A Comparative Analysis of Intelligent Techniques for Detecting Anomalous Internet Traffic

BY

Abdul-Rahman Salah Mohammed Shaheen

A Thesis Presented to the

DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

COMPUTER ENGINEERING

SPTEMPER 2010

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by **ABDUL-RAHMAN SALAH MOHAMMED SHAHEEN** under the direction of his thesis adviser and approved by his thesis committee, has been presented to and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN COMPUTER ENGINEERING**.

Thesis Committee

Dr. Zubair Baig (Adviser)

Dr. Radwan E. Abdel-Aal (Member)

Dr. Sadiq M. Sait (Member)

Dr. Basem Al-Madani
Department Chairman

Dr. Salam A. Zummo
Dean of Graduate Studies

11/10/10
Date

Dedication

To the loving memory of my mother.

To my father and my step mother for their support and patience.

To all my brothers and sisters.

To my nephew Salah who brought joy to our family.

ACKNOWLEDGMENTS

All praise and glory are for Allah (swt), to him belongs the sovereignty of the heavens and the earth. May His blessings and mercies be upon the noblest of mankind, Muhammad (pbuh), his household, his companions and the generality of the true believers to the last day. I am grateful to Allah (swt) for all his favors bestowed on me since my birth, which indeed are innumerable, and the greatest of his bounties on me is being a Muslim.

I acknowledge the support provided by my parents, siblings and other relatives. My warmest appreciation goes to my thesis advisor, Dr. Zubair Baig, for his intellectual guidance, support and readiness to assist me in carrying out research work associated with this thesis. I appreciate the direction provided by him at every stage of this thesis. I also appreciate the guidance provided by Professor Radwan AbdelAal in formulating and implementing a GMDH-based approach for intrusion detection, which constitutes one of the components of the thesis. I would also like to appreciate the helpful suggestions and feedback provided by Professor Sadiq M. Sait.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

THESIS ABSTRACT

NAME: Abdul-Rahman Salah Mohammed Shaheen

TITLE OF STUDY: A Comparative Analysis of Intelligent Techniques for Detecting Anomalous Internet Traffic

MAJOR FIELD: Computer Engineering

DATE OF DEGREE: September 2010

Detecting anomalous traffic on the Internet has remained an issue of concern for the community of security researchers over the years. The advances in the area of computing performance, in terms of processing power and storage, have fostered their ability to host resource-intensive intelligent algorithms, to detect intrusive activity, in a timely manner. As part of this thesis, we study and analyze the performance of two such machine learning algorithms, namely, Average One-Dependency Estimators (AODE) and Abductive networks (GMDH), when implemented as part of an intrusion detection system, to detect anomalies on a dataset of Internet traffic activity. The simulation results obtained for the two approaches are compared and analyzed based on several performance metrics.

Keywords: *IDS, Anomalous traffic, Average One Dependency Estimators (AODE), Group Method for Data Handling (GMDH), Feature Selection.*

ملخص الرسالة

الاسم: عبدالرحمن صلاح محمد شاهين

عنوان الرسالة: مقارنة تحليلية للطرق الذكية للكشف عن الهجمات على شبكة الإنترنت

التخصص: هندسة الحاسب الآلي

تاريخ التخرج: سبتمبر ٢٠١٠

يبقى الكشف عن الهجمات الجديدة والمستحدثة المتنوعة على شبكة الإنترنت مسألة مثيرة للقلق بالنسبة للباحثين في مجال الأمن على مر السنين. وقد عززت أوجه التقدم في مجال أداء الحوسبة، من حيث قوة المعالجة والتخزين، وقدرتها على استضافة الخوارزميات الذكية الكثيفة الاستخدام للموارد، للكشف هذه الهجمات الجديدة، في الوقت المناسب .وكجزء من هذه الرسالة، قمنا بدراسة وتحليل أداء اثنين من خوارزميات التعلم الآلي، وهي واحدة متوسط التبعية المقدرون (AODE) وشبكات Abductive (GMDH) ، عند تنفيذها كجزء من نظام لكشف التسلل الى الأنظمة الحاسوبية ، وللكشف عن الهجمات المستحدثة على شبكة الإنترنت. وفي النهاية تم مقارنة نتائج المحاكاة التي حصل عليها وتحليلها بناء على مجموعة من مقاييس الأداء.

CHAPTER 1

INTRODUCTION

With the rapid proliferation of Internet traffic and convenient availability of open source tools to launch malicious attacks, the challenge for network intrusion detection has increased manifold over the past few years. A simple attack launched by a novice can cause huge financial losses to an organization. Detecting such attacks is an important first step in securing a computer network. In general, malicious attacks against computer networks can be categorized into four types - Denial of Service (DoS), Probing, User to Root (U2R), and Remote to Local (R2L) attacks.

In the case of Distributed Denial of Service attacks, machines participating in launching the attack do not rely on a protocol vulnerability, but rather prevent legitimate users from using network resources, by sending high volumes of traffic toward the victim machine. As a result, traffic will aggregate from different streams at the victim, causing the victim to be incapacitated. These malicious machines will keep sending high rates of traffic purposely regardless of time outs.

However, legitimate users who uses a standard implementation of the TCP protocol with slow start implementation will suffer timeouts that will affect their congestion window, which will throttle their sending capacity. As a result, legitimate traffic will be proportionally low as compared to the attack traffic during an attack time.

As can be voted, fast detection of such attacks leads to a faster contisenus which will help the victim to trigger appropriate prevention measures that mitigate the effects of the attack. Attacks victims ranges from a single machine to an entire infrastructure network. The attacks range from Teardrop attacks to denial of service attacks on the network infrastructure. The latter case is the most critical. Denial of service attacks can result in interruption of businesses or even government services which can effects millions of clients.

Intrusion Detection Systems (IDS) have gained significance due to their ability to defend computer networks against the continuous evolution of various types of attacks. Network-based IDS' detect these attacks by monitoring ingoing and outgoing traffic in order to detect the existence of possible outliers in the traffic patterns, where outliers can be anomalous traffic. Intrusion detecion systems can be categorized into two types - *a*) Anomaly detectors, which detect deviations of network traffic behavior from predefined normal traffic profiles, and *b*) Misuse detectors, which attempt to find signatures of malicious patterns known to the IDS beforehand [1]. Anomaly detection techniques define the normal profile of traffic, and attempt to track deviations from that profile to classify them as intrusions.

False alarms can have very high cost depending on the defense mechanism used by the victim network. For example, if the victim network uses client puzzles as a defense mechanism, clients will not be able to connect to the network unless they solve cryptographic puzzles. These puzzles are designed to take enough CPU time as a proof of work that the client is sincere in need for the service [2]. Malicious and non-malicious clients should solve the puzzle before getting access to the service. This will allow the server to serve under DoS attacks at the cost of degraded service. So as you can see false alarm can lead to degrade the service [2].

Signature based intrusion detection systems use a database of intrusion signatures for intrusion detection. Because of this, they can not detect novel intrusions, whose signatures do not exist in the signatures database. As a result, to keep up with intrusions, they need a frequent update of the signatures database. Using a signature database makes these types of intrusion detection systems attractive in terms of accuracy and speed, where the accuracy is measured in terms of less number of false positives [3, 4].

