) = sum(sum(tempMat));
    FP(class) = sum(value1(:,class))-TP(class);
    FN(class) = sum(value1(class,:))-TP(class);
end

for class = 1:numOfClasses
    sensitivity(class) = TP(class) / (TP(class) + FN(class));
    specificity(class) = TN(class) / (FP(class) + TN(class));
    precision(class) = TP(class) / (TP(class) + FP(class));
    f_score(class) = 2*TP(class)/(2*TP(class) + FP(class) + FN(class));
end
field3 = 'sensitivity'; value3 = sensitivity;
field4 = 'specificity'; value4 = specificity;
field5 = 'precision'; value5 = precision;
field6 = 'recall'; value6 = sensitivity;
field7 = 'Fscore'; value7 = f_score;
stats =
struct(field1,value1,field2,value2,field3,value3,field4,value4,field5,value5,fi
eld6,value6,field7,value7);
end
```

Chapter Four
Experimental Results and Systems
Performance Analysis

Chapter Four

Experimental Results and Systems Performance

Analysis

4.1 Implementation Specifications

In this chapter experimental results obtained from the testing of the proposed Intrusion Detection Systems explained in chapter three will be discussed, the systems are implemented using:

- ✓ MATLAB 2012a as a powerful integrated development environment.
- ✓ Windows 7 platform with Intel Core i7 2Due CPU 2.8 GHZ with RAM 8.0GB.
- ✓ NSL-KDD dataset (<http://nsl.cs.unb.ca/NSL-KDD/>).

4.2 Evaluation Criteria

The proposed systems tested using NS L- KDD data set separated into two sets:

- The first is training stage.
- The second is testing stage.

The neural network is trained using less than 8% of the NSL-KDD labelled dataset until we reach the desired MSE, then the neural network tested using less than 2% of the dataset.

The performance of the proposed IDS systems is evaluated using true positive (TP), true negative (TN), false negative (FN), false positive (FP), Accuracy, False

Positive Rate (F PR), False Negative Rate (F NR), True Negative Rate (T NR), Detection Rate (DR), and Precision.

True positive indicates the quantity of aggressions connection records that exactly classified by IDS and it considered a sign of proper detection of attack whereas the true negative indicates the number of legal records that are correctly classified.

False positive indicates the connection records that were incorrectly classified as invalid records whereas they are valid connections and it represent the accuracy of the detection system.

False negative indicates connection records that were incorrectly classified as legal activities whereas they are intrusion activities (attacks).

A false negative is a direct sign of the inability of IDS to detect the intrusion; the formulas bellow indicates the evaluation measure used to evaluate the three systems

$$(T\ PR), (DR) \text{ or sensitively} = \frac{TP}{TP+FN} \dots\dots\dots (4.1)$$

$$(T\ NR) \text{ or specificity} = \frac{TN}{TN+FP} \dots\dots\dots (4.2)$$

$$(FP\ R) = \frac{FP}{TN+FP} = 1 - \text{specificity} (T\ NR) \dots\dots\dots (4.3)$$

$$(FN\ R) = \frac{FN}{TP+FN} = 1 - \text{sensitivity} (DR) \dots\dots\dots (4.4)$$

$$\text{Accuracy (AC)} = \frac{TN+TP}{TN+TP+FN+FP} \dots\dots\dots (4.5)$$

$$\text{Precision} = \frac{TP}{TP+FP} \dots\dots\dots (4.6)$$

4.3 Experimental Results of the Proposed K-Means IDS Algorithm

The K-Means implementation consists of two stages: Training and Testing, in the training stage the K-Means IDS is trained to attain the best centroids that will be used as initial centroids in the testing stage of the algorithm.

4.3.1 Experimental Results of the K-Means IDS Algorithm: Training Phase

The statistics of the five classes in the training dataset are shown in table (4.1) and graphically in figure (4.1).

Table 4.1: shows the number of record classes in 8% training NSL-KDD dataset.

Record Type	Normal	DoS	Probe	R2L	U2R
Record Frequency	5551	3872	968	126	27

Total number of Records = 10544

Total number of Attack records = 4993

Figure (4.1) depicts the distribution of the five classes of connection records that classified by K-Means IDS algorithm.

The Statistics of Record Classes in K-Means IDS Training Stage

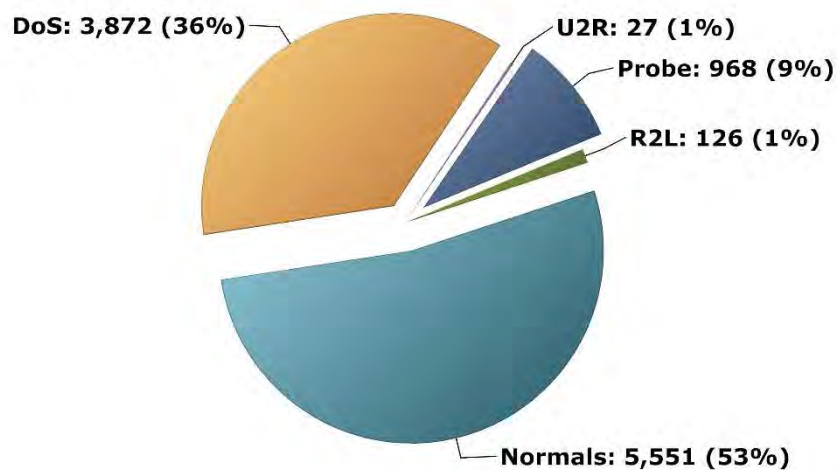


Figure 4.1: The Statistics of record classes in K-Means IDS Training Stage.

The confusion matrix followed by the performance evaluation results for each type of attack for the Training stage of K-Means based IDS are best illustrated in the tables (4.2) , (4.3) and (4.4).

Table 4.2: Confusion Matrix of Training Stage of K-Means IDS.

Actual	Predicted Normal	Predicted DoS	Predicted Probe	Predicted R2L	Predicted U2R	Total
Normal	2220	31	34	1801	1465	5551
DoS	27	2840	604	278	123	3872
Probe	52	40	420	13	443	968
R2L	1	1	2	21	101	126
U2R	1	0	0	0	26	27
Total	2301	2912	1060	2113	2158	10544

Table 4.3: The performance evaluation metrics of the K-Means based IDS in Training phase.

Attack Type	TN	FP	FN	TP	Specificity (TNR)	Sensitivity (TPR, DR, Recall)	FPR(1-specificity)	FNR(1-sensitivity)	Accuracy	Precision
DoS	2220	31	27	2840	0.986228	0.990582	0.0137717	0.00941751	0.988667	0.989202
Probe	2220	34	52	420	0.984916	0.889831	0.0150843	0.110169	0.968452	0.92511
R2L	2220	1801	1	21	0.552101	0.954545	0.447899	0.0454545	0.554291	0.0115258
U2R	2220	1465	1	26	0.602442	0.962963	0.397558	0.037037	0.605065	0.017438

Table 4.4: Confusion Matrix (total) of the K-Means based IDS Training phase.

	Predicted Positive	Predicted Negative
Actual Positive	4912(TP)	81(FN)
Actual Negative	3331(FP)	2220(TN)

The training stage of K-Means algorithm is just to obtain the optimal (best) centroids to be used as initial centroids for the testing stage. To judge on these centroids, if they are optimal or not, we use the confusion matrix, and the performance

metrics of the training stage, and the results shown above represent best performance, so we adopt the centroids that resulted from this stage and use it in the testing stage as shown in the following section. The overall performance of the algorithm in training phase is listed in the table (4.5).