Anomaly based intrusion detection systems can detect novel attacks by generalization of learning rules. they profile the normal behavior of network traffic and detect deviations from that normal behavior to classify them as intrusions. Anomaly based IDS use machine learning and statistical information used to detect the existence of attacks. With the ability to generalize rules from learned data, anomaly based IDS perform generalization of attacks and fault tolerance to imprecise and uncertain information. Examples of such approaches include Sup-

port Vector Machines SVMs, Naïve Bayesian, Decision Trees, Neural Networks and Genetic Programming [3, 4]. For this thesis, my research will focus on the measurement of performance of anomaly based intrusion detection systems, when two specific algorithms are applied, namely, AODE and GMDH.

1.1 Research Motivation

The motivation for this work lies in the lack of an existing mechanism for rapid and intelligent detection of anomalies in network traffic. It is intended to propose two machine learning-based approaches for detecting network intrusions, and to compare and contrast simulation results acquired when the two schemes are implemented and tested. The resulting analysis may pave inroads towards a new breed of intrusion detection systems that will possibly be adaptable at a large scale for effective and efficient detection.

1.2 Thesis Objectives

The main objective of this thesis is to explore the use of an alternative technique of abductive networks, namely, the group method of data handling (GMDH) algorithm [4][5], and the use of Averaged One-Dependence Estimators (AODE), for detecting anomalous network traffic. The results of the two methods are compared with each other and with results documented from previously proposed intelligent intrusion detection systems.

1.3 Thesis Contributions

To achieve the thesis objectives, the contributions of this thesis are listed as follows:

- Review the existing literature on the use of intelligent techniques for anomalous traffic detection.
- Propose a GMDH-based and an AODE-based technique for intrusion detection.
- Implement the two techniques using software-based tools and perform a simulation.
- Compare the results of the proposed approaches in terms of defined performance metrics.

1.4 Thesis Organization

The thesis is organized as follows. Chapter 2 gives an overview of intelligent approaches and techniques used in anomalous traffic detection. In Chapter 3, different approaches were explored for selecting the most representative features of network traffic based on several defined criteria. An implementation of a GMDH-based IDS discussed in Chapter 4. The results acquired from simulation are analyzed. Subsequently, (Chapter 5) discusses the AODE-based anomalous traffic detection technique, with simulation and analysis. The thesis finally concludes

in Chapter 6, where the overall comparison of the two proposed approaches is elaborated upon, and a list of potential improvements and suggestions for future work are provided.

CHAPTER 2

ARTIFICIAL INTELLIGENCE FOR INTRUSION DETECTION

In this chapter, *Anomalous Traffic Detection* techniques used in the literature are summarized. In addition, applications of neural networks in anomalous traffic detection from literature are also summarized.

2.1 Multilayer Perceptron

Multilayer Perceptron (MLP) is a supervised learning algorithm, which uses a feed-forward structure to solve classification problems. MLP neural networks are trained by manipulating the weights of the neural network connections. The network weights are updated by using different functions during the training period, such as the gradient-based optimization algorithm. The most common MLP training functions are shown in Table 8. When the network converges to the local minima of error, the *output layer* of the network will show the expected result

when *data* is fed into the *input layer*. Faraoun and Boukelif [5] propose a hybrid method of k-means algorithm and MLP. They use the k-means algorithm to group the input data into a number of clusters, which in their case is 20 (based on the number of attacks provided in KDD99). The distances between the centers of clusters and input data points are calculated, and only the most discriminating samples that cover the maximum the region of each class, are selected for the learning process. The selected samples are then presented to the MLP network for classification into four classes of attacks, namely, DoS, Probing, U2R, and R2L.

An MLP for misuse detection is proposed in two configurations by Cannady [6]. The first configuration is a stand alone and the second configuration uses a rule-based expert system. The proposed scheme uses nine traffic features as input to the MLP, which means the input layer should contain nine neurons. The MLP network consists of three layers: the input layer with nine neurons, the hidden layer and the output layer with two neurons, with all layers being fully connected. The Sigmoid function is used as a transfer function between the neurons. The author uses 10,000 data points, with 1,000 of them used for testing, and the remaining for training.

2.2 Self-Organizing Maps

A Self- Organizing Maps (SOM) is an unsupervised learning algorithm to group similar data into clusters in the input space. It is a data visualization technique

Function	Description
Traingd	Basic gradient descent. Slow response, can be used in incremental mode training.
Traingdm	Gradient descent with momentum. Generally faster than traingd. Can be used in incrementa mode training.
Traingdx	Adaptive learning rate. Faster training than traingd, but can only be used in batch mod training.Resilient back propagation. Simple batch mode training algorithm with fast convergence an minimal storage requirements.
Trainrp	Fletcher-Reeves conjugate gradient algorithm. Have smallest storage requirements of th conjugate gradient algorithms.
Traincgf	Polak-Ribiere conjugate gradient algorithm. Slightly larger storage requirements than traincgf.Faster convergence on some problems.
Traincgp	Powell-Beale conjugate gradient algorithm. Slightly larger storage requirements than traincgp.
Traincgb	Generally faster convergence.
Trainscg	Scaled conjugate gradient algorithm. The only conjugate gradient algorithm that requires n line search. A very good general purpose training algorithm.
Trainbfg	BFGS quasi-Newton method. Requires storage of approximate Hessian matrix and has mor computation in each iteration than conjugate gradient algorithms, but usually converges i fewer iterations.
Trainoss	One step secant method. Compromise between conjugate gradient methods and quasi-Newto methods.
Trainlm	Levenberg-Marquardt algorithm. Fastest training algorithm for networks of moderate size. Ha memory reduction feature for use when the training set is large.

Table 2.1: MLP training functions [7]

that produces a low dimensional topological map to help people to understand the original high dimensional data. Once the neural network is trained, *the map converges* to a stationary distribution and *shows a clear separation between normal traffic and attack traffic*. The output neurons are considered as the counts for normal and attack traffic points. After building the map using training data, future connections can be quickly classified as normal or attacks based on their location in map. Kayacik et al. [8, 9], Depren et al. [10], and DeLooze [9] used SOM in their IDS research.

2.3 Emergent Self-Organizing Maps

Kohonens Emergent Self-Organizaing Maps belong to Kohonens Self-Organizing maps (KSOMs) which has its base in biology. It is also called a *winner-take-all* unsupervised learning neural network. They are unsupervised because there is no target vector which requires the administrator to label the clusters into normal cluster and attack clusters. This approach has advantage of combining machine learning and visualization techniques. However, KSOMs has limited number of neurons in order of tens which is not enough for huge traffic analysis. Emergent Self-Organizing Maps solve this limitation. They produce topological maps that illustrate the input data according to their similarity [11]. The map will represent the network traffic in data points in clusters which help to classify it into normal or anomalous depending on the position of its best match cluster. Valleys will have the data points belong to same class of traffic. Borders