Table 4.5: Total performance of the K-Means based IDS Training phase.

Performance Metric	Value
Accuracy	0.676
Precision	0.595
TPR	0.983
TNR	0.399
FPR	0.600
FNR	0.0162

4.3.2 Experimental Results of the K-Means IDS Algorithm: Testing Phase

In testing phase, we used a sample of 2094 records to test the performance of our proposed K-Means IDS as shown in table (4.6).

Table 4.6: shows the number of record classes in 2% testing NSL-KDD dataset.

Record Type	Normal	DoS	Probe	R2L	U2R
Record Frequency	927	693	202	265	7

Total number of Records = 2094

Total number of Attack records = 1167

Figure (4.2) shows the records classes distribution in the testing data set, where this time , Probes records come below the R2L classes and this different distribution will prove the efficiency of our proposed system to detect the attacks records regardless its frequency and without bias toward high frequent records.

This is proved by the confusion matrix and the values of performance evaluation metrics listed in tables (4.7), (4.8) and (4.9).

The Statistics of Record Classes in K-Means IDS Testing Stage

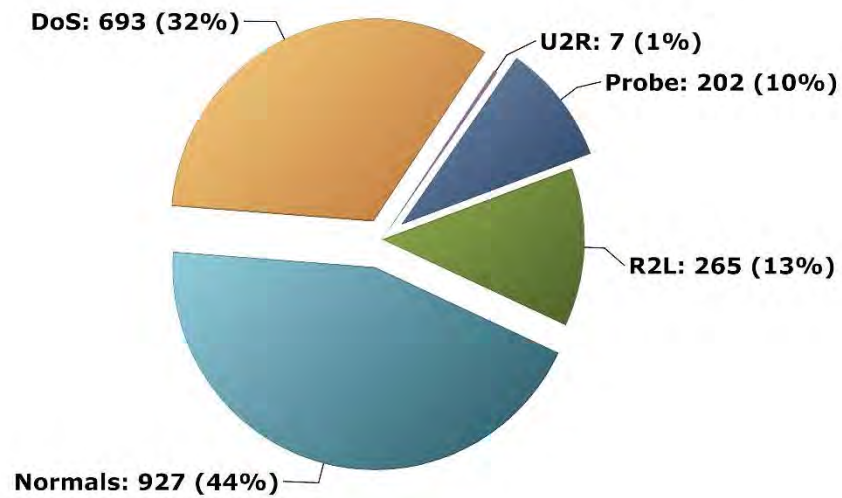


Figure 4.2: The Statistics of record classes in K-Means IDS Testing Stage.

We achieve high detection rate reach up to (100%) for DoS attacks and U2R attacks, which means high detection rates for two types of attacks different extremely in their distribution and frequency. Moreover, we achieve high accuracy reach up to (99.816 %) for DoS attacks and (96.812 %) for Probe attacks.

Table 4.7: Confusion Matrix of Testing Stage of K-Means IDS.

Actual	Predicted Normal	Predicted DoS	Predicted Probe	Predicted R2L	Predicted U2R	Total
Normal	375	1	11	430	110	927
DoS	0	169	293	175	56	693
Probe	5	25	111	0	61	202
R2L	13	1	13	98	140	265
U2R	0	0	0	0	7	7
Total	393	196	428	703	374	2094

Table 4.8: The performance evaluation metrics of the K-Means based IDS in Testing phase.

Attack Type	TN	FP	FN	TP	Specificity (TNR)	Sensitivity (TPR, DR, Recall)	FPR(1-specificity)	FNR(1-sensitivity)	Accuracy	Precision
DoS	375	1	0	169	0.99734	1	0.00265957	0	0.998165	0.994118
Probe	375	11	5	111	0.971503	0.956897	0.0284974	0.0431034	0.968127	0.909836
R2L	375	430	13	98	0.465839	0.882883	0.534161	0.117117	0.516376	0.185605
U2R	375	110	0	7	0.773169	1	0.226804	0	0.776423	0.0598291

Table 4.9: Confusion Matrix (total) of the K-Means based IDS Testing phase.

	Predicted Positive	Predicted Negative
Actual Positive	1149 (TP)	18 (FN)
Actual Negative	552(FP)	375(TN)

Moreover, the total performance of the algorithm in its testing phase is illustrated in table (4.10).

Table 4.10: Total performance of the K-Means based IDS Testing phase.

Performance Metric	Value
Accuracy	0.727
Precision	0.675
TPR	0.984
TNR	0.404
FPR	0.595
FNR	0.015

Figure (4.3) depicts the graphical comparison of accuracy of the experimental results of K-Means IDS obtained in case of training stage to that obtained in testing stage.

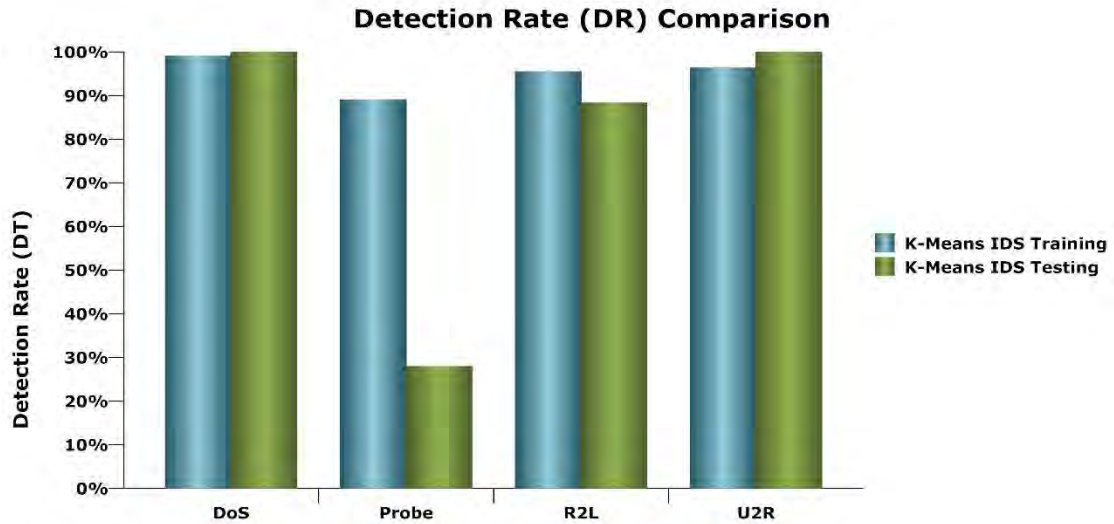


Figure 4.3: Graphical comparison of K-Means IDS Training stage and Testing stage

4.4 Experimental Results of the Proposed Novel KNN IDS Algorithm

Unlike many clustering techniques and artificial learners, the Novel KNN algorithm do not abstract any information from the training data during the learning phase, or as we called it the Training phase. Training Phase is merely a question of encapsulating the training data, and this is the case not for Novel KNN but for all *instance-based* learners.

All examples -based learning are called stupid learners since they keep the training data set to use it in the testing phase later on, so we will evaluate the performance of the testing phase, which represent the actual operation of the Novel KNN IDS algorithm.

The statistics of the five classes in the training dataset that will be kept by the Novel KNN to later usage in the testing phase is shown in table (4.1) and graphically in figure (4.1) above. Moreover, the statistics of the five classes in the testing dataset that used to evaluate the performance of the proposed Novel KNN IDS is shown in table (4.5) and graphically in figure (4.2) above.

The confusion matrix and the metrics values of the performance evaluation of the proposed Novel KNN IDS are listed in tables (4.11), (4.12) and (4.13).

Table 4.11 Confusion Matrix of Testing Stage of proposed Novel KNN IDS.