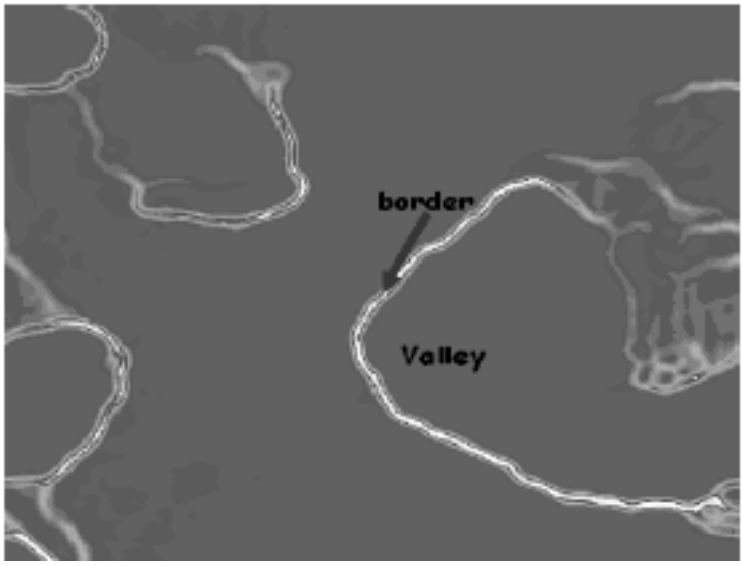


Figure 2.1: ESOM Map Demo
[9, 11]

will have some points that can be classified to the nearest matched valley. ESOM sample map is shown in Figure 2.1

2.4 The Random Neural Network

The Random Neural Network (RNN) model was introduced by Gelenbe [12, 13]. It is used successfully for a wide range of applications. It comes in two architectures, namely, feed-forward or a fully recurrent architecture. RNNs have *strong generalization capabilities*, even when training data set is relatively small compared to the actual testing data. The model also achieves fast learning due to its computational simplicity for the weight updating process. RNN was used by Oke et al in [14] for DDoS attack detection. RRN was used in conjunction with statistical variables like maximum likelihood, Hurst parameter and Entropy.

Hurst parameter gives network traffic *self-similarity* while Entropy shows how much data is contained in the traffic, that differentiates significantly between normal traffic and anomalous traffic [14].

2.5 Bayesian Learning

It is a supervised generative learning technique that uses probabilistic models and prior knowledge in classification of an observation. Bayesian integration does not suffer from the overfitting problem, according to [15]. Prior knowledge can be incorporated naturally and all uncertainty is manipulated in a consistent manner [15]. Moreover it is possible to learn model structures and readily compare between model classes. There are different classes of BN classifiers includes Nave-Bayes, Tree augmented Nave-Bayes (TANs), Bayesian network augmented Nave-Bayes (BANs), Bayesian multi-nets and general Bayesian networks (GBNs). It is used extensively in text categorization in general and studied extensively in SPAM detection. However, it was used rarely in intrusion detection even though it has very good generalization ability. Bayesian networks suffer from assuming independence between the events which make it less accurate. Many proposals were proposed to solve this problem including LBR and Super-Parent TAN. Although, these techniques demonstrate high error performance, they suffer from high computational cost. A new approach, with accuracy performance comparable to LBR and Super-Parent TAN, solved the problem of independence by averaging all of constrained class of classifiers. AODE solves also the problem of

high computation of its precedents. AODE has low variance and is suitable for incremental learning [16].

2.6 GMDH

This method was proposed in 1968 by Prof. Ivakhnenko at the Institute of Cybernetics in Kiev. Group method of data handling (GMDH) is a family of *inductive algorithms* for computer-based mathematical modeling of multi-parametric datasets that automatically synthesize optimized polynomial network structures. The most popular base function used in GMDH is the gradually complicated Kolmogorov-Gabor polynomial:

$$y(x_1, x_2, ..., x_n) = a_0 + \sum_{i=1}^{n} a_i x_i + \sum_{i=1}^{n} \sum_{j=i}^{n} a_{ij} x_i x_j + \sum_{i=1}^{n} \sum_{j=i}^{n} \sum_{k=j}^{n} a_{ijk} x_i x_j x_k + ... \quad (2.1)$$

The resulting structure is a feed-forward structure network, *aka* polynomial neural network, because of its multi-layered structure, with each node having a polynomial activation function.

GMDH uses certain criterion to describe the model requirement. For e.g, the criterion of regularity, and Criterion of Unbiasedness. Usually, the criterion is calculated using different part of data that have not been used for coefficient estimation.

2.7 Summary

Due to the high computation overhead and low detection rates, the intelligent techniques defined in this chapter do not provide for a strong intrusion detection mechanism. GMDH and AODE have not been studied yet. Due to their respective tendencies to correlate data in different manners, we postulate that their application to an IDS holds promise. We therefore introduce our proposed IDS', based on these two techniques, in the following chapters.

CHAPTER 3

DATA SET PREPROCESSING

3.1 The Dataset

NSL-KDD is a dataset proposed by Tavallaee et al. to solve some of the problems of the KDD'99 data set which are mentioned in [17, 18]. NSL-KDD dataset is a reduced version of the original KDD-99 dataset. NSL-KDD consists of the same features as KDD-99. Table 3.1 shows each feature with its type and description. The KDD-99 dataset consists of 41 features and one class attribute. The class attribute has 21 classes that fall under the previously defined four types of attacks: Probe attacks, User to Root (U2R) attacks, Remote to Local (R2L) attacks and Denial of Service attacks. This dataset has a binary class attribute. Also, it has a reasonable number of training and test instances which makes it practical to run the experiments on.

Table 3.1: NSL-KDD Original Feature Description.

Feature	Description	Type
1 duration	length (no. of seconds) of the connection	continuous
2 protocol_type	type of the protocol	discrete
3 service	network service on the destination	discrete
4 flag	status flag of the connection	discrete
5 src_bytes	no. of data bytes from source to destination	continuous
6 dst_bytes	no. of data bytes from destination to source	continuous
7 land	1 if connection is from/to the same host/port; 0 otherwise	discrete
8 wrong_fragment	no. of wrong fragments	continuous
9 urgent	no. of urgent packets	continuous
10 hot	no. of hot indicators	continuous
11 num_failed_logins	no. of failed logins	continuous
12 logged_in	1 if successfully logged in; 0 otherwise	discrete
13 num_compromised	no. of compromised conditions	continuous

Continued on next page

Table 3.1 –NSL-KDD Original Feature Description – continued

Feature	Description	Type
14 root_shell	1 if root shell is obtained; 0 otherwise	continuous
15 su_attempted	1 if su root command attempted; 0 otherwise	continuous
16 num_root	no. of root accesses	continuous
17 num_file_creations	no. of file creation operations	continuous
18 num_shells	no. of shell prompts	continuous
19 num_access_files	no. of operations on access control files	continuous
20 num_outbound_cmds	no. of outbound commands in an ftp session	continuous
21 is_host_login	1 if the login belongs to the hot list; 0 otherwise	discrete
22 is_guest_login	1 if the login is a guest login; 0 otherwise	discrete
23 count	no. of connections to the same host as the current connection in the past two seconds	continuous
24 srv_count	no. of connections to the same service as the current connection in the past two seconds	continuous

Continued on next page

Table 3.1 –NSL-KDD Original Feature Description – continued

Feature	Description	Type
25 serror_rate	% of connections that have SYN errors	continuous
26 srv_serror_rate	% of connections that have SYN errors	continuous
27 rerror_rate	% of connections that have REJ errors	continuous
28 srv_rerror_rate	% of connections that have REJ errors	continuous
29 same_srv_rate	% of connections to the same service	continuous
30 diff_srv_rate	% of connections to different services	continuous
31 srv_diff_host_rate	% of connections to different hosts	continuous
32 dst_host_count	count of connections having the same destination host	continuous
33 dst_host_srv_count	count of connections having the same destination host and using the same service	continuous

Continued on next page

Table 3.1 –NSL-KDD Original Feature Description – continued

Feature	Description	Type
34 dst_host_same_srv_rate	% of connections having the same destination host and using the same service	continuous
35 dst_host_diff_srv_rate	% of different services on the current host	continuous
36 dst_host_same_src_port_rate	% of connections to the current host having the same src port	continuous
37 dst_host_srv_diff_host_rate	% of connections to the same service coming from different hosts	continuous
38 dst_host_serror_rate	% of connections to the current host that have an S0 error	continuous
39 dst_host_srv_serror_rate	% of connections to the current host and specified service that have an S0 error	continuous
40 dst_host_rerror_rate	% of connections to the current host that have an RST error	continuous
41 dst_host_srv_rerror_rate	% of connections to the current host and specified service that have an RST error	continuous

3.2 Nominal features

GMDH-type algorithms deal only with numeric features. This limitation is inherited from the regression based nature of GMDH. As can be seen from Table 3.1, all features are numeric except three features, namely, protocol_type, service and flag. These features need to be converted into numeric values, so that the GMDH algorithm can be applied on this dataset. This can be done using feature transformation, by replacing the $k-valued$ features with k-binary features. Each feature takes a binary value 0 or 1, to represent the existence of a value. We used $Weka_1$ NominalToBinary filter to get a dataset with features compatible with the inputs of GMDH AIM networks. The resulting dataset was found to contain 123 numeric features. This number is reduced to 75, by the use of feature selection algorithms, as described in Section 3.4.

3.3 Discretization of numeric features

Averaged One Dependence Estimators, on the other hand, can not deal with numeric features. AODE can deal only with nominal attributes. As a result, we need to discretize numeric values of the dataset. For sake of comparison with GMDH, the 75 features dataset used by GMDH algorithm are discretized, which is accomplished using the supervised version of the attribute discretize filter in Weka. There are two discretization approaches implemented in Weka. The first approach is to discretize features without using the knowledge of class values. This approach is called unsupervised discretization. The other approach, on the other

hands, uses class knowledge for discretization. This approach is called supervised discretization. Supervised discretization uses equal frequency binning with some conservations. It sacrifices the equal frequency property to preserve class separation in each bin. We used supervised discretization algorithm because it helps to improve the performance of intrusion detection systems.

3.4 Feature Selection

Not every feature is relevant to the intrusion detection task, as they do not contribute much to the process. Redundant features are defined as features correlated with one or more other features. A key problem is how to choose the most relevant features which will be used for intrusion detection. Since not every feature of the training data may be relevant to the detection task and, in the worse case, irrelevant features may introduce noise and redundancy into the design of intrusion detection systems, choosing a good subset of features is critical to improve the performance of such systems.

Strategies used in selecting the most prominent features can be categorized into two main types: *filter* methods and *wrapper* methods. Filter methods filter the features to create the most prominent feature subset before the start of learning process. On the other hand, wrapper methods incorporate learning techniques in selection of the most prominent feature subset. This makes the results of wrapper methods more optimized for classification algorithm because

of incorporation of feature selection in the algorithm [19–21]. In this chapter, we will use both approaches to select the most prominent attributes. Feature selection algorithms were used to meet the constraints of the AbTech ModelQuest Prospector abductive network synthesizer of 75 features. In addition, they were also used to rank these selected features.

3.4.1 Wrapper Method

Abductive Network

For the Wrapper method, we followed the procedure proposed by Abdel-Aal et. al in [22]. Feature ranking using abductive networks does so based on the predictive quality of the data, based on the following steps:

1. Change setting of model synthesis to select three inputs at a time. This method is used to restrict number of selected features to three which will help in ranking the features in groups of maximum size three.

2. Remove selected features to force the model to select from less predictive remaining features.

3. Repeat the process until all features are selected or no further features can be selected. Change model complexity in steps from small to large, if needed, to force the modeler to select remaining features.

After feature ranking we do feature selection, by applying the top ranked features one by one as long as the accuracy of selected model is non-decreasing.

We stop when the accuracy drops, as an indication of model overfitting. We follow this procedure at different levels of model complexity, numbers of layers, and numbers of inputs.

3.4.2 Filter Method

The features are filtered to create the most prominent features subset before the start of learning process. Filter methods use heuristics for feature ranking, such as information gain heuristic.

Information Gain

Information gain is used to rank attribute individually on basis of separating the classes of the training examples. Attribute rank can be calculated using information gain with respect to class using the following formula:

$$Information\ gain = (D_x) - (D_{-x}) \tag{3.1}$$

where, D_x is the information which includes attribute x, and D_{-x} is information which excludes attribute x. The value of D_{-x} is calculated as the average of each value that this particular attribute can take.

The information itself is calculated using the entropy equation:

$$entropy = \sum_{k=1}^{n} p_k log p_k \tag{3.2}$$

We used this method for ranking NSL-KDD-99 features to select the most

prominent 75 features in order to overcome the restriction of the AbTech ModelQuest Prospector abductive network synthesizer features. We have 70 services and we want to reduce them into 22 services to get a total of 75 features.

Chi-square

Chi-square is a statistical method used to rank features individually on the basis of separating the classes of the training examples. Attribute ranks can be calculated in the following way:

1. Create a table that calculates frequencies of values of attributes under each class.

2. Calculate the expected value (EV)for each attribute, as follows:

$$EV = \frac{(class\ count\ for\ that\ value\ *\ values\ count\ for\ that\ class)}{total\ number\ of\ instances} \quad (3.3)$$

3. Calculate:

$$ChiVal(attribute) = \sum_{k=1}^{v} \sum_{l=1}^{c} ChiCell(expected_{k,l}) \quad (3.4)$$

, where v is the values count for the attribute and, c is the class count.

4. ChiCell calculated as following:

$$diff * diff/expected \tag{3.5}$$

where

$$diff = expected - valuefrequency \tag{3.6}$$

We used this method for ranking NSL-KDD-99 features based on ChiVal in Equation 3.4 to select the most prominent 75 features in order to overcome the restriction of the AbTech ModelQuest Prospector abductive network synthesizer of 75 features. We have 70 services and we want to reduce them into 22 services to obtain a total of 75 features. Table 3.4 shows the selected services.

Information gain ratio

Information gain ratio is a modification of information gain that solves the problem of its bias toward features which have many values. For example, if a dataset contains the serial numbers of customers, then the information gain of the customer serial number will be high, and it will be used at the high level in decision trees. This bias degrades the ability of learning algorithms, such as decision trees, of generalization of new customers because the serial number will be considered on the top of decision tree, which does not have any indicator to generalize with.

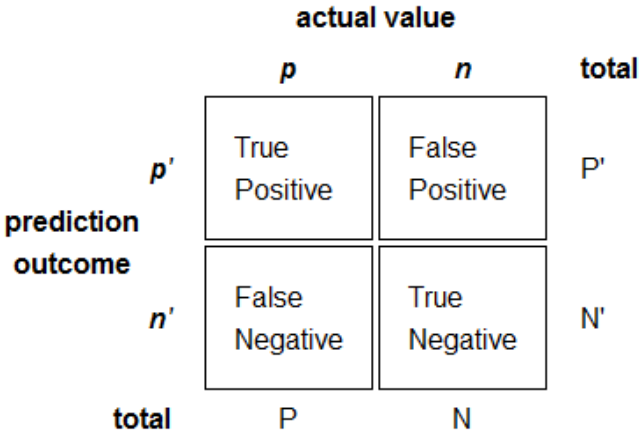


Table 3.2: Confusion Matrix

Information gain ratio corrects the information gain by taking the intrinsic information of using that attribute as split into account.Intrinsic information is the entropy of distribution of instances values.