Actual	Predicted Normal	Predicted DoS	Predicted Probe	Predicted R2L	Predicted U2R	Total
Normal	76	52	4	459	336	927
DoS	57	436	32	152	16	693
Probe	29	65	69	0	39	202
R2L	98	0	4	100	63	265
U2R	1	0	0	0	6	7
Total	261	553	109	711	460	2094

Table 4.12: The performance evaluation metrics of the Novel KNN based IDS.

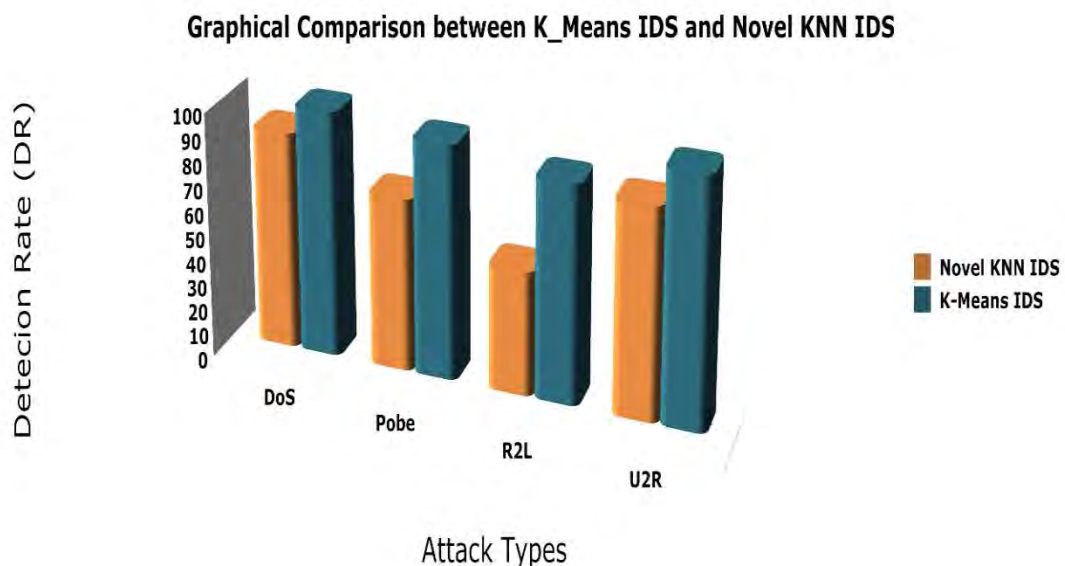
Attack Type	TN	FP	FN	TP	Specificity (TNR)	Sensitivity (TPR ,DR ,Recall)	FPR(1-specificity)	FNR(1-sensitivity)	Accuracy	Precision
DoS	76	52	57	436	0.59375	0.884381	0.40625	0.115619	0.824477	0.893443
Probe	76	4	29	69	0.95	0.704082	0.05	0.295918	0.814607	0.945205
R2L	76	459	98	100	0.142056	0.505051	0.857944	0.494949	0.240109	0.178891
U2R	76	336	1	6	0.184466	0.857143	0.815534	0.142857	0.195704	0.0175439

Table 4.13: Confusion Matrix (total) of the Novel KNN based IDS.

	Predicted Positive	Predicted Negative
Actual Positive	982 (TP)	185 (FN)
Actual Negative	851(FP)	76(TN)

As table (4.13) shows the use of the equation (4.1), the overall detection rate is 84.14%, whereas in case of K-Means IDS, we attained 98.45%, since the Novel KNN algorithm is instance-based learning algorithm that depends only on the similarity function, and begin the testing phase directly. Whereas, in the case of K-Means algorithm; it first attain the best centroids, and then begin the testing, which enhance the performance of the algorithm?

A graphical comparison between K-Means algorithm and Novel KNN algorithm is depicted in figure (4.4) in terms of detection rate for each type of attack.

**Figure 4.4: Graphical detection rate comparison between K-Means IDS and Novel KNN IDS.**

4.5 Experimental Results of the Proposed BPRLS IDS

As all supervised learning algorithms, the proposed BPRLS IDS contains two main phases: Training and testing. In the first stage, BPRLS IDS network will be trained to attain the optimal weights that will be used in the second stage.

The proposed system depends in its core on Kalamn theory, which belongs to the recursive least squares family, which will enhance the back propagation algorithm extremely. Consequently, it is sufficient to train the network using less than less than 5% of training dataset, and testing the results with less than 1% of the dataset. On the other hand, if we increase the number of records used in either Training or Testing stage, then the algorithm will diverge because it will reach to extremely low levels of error (approximately zero) which cause the Kalman Matrix gain to be a singular one and this affect the total results of algorithm .

Since the algorithm can learn with less number of records and can reach to satisfied levels of MSE, then the advantage will employed to build IDS with high performance.

Experimental results of the proposed BPRLS IDS will be examined in two separate experiments; each one has its own network parameters and topology in training and testing phases.

4.5.1 Experimental Results of the First Experiment of BPRLS IDS

Algorithm: Training Phase

The topology and the parameters of the recursive least square neural network is listed in table (4.14) followed by the statistics of the five classes in the training dataset shown in table (4.15) and graphically in figure (4.5).

Table 4.14: shows parameters and topology of our proposed Neural Network of first experiment.

Parameters	Value ('Description')
Layers	[41 20 10 1]
Performance	MSE
μ	40
B	0.99
Transfer function	Sigmoid Function
No. of epochs	30

Table 4.15: shows the number of record classes in less than 5% of training NSL-KDD dataset.

Record Type	Normal	DoS	Probe	R2L	U2R
Record Frequency	927	693	202	256	7

Total number of Records = 2094

Total number of Attack records = 1167

Figure (4.5) depicts the distribution of the five classes of connection records that classified by BPRLS IDS algorithm.

The Statistics of Record Classes in BPRLS IDS Training Stage

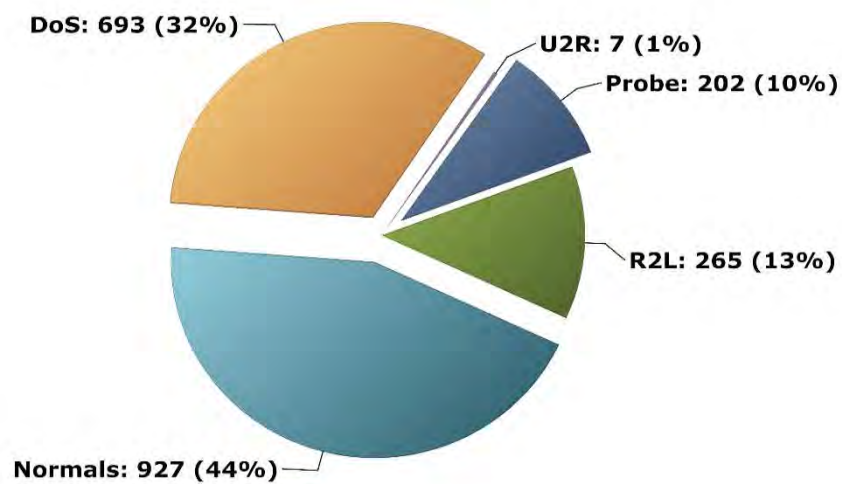


Figure 4.5: the Statistics of record classes in BPRLS IDS Training Stage

The confusion matrix followed by the performance evaluation results of the first experiment for each type of attack for the Training stage of BPRLS based IDS are best illustrated in the tables (4.16) , (4.17) ,(4.18) and (4.19).

Table 4.16: First Experiment Confusion Matrix of Training Stage of BPRLS IDS.