Gain Ratio is large when the data is evenly spread and is small when the data has one value.

Gain Ratio is calculated as following:

$$Gain\ Ratio(" Feature") = \frac{gain(" Atrribute")}{intrinsic\ info(" Atrribute")} \tag{3.7}$$

$$intrinsic\ info(" Atrribute") = -\sum \frac{\mid S_i \mid}{\mid S \mid} * log_2 \frac{\mid S_i \mid}{\mid S \mid} \tag{3.8}$$

3.5 Performance Measures

Performance of classifiers in Artificial Intelligence is measured using several metrics. The confusion matrix is a visualization tool for tabulating the overall

performance of the classifier, as shown in Table 3.2. Each row of the matrix represents the instances in a predicted class, while each column represents the instances in an actual class. The following simple measures are derived from confusion matrix from Table 3.2, and will be used in evaluating our classifiers:

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP} \tag{3.9}$$

$$Recall = truepositiverate = \frac{TP}{TP + FN} \tag{3.10}$$

$$Precision = \frac{TP}{TP + FP} \tag{3.11}$$

$$Specificity = \frac{TN}{TN + FP} \tag{3.12}$$

$$Detection\ rate = \frac{total\ number\ of\ detected\ attacks}{Total\ Number\ of\ attacks} * 100\% \tag{3.13}$$

where:

TP: true positive is the number of positives classified as positive.

FP: false negative is the number of negatives classified as positive.

TN: true negative is the number of negatives classified as negative.

FN: false negative is the number of positives classified as negative.

3.5.1 ROC graphs

Receiver Operator Characteristic (ROC) graphs were developed during WWII to statistically model false positive and false negative detections of radar operators.

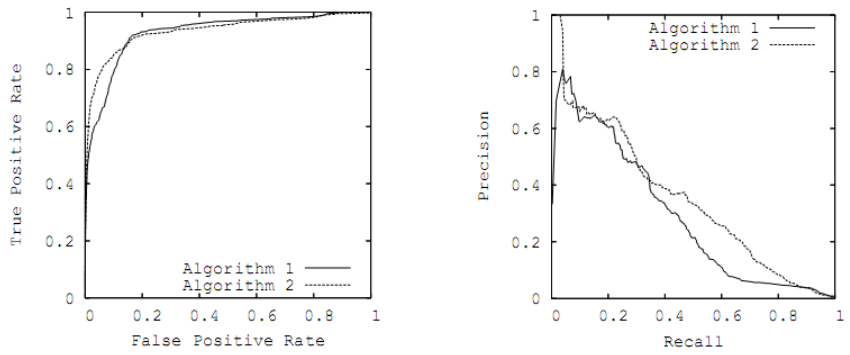
It is a standard graph for measurements used in medicine and biology. ROC curves are generated by plotting outputs of classifiers over different threshold values. Each threshold value will result in one point in the ROC curve. Each point represents different trade off (cost values) between false positives and false negatives. The ROC area represents performance averaged over all possible cost ratios [23, 24].

The ROC area quantifies classifier performance as follows:

- 1.0: perfect prediction
- 0.9: excellent prediction
- 0.8: good prediction
- 0.7: mediocre prediction
- 0.6: poor prediction
- 0.5: random prediction
- less than 0.5: poor prediction.

3.5.2 Precision-Recall graphs

Precision-Recall (PR) curves, often used in Information Retrieval [25, 26], have been cited as an alternative to ROC curves for tasks with *a large skew in the class distribution* [27, 28]. An important difference between ROC space and PR space is the visual representation of the curves. Looking at PR curves can expose



(a) Comparison in ROC space

(b) Comparison in PR space

Figure 3.1: ROC and PR curves for performance measurements [23]

differences between algorithms that are not apparent in ROC space. Sample ROC curves and PR curves are shown in Figures 3.1(a) and 3.1(b) respectively. These curves, taken from the same learned models on a highly-skewed cancer detection dataset, highlight the visual difference between these spaces. The goal in ROC space is to be in the upper-left-hand corner, and when one looks at the ROC curves in Figure 3.1(a) they appear to be fairly close to optimal. In the PR space, algorithm 2 shows remarkable improvement over algorithm 1. In the PR space the goal is to be in the upper-right-hand corner, and the PR curves in Figure 3.1(b) show that there is still a vast room for improvement [23, 24].

3.6 Simulation and Results

Simulation on the NSL-KDD dataset described in Section 3.1 was performed. First, the 22 most prominent services using information gain, gain ratio and chi-square feature ranking methods described in Section 3.4 was selected. The selected

service features are shown in Table 3.4. The resulting dataset consists of 75 features that satisfy the constraint limit on the maximum number of input features of the GMDH AIM tool.

By following the procedure for feature ranking explained in section 3.4.1, we get an ordered list of features and the accuracy of the models as shown in Table 3.3. The top ranked features are then used to build models by adding one feature at a time. We developed 65 abductive network models with 4 layers and different CPM. Figure 3.3 shows the accuracy of the synthesized models at different number of input features with diffident CPM values. Figure 3.3 shows that no overfitting occurs when the number of network layers is limited to 4 layers (the default setting of AIM tool). However, this figure shows that the accuracy of the synthesized models improves at certain points. The overfitting does not occur which shows that the synthesized networks are not complex enough to fit the data and there is room for improvement. So, we run the simulation again with the number of network layers set to five and by using different CPM values. Figure 3.2 shows that the model starts overfitting when the $k = 14$ and stabilizes when $k = 20$.

From previous simulations, we get the best selected features that will be used in next two chapters. We select the features that selected at each model that gives better performance than previous models and combine them with the selected top k features resulting using a five layers network. The top selected features were found to be "protocol=icmp","Flag=SF", "same_srv_rate", "loged_in", "dst_host_same_srv_rate", "dst_host_same_src_port_rate". "protocol=icmp" helps

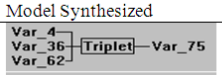
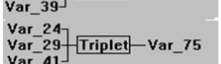
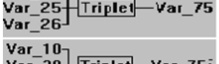
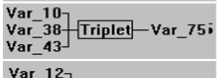
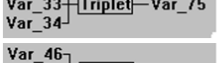
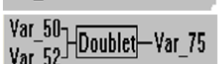
Step	Excluded Features	Model Synthesized	Accuracy
1	None		0.929642
2	4, 36, 62		0.929642
3	4, 36, 62, 45, 67, 69		0.90983
4	4, 36, 62, 45, 67, 69, 66, 72, 73		0.8857
5	4, 36, 62, 45, 67, 69, 66, 72, 73, 59, 71, 74		0.911354
6	4, 36, 62, 45, 67, 69, 66, 72, 73, 59, 71, 74, 58, 60, 70		0.89205
7	4, 36, 62, 45, 67, 69, 66, 72, 73, 59, 71, 74, 58, 60, 70, 14, 32, 61		0.887478
8	4, 36, 62, 45, 67, 69, 66, 72, 73, 59, 71, 74, 58, 60, 70, 14, 32, 61, 2, 19, 56		0.904496
9	4, 36, 62, 45, 67, 69, 66, 72, 73, 59, 71, 74, 58, 60, 70, 14, 32, 61, 2, 19, 56, 57, 63, 64		0.87427
10	4, 36, 62, 45, 67, 69, 66, 72, 73, 59, 71, 74, 58, 60, 70, 14, 32, 61, 2, 19, 56, 57, 63, 64, 3, 65, 68		0.605283
11	4, 36, 62, 45, 67, 69, 66, 72, 73, 59, 71, 74, 58, 60, 70, 14, 32, 61, 2, 19, 56, 57, 63, 64, 3, 65, 68, 11, 20, 28		0.544069
12	4, 36, 62, 45, 67, 69, 66, 72, 73, 59, 71, 74, 58, 60, 70, 14, 32, 61, 2, 19, 56, 57, 63, 64, 3, 65, 68, 11, 20, 28, 1, 31, 39		0.549911
13	4, 36, 62, 45, 67, 69, 66, 72, 73, 59, 71, 74, 58, 60, 70, 14, 32, 61, 2, 19, 56, 57, 63, 64, 3, 65, 68, 11, 20, 28, 1, 31, 39, 24, 29, 41		0.540513
14	4, 36, 62, 45, 67, 69, 66, 72, 73, 59, 71, 74, 58, 60, 70, 14, 32, 61, 2, 19, 56, 57, 63, 64, 3, 65, 68, 11, 20, 28, 1, 31, 39, 24, 29, 41, 6, 18, 23		0.542799
15	4, 36, 62, 45, 67, 69, 66, 72, 73, 59, 71, 74, 58, 60, 70, 14, 32, 61, 2, 19, 56, 57, 63, 64, 3, 65, 68, 11, 20, 28, 1, 31, 39, 24, 29, 41, 6, 18, 23, 5, 25, 26		0.537719
16	4, 36, 62, 45, 67, 69, 66, 72, 73, 59, 71, 74, 58, 60, 70, 14, 32, 61, 2, 19, 56, 57, 63, 64, 3, 65, 68, 11, 20, 28, 1, 31, 39, 24, 29, 41, 6, 18, 23, 5, 25, 26, 10, 38, 43		0.543815
17	4, 36, 62, 45, 67, 69, 66, 72, 73, 59, 71, 74, 58, 60, 70, 14, 32, 61, 2, 19, 56, 57, 63, 64, 3, 65, 68, 11, 20, 28, 1, 31, 39, 24, 29, 41, 6, 18, 23, 5, 25, 26, 10, 38, 43, 12, 16, 17		0.541529
18	4, 36, 62, 45, 67, 69, 66, 72, 73, 59, 71, 74, 58, 60, 70, 14, 32, 61, 2, 19, 56, 57, 63, 64, 3, 65, 68, 11, 20, 28, 1, 31, 39, 24, 29, 41, 6, 18, 23, 5, 25, 26, 10, 38, 43, 12, 16, 17, 7, 9, 21		0.538989
19	4, 36, 62, 45, 67, 69, 66, 72, 73, 59, 71, 74, 58, 60, 70, 14, 32, 61, 2, 19, 56, 57, 63, 64, 3, 65, 68, 11, 20, 28, 1, 31, 39, 24, 29, 41, 6, 18, 23, 5, 25, 26, 10, 38, 43, 12, 16, 17, 7, 9, 21, 8, 13, 22		0.534417
20	4, 36, 62, 45, 67, 69, 66, 72, 73, 59, 71, 74, 58, 60, 70, 14, 32, 61, 2, 19, 56, 57, 63, 64, 3, 65, 68, 11, 20, 28, 1, 31, 39, 24, 29, 41, 6, 18, 23, 5, 25, 26, 10, 38, 43, 12, 16, 17, 7, 9, 21, 8, 13, 22, 15, 37, 55		0.527305
21	4, 36, 62, 45, 67, 69, 66, 72, 73, 59, 71, 74, 58, 60, 70, 14, 32, 61, 2, 19, 56, 57, 63, 64, 3, 65, 68, 11, 20, 28, 1, 31, 39, 24, 29, 41, 6, 18, 23, 5, 25, 26, 10, 38, 43, 12, 16, 17, 7, 9, 21, 8, 13, 22, 15, 37, 55, 30, 33, 34		0.529337
22	4, 36, 62, 45, 67, 69, 66, 72, 73, 59, 71, 74, 58, 60, 70, 14, 32, 61, 2, 19, 56, 57, 63, 64, 3, 65, 68, 11, 20, 28, 1, 31, 39, 24, 29, 41, 6, 18, 23, 5, 25, 26, 10, 38, 43, 12, 16, 17, 7, 9, 21, 8, 13, 22, 15, 37, 55, 30, 33, 34, 46, 48, 49		0.526289
23	4, 36, 62, 45, 67, 69, 66, 72, 73, 59, 71, 74, 58, 60, 70, 14, 32, 61, 2, 19, 56, 57, 63, 64, 3, 65, 68, 11, 20, 28, 1, 31, 39, 24, 29, 41, 6, 18, 23, 5, 25, 26, 10, 38, 43, 12, 16, 17, 7, 9, 21, 8, 13, 22, 15, 37, 55, 30, 33, 34, 46, 48, 49, 50, 52		0.526797
24	4, 36, 62, 45, 67, 69, 66, 72, 73, 59, 71, 74, 58, 60, 70, 14, 32, 61, 2, 19, 56, 57, 63, 64, 3, 65, 68, 11, 20, 28, 1, 31, 39, 24, 29, 41, 6, 18, 23, 5, 25, 26, 10, 38, 43, 12, 16, 17, 7, 9, 21, 8, 13, 22, 15, 37, 55, 30, 33, 34, 46, 48, 49, 50, 52, 40, 47, 51		0.526797
25	4, 36, 62, 45, 67, 69, 66, 72, 73, 59, 71, 74, 58, 60, 70, 14, 32, 61, 2, 19, 56, 57, 63, 64, 3, 65, 68, 11, 20, 28, 1, 31, 39, 24, 29, 41, 6, 18, 23, 5, 25, 26, 10, 38, 43, 12, 16, 17, 7, 9, 21, 8, 13, 22, 15, 37, 55, 30, 33, 34, 46, 48, 49, 50, 52, 40, 47, 51, 35		0.526797
26	4, 36, 62, 45, 67, 69, 66, 72, 73, 59, 71, 74, 58, 60, 70, 14, 32, 61, 2, 19, 56, 57, 63, 64, 3, 65, 68, 11, 20, 28, 1, 31, 39, 24, 29, 41, 6, 18, 23, 5, 25, 26, 10, 38, 43, 12, 16, 17, 7, 9, 21, 8, 13, 22, 15, 37, 55, 30, 33, 34, 46, 48, 49, 50, 52, 40, 47, 51, 35, 42, 44, 54		

Table 3.3: Feature Selection Using GMDH.