Actual	Predicted Normal	Predicted DoS	Predicted Probe	Predicted R2L	Predicted U2R	Total
Normal	773	125	24	5	0	927
DoS	26	636	31	0	0	693
Probe	1	30	166	5	0	202
R2L	1	20	89	155	0	265
U2R	1	1	2	3	0	7
Total	802	812	312	168	0	2094

Table 4.17: First Experiment performance evaluation metrics of BPRLS IDS in Training phase.

Attack Type	TN	FP	FN	TP	Specificity (TNR)	Sensitivity (TPR, DR, Recall)	FPR(1-specificity)	FNR(1-sensitivity)	Accuracy	Precision
DoS	773	125	26	636	0.860802	0.960725	0.139198	0.0392749	0.903205	0.835742
Probe	773	24	1	166	0.969887	0.994012	0.0301129	0.00598802	0.974066	0.873684
R2L	773	5	1	155	0.993573	0.99359	0.00642674	0.00641026	0.993576	0.96875
U2R	773	0	1	0	100%	0	0	1	0.998708	0

Table 4.18: First Experiment Confusion Matrix (total) of the IDS BPRLS Training phase.

	Predicted Positive	Predicted Negative
Actual Positive	1138(TP)	29(FN)
Actual Negative	154(FP)	773(TN)

Table 4.19: Total performance of the IDS BPRLS Training phase.

Performance Metric	Value
Accuracy	0.912
Precision	0.880
TPR	0.975
TNR	0.083
FPR	0.166
FNR	0.024

The training stage of BPRLS algorithm is just to obtain the optimal (best) weights; to be used to build the RLS BP neural network for the testing stage. To judge on these weights if they are optimal or not, our neural network will train until it reaches the optimal MSE. Then, the results will be examined of confusion matrix, and, the performance metrics of the training stage, and, the results that shown above represent satisfied performance, so we adopt the weights that obtained in this stage, and, use it in the testing stage as shown in the following section.

4.5.2 Experimental Results of the First Experiment of BPRLS IDS

Algorithm: Testing Phase

The topology and the parameters of the recursive least square neural network, which were used in the testing stage, exactly the same as that used in the training stage. The statistics of the five classes in the training dataset are shown in table (4.20), and, graphically in figure (4.6).

Table 4.20 shows the number of record classes in less than 1% of testing NSL-KDD dataset.

Record Type	Normal	DoS	Probe	R2L	U2R
Record Frequency	499	346	98	46	11

Total number of Records = 1000 Total number of Attack records = 501

The Statistics of Record Classes in BPRLS IDS Testing Stage

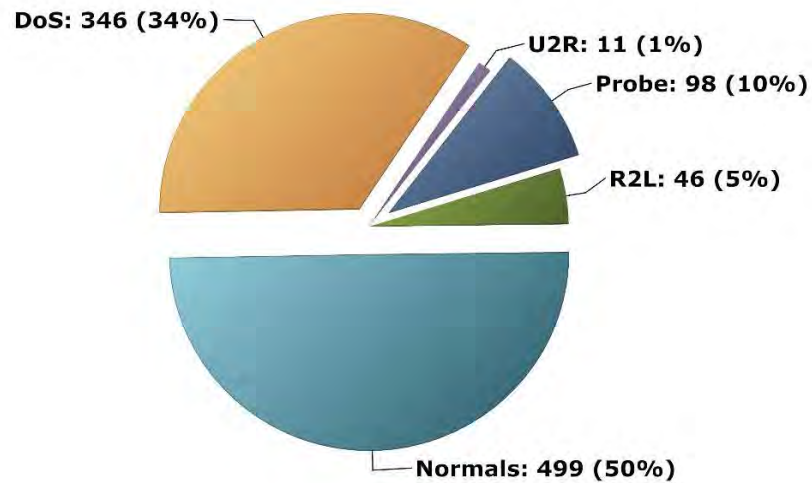


Figure 4.6: The Statistics of record classes in BPRLS IDS Testing Stage in the first experiment.

The confusion matrix followed by the performance evaluation results listed in the tables (4.21), (4.22) and (4.23).

Table 4.21: First Experiment Confusion Matrix of Testing Stage of BPRLS IDS.

Actual	Predicted Normal	Predicted DoS	Predicted Probe	Predicted R2L	Predicted U2R	Total
Normal	203	257	33	6	0	499
DoS	130	201	15	0	0	346
Probe	6	36	54	2	0	98
R2L	2	17	5	16	6	46
U2R	1	3	6	1	0	11
Total	342	514	113	25	6	1000

Table 4.22: First Experiment performance evaluation metrics of BPRLS IDS in Testing phase.

Attack Type	TN	FP	FN	TP	Specificity (TNR)	Sensitivity (TPR ,DR ,Recall)	FPR(1-specificity)	FNR(1-sensitivity)	Accuracy	Precision
DoS	203	257	130	201	0.441304	0.607251	0.558696	0.392749	0.510746	0.438865
Probe	203	33	6	54	0.860169	0.90	0.139831	0.1	0.868243	0.62069
R2L	203	6	2	16	0.971292	0.888889	0.0287081	0.111111	0.964758	0.727273
U2R	203	0	1	0	1	0	0	1	0.995098	0

Table 4.23:First Experiment Confusion Matrix (total) of the IDS BPRLS Testing phase.

	Predicted Positive	Predicted Negative
Actual Positive	362 (TP)	139 (FN)
Actual Negative	296(FP)	203(TN)

The overall performance of the first experiment is shown in the table (4.24) below which depends on the total confusion matrix that shown in the table (4.23) above.

Table 4.24:Total performance of the First Experiment of the IDS BPRLS.

Performance Metric	Value
Accuracy	0.565
Precision	0.5501
TPR	0.722
TNR	0.406
FPR	0.593
FNR	0.277

Figure (4.7) depicts the graphical comparison of detection rate of the first experiment of BPRLS IDS obtained in the case of training stage to that obtained in the testing stage. The results of both stages are close to each other, due to weight optimality, where we achieve $MSE = 0.1920$, and $MSE = 0.0081$ in training and testing phases respectively.

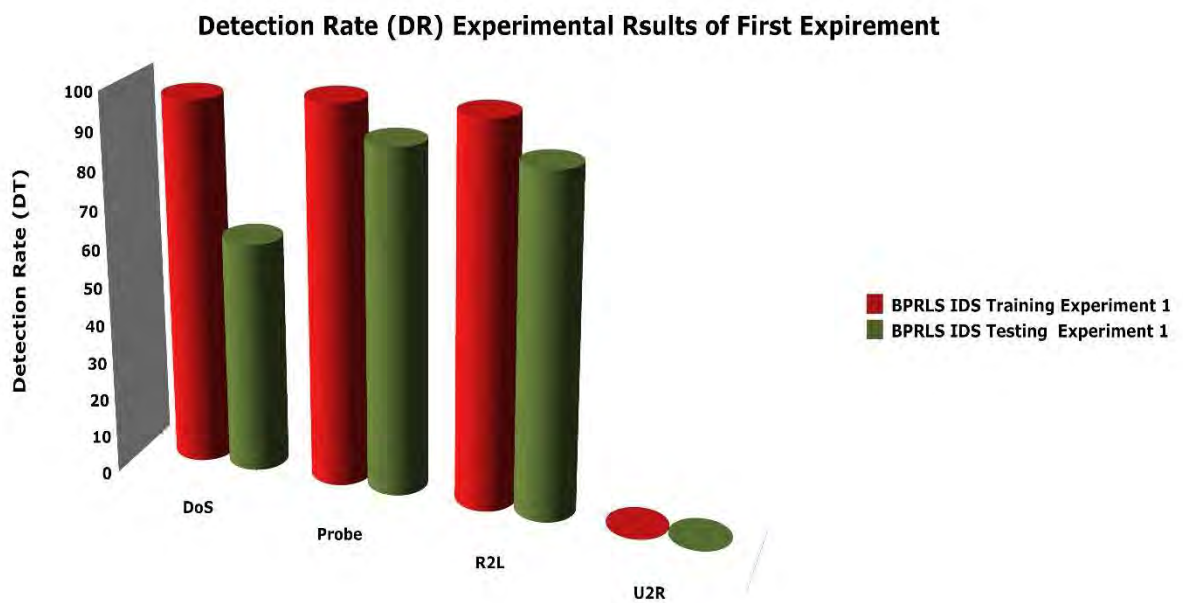


Figure 4.7: Graphical detection rate comparison between BPRLS IDS Training and testing phases of first experiment.

4.5.3 Experimental Results of the Second Experiment of BPRLS IDS

Algorithm: Training Phase

The topology and the parameters of the recursive least square neural network used in second experiment is listed in table (4.25). The statistics of the five classes in the training dataset are shown in table (4.15), and graphically in figure (4.5) above.

Table 4.25: shows parameters and topology of our proposed Neural Network of second experiment.

Parameters	Value ('Description')
Layers	[41 35 25 1]
Performance	MSE
μ	40
B	0.99
Transfer function	Sigmoid Function
No. of epochs	30

the number of nodes (neurons) we changed in each layer of BPRLS neural network. The confusion matrix followed by the performance evaluation results of the second experiment for each type of attack for the Training stage of BPRLS are illustrated in the tables: (4.26), (4.27), and (4.28).

Table 4.26: Second Experiment Confusion Matrix of Training Stage of BPRLS IDS.

Actual	Predicted Normal	Predicted DoS	Predicted Probe	Predicted R2L	Predicted U2R	Total
Normal	792	104	22	8	1	927
DoS	35	618	40	0	0	693
Probe	2	33	157	10	0	202
R2L	7	9	91	153	5	265
U2R	3	1	1	2	0	7
Total	839	765	311	173	6	2094

Table 4.27: Second Experiment performance evaluation of BPRLS IDS in Training phase.

Attack Type	TN	FP	FN	TP	Specificity (TNR)	Sensitivity (TPR, DR, Recall)	FPR(1-specificity)	FNR(1-sensitivity)	Accuracy	Precision
DoS	792	104	35	618	0.903346	0.946401	0.0966543	0.0535988	0.919607	0.855956
Probe	792	22	2	157	0.97867	0.987421	0.0221328	0.0125786	0.979185	0.877095
R2L	792	8	7	153	0.991837	0.95625	0.00816327	0.04375	0.986842	0.950311
U2R	792	1	3	0	0.998972	0	0.00102775	1	0.995902	0

Table 4.28: Second Experiment Confusion Matrix (total) of the IDS BPRLS Training phase.

	Predicted Positive	Predicted Negative
Actual Positive	1120(TP)	47(FN)
Actual Negative	135(FP)	792(TN)

The overall performance of the second experiment is shown in the table (4.29) below which depends on the total confusion matrix that shown in the table (4.28) above.

Table 4.29: Total performance of the Second Experiment of the IDS BPRLS in Training Phase.

Performance Metric	Value
Accuracy	0.91996
Precision	0.89243
TPR	0.95972
TNR	0.87804
FPR	0.12195
FNR	0.04027

4.5.4 Experimental Results of the Second Experiment of BPRLS IDS

Algorithm: Testing Phase

The topology and the parameters of the recursive least square neural network that used in the testing stage are exactly the same as that used in the training stage.

Tables (4.30), (4.31) and (4.32) show the confusion matrix and its associated performance metrics.

Table 4.30: Second Experiment Confusion Matrix of Testing Stage of BPRLS IDS.

Actual	Predicted Normal	Predicted DoS	Predicted Probe	Predicted R2L	Predicted U2R	Total
Normal	169	257	60	12	1	499
DoS	11	235	91	9	0	346
Probe	1	28	56	11	2	98
R2L	0	3	12	30	1	46
U2R	0	3	2	2	4	11
Total	181	526	221	64	8	1000

Table 4.31: Second Experiment performance evaluation of BPRLS IDS in Testing phase.

Attack Type	TN	FP	FN	TP	Specificity (TNR)	Sensitivity (TPR ,DR ,Recall)	FPR(1-specificity)	FNR(1-sensitivity)	Accuracy	Precision
DoS	169	257	11	235	0.396714	0.955285	0.603286	0.0447154	0.60119	0.477642
Probe	169	60	1	65	0.737991	0.982456	0.262009	0.0175439	0.786713	0.482759
R2L	169	12	0	30	0.933702	100%	0.0662983	0	0.943128	0.714286
U2R	169	1	0	4	0.994118	100%	0.00588235	0	0.994253	0.8

Table 4.32: Second Experiment Confusion Matrix (total) of the IDS BPRLS Testing phase.

	Predicted Positive	Predicted Negative
Actual Positive	489(TP)	12(FN)
Actual Negative	330(FP)	169(TN)

The overall performance of the second experiment is shown in the table (4.33) below which depends on the total confusion matrix that shown in the table (4.32) above.

Table 4.33: Total performance of the Second Experiment of the IDS BPRLS in Testing phase.

Performance Metric	Value
Accuracy	0.658
Precision	0.59707
TPR	0.9760
TNR	0.3386
FPR	0.6613
FNR	0.02395

Figure (4.8) shows that the obtained detection rate in all types of attacks is better than that obtained in the first experiment, due to the change in the number of nodes in our neural network .This is equivalent to ANN tuning process. Where we achieve MSE = 0.2213, and MSE = 0.0087 in training and testing phases respectively.

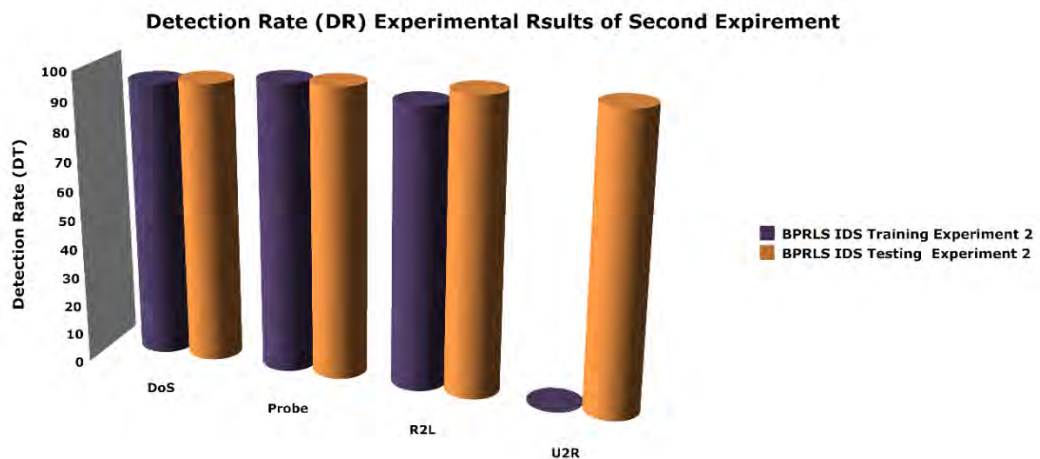


Figure 4.8: Graphical detection rate comparison between BPRLS IDS Training and Testing phases of Second experiment.