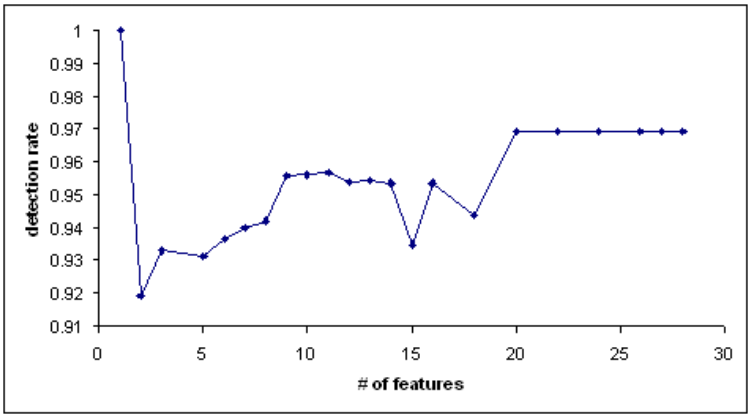


Figure 3.2: Feature selection detection rate vs. number of features used, to build abductive network models with 5 layers

in discriminating between normal traffic and DoS ping flood attack traffic. "Flag=SF" shows that the connection has been created and terminated normally, which helps in defining normal traffic profile and detecting the deviation from normal profiles. "same_srv_rate" shows the percentage of connections that were made to the same service, among the connections aggregated in count. "same_srv_rate" helps in predicting exhaustive attacks. "logged_in" has a great discrimination power in predicting R2L attacks."dst_host_same_srv_rate" gives percentage of connections from clients to the same service in the same host. "dst_host_same_srv_rate" and "dst_host_same_src_port_rate" give the percentage of connections to the current host having the same src port. "dst_host_same_src_port_rate" helps in predicting if there is a possible DDoS attack or not. "hot" indicators can be used to monitor if any hidden directories are created during the FTP session, which belongs to remote-to-local (R2L) attacks. All selected features are from the top 30 features agreed upon by the GMDH,Chi-

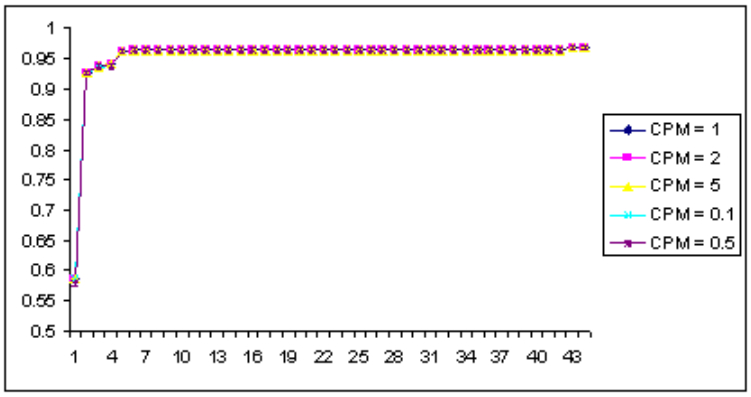


Figure 3.3: Feature selection accuracy vs. number of features used, to build abductive network models

square, information gain and gain ratio. Except for "hot" which is selected based on the GMDH feature selection simulation.

Service	**Information Gain**	**Gain Ratio**	**Chi-Square**
http	0.2658512	0.29388	39832.01107
private	0.1697134	0.25498	25477.78802
domain_u	0.0675209	0.18124	8445.63123
smtp	0.0410314	0.12836	5678.2804
eco_i	0.022086	0.09792	3474.78694
ecr_i	0.0188803	0.114	2834.18276
ftp_data	0.0064218	0.02105	1074.36367
Z39_50	0.0075894	0.12853	996.92344
uucp	0.006864	0.12635	901.49764
courier	0.0064574	0.12506	848.0208
bgp	0.0062454	0.12437	820.13544
whois	0.0060952	0.12387	800.38977

Continued on next page

Table 3.5 Feature ranking to select the most prominent services

Service	Information Gain	Gain Ratio	Chi-Square
uucp_path	0.0060599	0.12375	795.74451
iso_tsap	0.0060423	0.12369	793.42199
imap4	0.005465	0.11765	734.08515
nnsp	0.005539	0.12194	727.26134
vmnet	0.0054243	0.12153	712.1805
ctf	0.004948	0.11975	649.57046
csnet_ns	0.0047893	0.11913	628.71242
supdup	0.0047804	0.1191	627.55382
discard	0.0047275	0.11889	620.60258
http_443	0.004657	0.11861	611.3353
daytime	0.0045777	0.11829	600.91102
gopher	0.0045513	0.11818	597.4366
efs	0.0042605	0.11697	559.22886
systat	0.00419	0.11667	549.96941
link	0.0041724	0.11659	547.65473
exec	0.0041636	0.11655	546.49742
hostnames	0.0040403	0.11601	530.29699
name	0.003961	0.11566	519.88433
urp_i	0.0041202	0.09424	515.41122
mtp	0.0038553	0.11518	506.00311
domain	0.0033385	0.08008	502.73541
echo	0.0038113	0.11498	500.22006

Continued on next page

Table 3.5 Feature ranking to select the most prominent services

Service	Information Gain	Gain Ratio	Chi-Square
klogin	0.0038025	0.11494	499.0635
login	0.0037673	0.11478	494.43746
ldap	0.0036	0.11399	472.46779
netbios_dgm	0.003556	0.11378	466.68741
time	0.0029305	0.06252	462.50033
sunrpc	0.0033448	0.11274	438.94796
netbios_ssn	0.0031776	0.11188	416.99508
netstat	0.00316	0.11179	414.68464
netbios_ns	0.0030457	0.11118	399.66855
finger	0.0021392	0.0201	368.3909
kshell	0.0026236	0.1088	344.25146
nntp	0.0025973	0.10864	340.78929
ssh	0.0024276	0.09733	336.88638
auth	0.0018888	0.02938	319.60455
sql_net	0.0021491	0.10577	281.9578
telnet	0.0011589	0.00865	202.25324
IRC	0.0012728	0.07912	159.32225
ntp_u	0.0012061	0.08227	146.45907
rje	0.0007537	0.09232	98.84794
pop_2	0.0006835	0.09124	89.64708
remote_job	0.0006835	0.09124	89.64708
printer	0.0006046	0.08992	79.29752

Continued on next page

Table 3.5 Feature ranking to select the most prominent services

Service	Information Gain	Gain Ratio	Chi-Square
other	0.0004133	0.0019	71.57628
shell	0.0003911	0.06131	58.49029
X11	0.0002957	0.04185	43.11474
pop_3	0.0001825	0.00842	30.716
urh_i	0.0000717	0.05998	8.70687
red_i	0.0000574	0.05873	6.96538
pm_dump	0.0000438	0.06869	5.74328
aol	0	0	0
ftp	0	0	0
harvest	0	0	0
http_2784	0	0	0
http_8001	0	0	0
tftp_u	0	0	0
tim_i	0	0	0

3.7 Summary

In this chapter, we elaborated upon the dataset used for the implementation and testing of the GMDH-based and AODE-based intrusion detection systems. We ranked and selected features based on four different feature selection algo-

Selected Services	Differentiation Ability
http	sweep, neptune
private	ipsweep, neptune, portsweep, satan, teardrop
domain_u	satan
Z39_50	neptune and portsweep.
smtp	ipsweep, portsweep, neptune and satan
uucp	portsweep and satan
courier	portsweep and neptune
bgp	neptune
whois	ipsweep, neptune and portsweep
uucp_path	neptune and portsweep
iso_tsap	neptune.00
nnsp	neptune
vmnet	neptune, portsweep and satan
ctf	neptune and nmap scan
csnet_ns	neptune and portsweep
supdup	neptune and portsweep
discard	neptune and satan
http_443	neptune and portsweep
daytime	neptune and portsweep
gopher	ipsweep and neptune
imap4	imap and neptune
efs	neptune and portsweep
systat	neptune and portsweep
link	neptune, ipsweep and portsweep
exec	neptune

Table 3.4: Selected services with differentiation ability between normal and anomalous traffic.