4.6 Experimental Results Comparative Analysis between the Proposed Intrusion Detection Systems.

This section is dedicated to compare the performance of the proposed intrusion detection systems: K-Means IDS, Novel KNN IDS, and BPRLS IDS. At first, the comparison of detection rate and the achieved accuracy. As figure (4.9) shows that BPRLS IDS occupy the first place, where it has the highest detection rates comparing to the other proposed techniques, which proves the high capabilities of the recursive least squares back propagation algorithm.

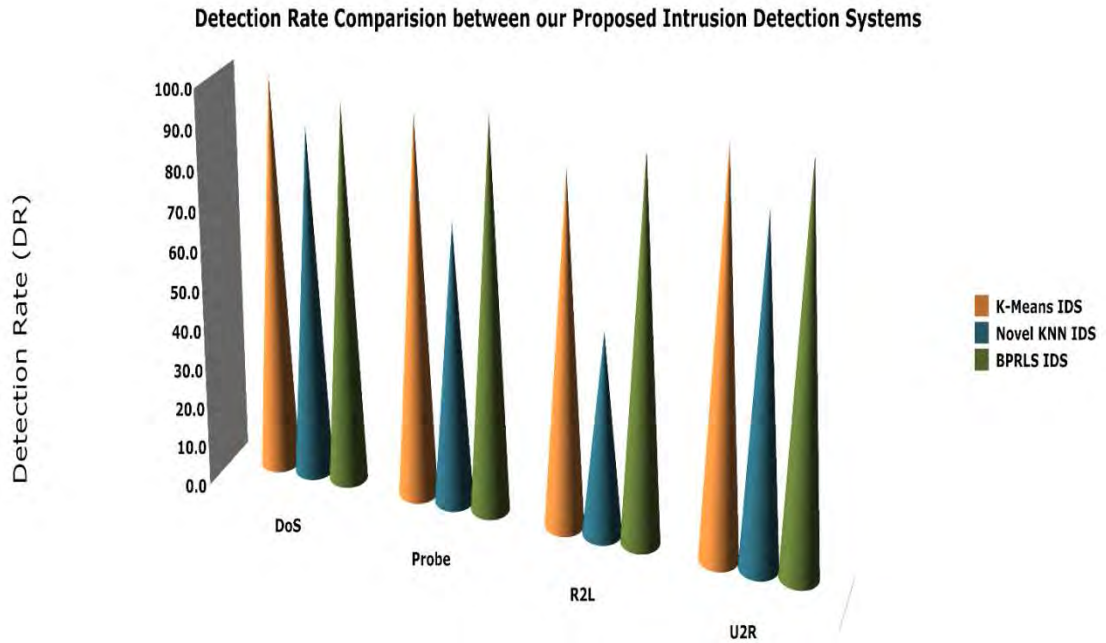


Figure 4.9: Graphical detection rate comparison between the intrusion detection systems.

In the case of comparing in terms of *accuracy*, our BPRLS algorithm occupies the first place once again, and the Novel KNN technique comes at last. This indicate the proposed Novel KNN algorithm has a weak point in differentiation the normal records, and although it is powerful in detecting the least frequent attacks, it has a

problem in distinguish it whether it is normal or attack record and bias toward least frequent attacks, which leads to low accuracy and appears apparently in figure(4.10).

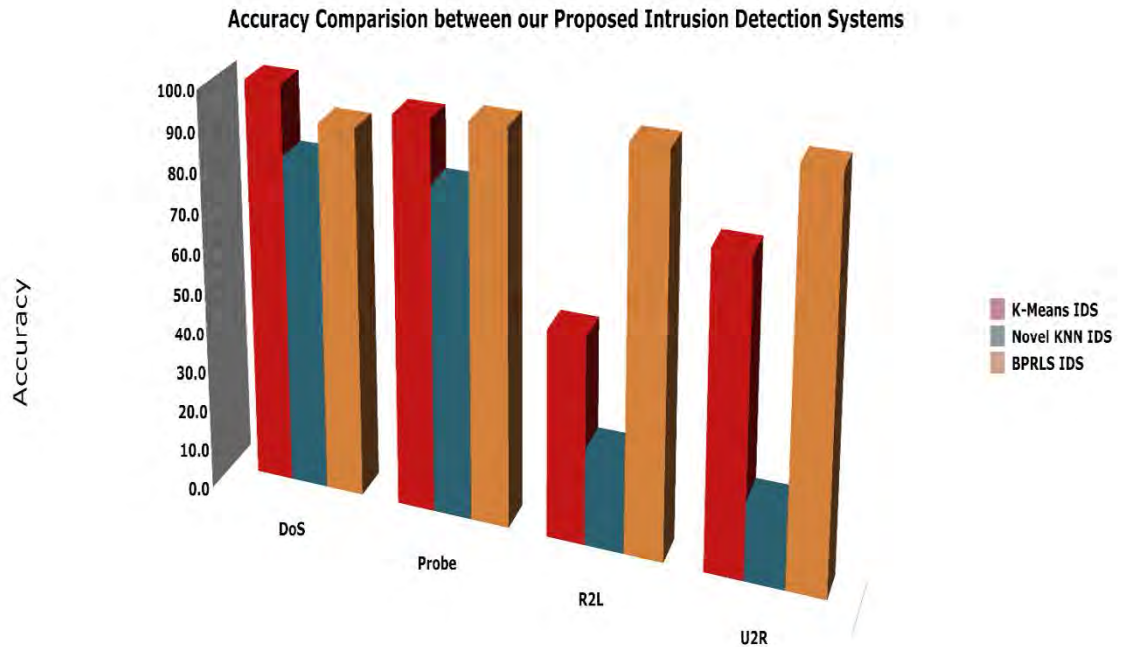


Figure 4.10: Graphical accuracy comparison between the intrusion detection systems.

4.7 Experimental Results Comparison between the Proposed System and Other Systems

This section dedicated for comparative analysis of the proposed system with former techniques. Figure (4.11) demonstrates the comparison of the proposed IDS system with the other state of art techniques, and shows the superiority of the proposed technique performance in the less frequency-attack.

The proposed technique performs efficiently in case of less frequent attacks: U2R and R2L. As shown in figure (4.11), it is apparent that our proposed technique has obtained a reliable peak accuracy values for both U2R and R2L.

In case of R2L and U2R intrusions, maximum accuracy value of 94.3% and 99.42% respectively were reached, when compared to other method.

We have attained 78.67% accuracy in case of Probe attack and it is good in compared with other method results. In case of DoS attack, we have 60.1% accuracy, but as the table (4.24) and table (4.28) illustrate, we have reached 94.64% and 95.52% detection rate for this type of attack in Training and Testing phase respectively.

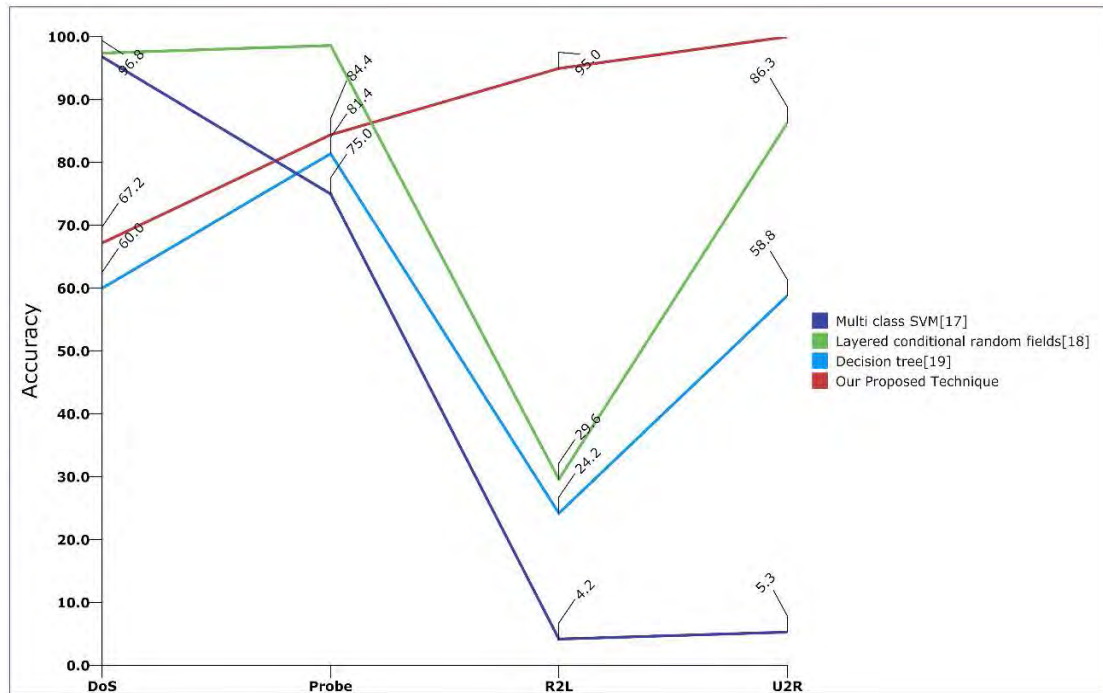


Figure 4.11: Graphical comparison between our proposed intrusion detection system and other proposed systems.

Chapter Five
Conclusions, Recommendations and
Future work

Chapter Five

Conclusions, Recommendations and Future work

5.1 Introduction

This chapter presents the conclusions, suggestions, a set of recommendations, and future work for further investigation and extensive research in this interesting field.

5.2 Conclusions and Results Discussion

The contribution to three different techniques is well demonstrated, since it have designed and implemented three intrusion detection systems; that are able to detect a wide variety of attacks. Two of the proposed systems belong to the clustering techniques and the third utilizes the neural network technique.

The classification capabilities of K-Means are more reliable than that of Novel KNN for all types of intrusions; which indicates that a hybrid system of K-Means and RLSBP ANN will be more efficient than a hybrid system composed of Novel KNN and RLSBP ANN.

For the evaluation metric calculation, False Positive, False Negative, True positive, and True Negative were used. In addition, for the evaluation results about 8% of the full huge NSL-KDD dataset were used; which amounted to about 10,000 connection records (data points).

The performance evaluation was made for both testing and training and for all types of attacks (DoS, Probe, U2R and R2L); This Thesis technique: recursive least squares back propagation IDS achieved better results; when compared with other existing techniques. Where it has attained about 100% and 94% accuracy for the least frequent attacks (U2R and R2L) and high detection rate reach up to 100% for both types of attacks.

Since stochastic nature of the neural networks in general, it is a common practice to report results of multiple experiments, and of multiple training and testing procedures, in this thesis we implemented the RLSBP in two experiments with different topology in each one. Although the attained classification results were better in the second experiment of higher nodes per layer

Despite of the computational complexity of this system, it is suitable for real time applications; because the run time is acceptable (elapsed time = 19.1 seconds) where this time was mostly due to the large number of training vectors of full features. Thus, the RLSBP IDS can operate as online classifier for different types of intrusions which has been trained for. The only factor that affect the neural network and makes it off-line, is the time that used to gather the necessary information that needed to compute features.

5.3 Recommendations and Future Work

The researcher author of this Thesis recommends strongly the following ideas:

1. It is an open project covering a broad range of subjects, and there are a lot of extended possibilities, the researcher can use any of available clustering techniques in association with RLBP ANN in a hybrid or hierarchal manner.
2. In order to avoid further complexity in the RLSBP neural network, one of clustering techniques can be used to make an initial classification of the connection records into two main general categories: normal and attack as a first step. Then, the records in each category can be further classified to their types of attacks.
3. NSL-data set were used with full features, however, features extraction is recommended as a pre-processing step; since it consists of two main phases:

features construction and features selection, which may enhance the overall effectiveness of IDS.

4. score normalization technique were used during the system implementation, however; for further investigation, we recommend the use of min-max normalization technique.
5. Further theoretical analysis is needed to find further optimality in choosing the number of layers, and the number of neurons per layer, to get better optimal performance for ANN.
6. Multilayer Recursive Least Squares Back Propagation ANN is one solution for reliable IDSs, However, it can look into some of the potential mathematical enhancements for these kinds of neural network, in such a way that can be employed for accuracy and detection rate enhancement .
7. Techniques for determining the optimal number of clusters (other than $K=5$) in the K-Means algorithm are needed since its performance is highly dependable on it.

References

- Jaiganesh, V., Sumathi, P., & Mangayarkarasi, S. (2013, February). An Analysis of Intrusion Detection System using back propagation neural network. ***In Information Communication and Embedded Systems (ICICES), 2013 International Conference on*** (pp. 232-236). [IEEE](#)
- Bottino, L. J. (2006, October). Security Measures in a Secure Computer Communications Architecture. ***In 25th Digital Avionics Systems Conference, 2006 IEEE/AIAA*** (pp. 1-18). [IEEE](#)
- Sharma, S. K., Pandey, P., Tiwari, S. K., & Sisodia, M. S. (2012, March). An improved network intrusion detection technique based on k-means clustering via Naïve bayes classification. ***In Advances in Engineering, Science and Management (ICAESM), 2012 International Conference on*** (pp. 417-422). [IEEE](#)
- Singh, N., & Chandra, N. (2014, April). Integrating Machine Learning Techniques to Constitute a Hybrid Security System. ***In Communication Systems and Network Technologies (CSNT), 2014 Fourth International Conference on*** (pp. 1082-1087). [IEEE](#).
- Tian, J., Gao, M., & Zhang, F. (2009, May). Network intrusion detection method based on radial basic function neural network. ***In E-Business and Information System Security, 2009. EBISS'09. International Conference on*** (pp. 1-4). [IEEE](#).

- Al-Sharafat,W.S.& Naoum, R. (2009, November). Significant of features selection for detecting network intrusions. ***In Internet Technology and Secured Transactions, 2009. ICITST 2009. International Conference for*** (pp. 1-6). [IEEE](#)
- Norouzian,M.R.,& Merati, S. (2011, February). Classifying attacks in a network intrusion detection system based on artificial neural networks. ***In Advanced Communication Technology (ICACT), 2011 13th International Conference on*** (pp. 868-873). [IEEE](#)
- Moradi, M., & Zulkernine, M. (2004, November). A neural network based system for intrusion detection and classification of attacks. ***In Proceedings of the 2004 IEEE international conference on advances in intelligent systems-theory and applications***, pp 12-44
- Jing, B., Xiaojing, C., & Kun, Z. (2010, March). A new method of data processing in the intrusion detection system with neural networks. ***In Education Technology and Computer Science (ETCS), 2010 Second International Workshop on*** (Vol. 2, pp. 343-345). [IEEE](#).
- Teknomo, K. (2006). ***K-means clustering tutorial***. *Medicine*, 100(4)
- Nazeer, KA Abdul, and M. P.(2009) Sebastian. "Improving the Accuracy and Efficiency of the k-means Clustering Algorithm." ***Proceedings of the World Congress on Engineering***. Vol. 1. PP 12-33.
- Jivani, A. G. (2013, January). The Novel k nearest Neighbor Algorithm. ***Computer Communication and Informatics (ICCCI), 2013 International Conference on*** (pp. 1-4). [IEEE](#).

- Na'mh, Z. (2012) *An Enhanced resilient backpropagation artificial neural network for intrusion detection system*. (MSc thesis). MEU: Amman, Jordan, PP 1-99.
- Scalero, R. S., & Tepedelenlioglu, N. (1992). A fast new algorithm for training feed forward neural networks. *Signal Processing, IEEE Transactions on*, 40(1), 202-210.
- Ham,F.(2001).*principles of neurocomputing for science and engineering*,(international edition).florida , PP 1-334.
- McClelland, J. L., Rumelhart, D. E., & PDP Research Group. (1986). Parallel distributed processing. *Explorations in the microstructure of cognition*, 2.
- Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., & Lang, K. J. (1989). Phoneme recognition using time-delay neural networks. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 37(3), 328-339
- Hu, W., Li, J., & Shi, J. (2006). Optimal *evaluation of feature selection in intrusion detection modeling*. *In Intelligent Control and Automation,. WCICA . The Sixth World Congress*, Vol. 2, PP. 5919-5922.
- Chakraborty, N.(2013) Intrusion Detection System and Intrusion Prevention System: A Comparative Study□. *IJCBB*, ISSN, 2229-6166.

- Yichun, P., Yi, N., & Qiwei, H. (2012, November). Research on Intrusion Detection System Based on IRBF. *In Computational Intelligence and Security (CIS), 2012 Eighth International Conference on* (pp. 544-548). [IEEE](#)
- Bulajoul, W., James, A., & Pannu, M. (2013, September). Network Intrusion Detection Systems in High-Speed Traffic in Computer Networks. *In e-Business Engineering (ICEBE), 2013 IEEE 10th International Conference on* (pp. 168-175). [IEEE](#)
- Scarfone, K., & Mell, P. (2007). Guide to intrusion detection and prevention systems (idps). *NIST special publication*, 800(2007), 94.
- Wankhade, K., Patka, S., & Thool, R. (2013, August). An efficient approach for Intrusion Detection using data mining methods. *In Advances in Computing, Communications and Informatics (ICACCI), 2013 International Conference on*(pp. 1615-1618). [IEEE](#).
- Khan, N. Y., Rauf, B., & Ahmed, K. (2010, February). Comparative study of intrusion detection system and its Recovery mechanism. *In Computer and Automation Engineering (ICCAE), 2010 The 2nd International Conference on*(Vol. 5, pp. 627-631). [IEEE](#)
- Elshoush, H. T., & Osman, I. M. (2011). Alert correlation in collaborative intelligent intrusion detection systems—A survey. *Applied Soft Computing*, 11(7), 4349-4365.
-

- Kezih, M., & Taibi, M. (2013, August). Evaluation effectiveness of intrusion detection system with reduced dimension using data mining classification tools. ***In Systems and Computer Science (ICSCS), 2013 2nd International Conference on*** (pp. 205-209). [IEEE](#).
- Chandrasekhar, A. M., & Raghuveer, K. (2013, January). Intrusion detection technique by using k-means, fuzzy neural network and SVM classifiers. ***In Computer Communication and Informatics (ICCCI), 2013 International Conference on*** (pp. 1-7). [IEEE](#).
- AL-Rashdan, W, Naoum, R, Al_Sharafat, W & Al-Khazaaleh, M. (2010). ***Novel network intrusion detection system using hybrid neural network (Hopfield and Kohonen SOM with conscience function)***. IJCSNS International Journal of Computer Science and Network Security, 10(11). PP11-55.
- Tavallaei, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set. ***In Proceedings of the Second IEEE Symposium on Computational Intelligence for Security and Defence Applications 2009***.
- Al-Janabi, S. T. F., & Saeed, H. A. (2011, December). A Neural Network Based Anomaly Intrusion Detection System. ***In Developments in E-systems Engineering (DeSE), 2011*** (pp. 221-226). [IEEE](#).
-

- Debar, H., Becker, M., & Siboni, D. (1992, May). A neural network component for an intrusion detection system. *In Research in Security and Privacy, 1992. Proceedings., 1992 IEEE Computer Society Symposium on* (pp. 240-250). [IEEE](#).
- Sharma, V., Rai, S., & Dev, A. (2012). A Comprehensive Study of Artificial Neural Networks. *International Journal*, 2(10).
- Naom, R.S. (2013). Lecture notes, Middle East University.
- Kukiela P., Kotulski Z. (2008) "Analysis of Different Architectures of Neural Networks For Applications in Intrusion Detection Systems", *International Multi conference on Computer Science and Information technology*, pp.807 – 811.
- Gu, R., Shen, F., & Huang, Y. (2013, October). A parallel computing platform for training large scale neural networks. *In Big Data, 2013 IEEE International Conference on* (pp. 376-384). [IEEE](#).
- Sathya, R., & Abraham, A. (2013). Comparison of Supervised and Unsupervised Learning Algorithms for Pattern Classification. *International Journal of Advanced Research in Artificial Intelligence*, 34-38.
- Riad, A. M., Elhenawy, I., Hassan, A., & Awadallah, N. (2013). VISUALIZE NETWORK ANOMALY DETECTION BY USING K-MEANS CLUSTERING ALGORITHM. *International Journal of Computer Networks & Communications*, 5(5).

- Wang, H., & Ma, R. (2009, March). Optimization of neural networks for network intrusion detection. *In Education Technology and Computer Science, 2009. ETCS'09. First International Workshop on* (Vol. 1, pp. 418-420). [IEEE](#).
- Mukhopadhyay, I., Chakraborty, M., Chakrabarti, S., & Chatterjee, T. (2011, December). Back propagation neural network approach to Intrusion Detection System. *In Recent Trends in Information Systems (ReTIS), 2011 International Conference on* (pp. 303-308). [IEEE](#).
- Norouzian, M. R., & Merati, S. (2011, February). Classifying attacks in a network intrusion detection system based on artificial neural networks. *In Advanced Communication Technology (ICACT), 2011 13th International Conference on* (pp. 868-873). [IEEE](#).
- Naoum, R. S., Abid, N. A., & Al-Sultani, Z. N. (2012). An Enhanced Resilient Backpropagation Artificial Neural Network for Intrusion Detection System. *IJCSNS*, 12(3), 11.
- Ambwani, T. (2003, July). Multi class support vector machine implementation to intrusion detection. In *Neural Networks, 2003. Proceedings of the International Joint Conference on* (Vol. 3, pp. 2300-2305). [IEEE](#).
- Gupta, K. K., Nath, B., & Kotagiri, R. (2010). Layered approach using conditional random fields for intrusion detection. *Dependable and Secure Computing, IEEE Transactions on*, 7(1), 35-49.

- Lee, J. H., Lee, J. H., Sohn, S. G., Ryu, J. H., & Chung, T. M. (2008, February). Effective value of decision tree with KDD 99 intrusion detection datasets for intrusion detection system. *In Advanced Communication Technology, 2008. ICACT 2008. 10th International Conference on* (Vol. 2, pp. 1170-1175). [IEEE](#).
- Hoque, M. S., Mukit, M., Bikas, M., & Naser, A. (2012). An implementation of intrusion detection system using genetic algorithm. *arXiv preprint arXiv:1204.1336*.
- Cleetus, N., & Dhanya, K. A. (2014, December). Genetic algorithm with different feature selection method for intrusion detection. *In Computational Systems and Communications (ICCSC), 2014 First International Conference on* (pp. 220-225). [IEEE](#).
- Benaicha, S. E., Saoudi, L., Guermeche, S. E. B., & Lounis, O. (2014, August). Intrusion detection system using genetic algorithm. In *Science and Information Conference (SAI), 2014* (pp. 564-568). [IEEE](#).
-