GMDH	chi-sequare	Information gain	Information Gain Ratio
4	38	38	36
36	39	39	32
62	63	63	45
45	62	62	59
67	36	66	58
69	66	36	71
66	67	67	72
72	68	68	39
73	45	71	62
59	71	45	14
71	56	72	63
74	58	58	38
58	72	59	19
60	59	56	11
70	32	32	66
14	14	70	70
32	70	14	67
61	65	65	68
2	69	69	41
19	19	19	20
56	64	64	56
57	57	57	23
63	74	74	24
64	11	11	16
3	73	73	6
65	60	60	17
68	61	61	10
11	3	20	25
20	20	3	5
28	4	4	21

Table 3.6: Top ranked 30 features using diffrent ranking algorithms.

rithms. The best selected features were subsequently tabulated, and their importance to intrusion detection was explained. Different evaluation measures for intrusion detection systems, such as ROC and PR curves, were also elaborated upon.

CHAPTER 4

GMDH-BASED INTRUSION DETECTION

Abductory inductive mechanism (AIM) is a powerful supervised inductive learning tool for automatically synthesizing neural network models from a database of input and output values [23]. The model emerging from the AIM synthesis process is a *robust and compact transformation* implemented as a layered abductive network of feed-forward functional elements as shown in Figure 4.1. The potential for GMDH in anomalous traffic has not been previously explored in the literature. However, compared to other neural networks and learning tools, this method offers the advantages of *faster model development* requiring little user intervention, *faster convergence* during model synthesis without the issue of getting trapped in local minima, *automatic selection of relevant input variables*, and *automatic configuration of the generated model structure*.

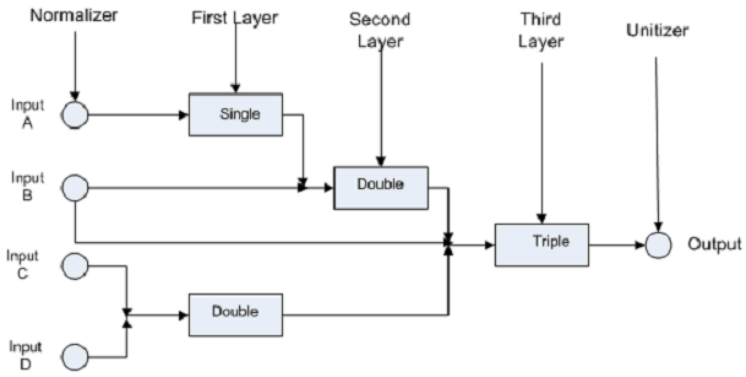


Figure 4.1: Abductive network operating with three layers, and various functional elements [29].

4.1 Abductive machine learning

The abductive machine learning approach is based on the self organizing group method of data handling (GMDH) [30]. The GMDH approach is a proven concept for iterated polynomial regression that can generate polynomial models in effective predictors. The iterative process involves using initially defined simple regression relationships, to derive more accurate representations in the next iteration in an evolutionary manner.

The algorithm selects the polynomial relationships and the input combinations that minimize the *prediction error*, during each iteration. This prevents exponential growth in the number of polynomial models generated. Iterations are stopped automatically at a point in time, when a balance between model complexity for accurate fitting of the training data, and model simplicity that allows it to generalize new data accurately, is achieved.

In the classical GMDH-based approach abductive network models are con-

structed by the following 6 steps [30]:

1. Separating the original data into training data and testing data.

2. Generating the combinations of the input variables in each layer.

3. Calculating the partial descriptors

4. Selecting optimum descriptors

5. Iteration until stopping criteria is met

The algorithm has three main components: Representation, Selection and Stopping. AIM uses PSE for selection and stopping, to avoid overfitting. Figure 4.2 shows the relationship that is represented through the following equations:

$$PSE = FSE + KP \tag{4.1}$$

$$KP = CPM * \frac{2K}{N} * S_p^2 \tag{4.2}$$

where:

PSE is the predicted squared error.

FSE is the fitting squared error on training data.

CPM is the complexity penalty multiplier.

N number of samples in training set.

S_p is the prior estimate for the variance of dependent variable.

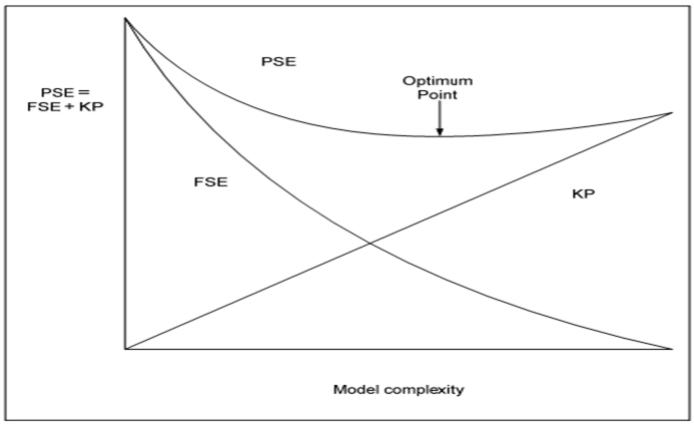


Figure 4.2: Relationship between FSE,PSE and KP [29].

4.2 AIM Functional Element

AIM supports the following functional elements:

- **Normaliser**: Transforms the original input into a normalized variable having a mean of zero and a variance of unity. The normaliser is represented as follows:

$$y = z_0 + z_1 x \tag{4.3}$$

where x is the input variable and y is the normalized input, while z_0 and z_1 are the coefficients.

- **Unitizer**: Converts the range of the network outputs to a range with the mean and variance of the output values used to train the network, which in other words maps the output of GMDH to the input problem space.

- **Single Node**: The single node has only one input and the polynomial equation is limited to the third degree, i.e.

$$y = z0 + z_1 x + z_2 x^2 + z_3 x^3 \tag{4.4}$$

where x is the input to the node, y is the output of the node and z_0, z_1, z_2 and z_3 are the node coefficients.

- **Double Node**: The double node takes two inputs and the third-degree polynomial equation includes cross term so as to consider the interaction

between the two inputs, i.e.

$$y = z_0 + z_1 x_i + z_2 x_j + z_3 x_i^2 + z_4 x_j^2 + z_5 x_i x_j + z_6 x_i^3 + z_7 x_j^3 \tag{4.5}$$

where x_i, x_j are the inputs to the node, y is the output of the node and z_0, z_1, z_2 and z_7 are the node coefficients.

- **Triple Node**: Similar to the single and double nodes, the triple node with three inputs has a more complicated polynomial equation allowing the interaction among these inputs.

4.3 Abductive Network Ensemble

Network ensemble is a learning approach where different networks cooperate to provide a solution to the same problem. These networks are trained on different portions of the training data set, with different input features, or different architectures. As a result, higher recognition rates are achievable because with diversity in decision making the error may cancel out. The final output may be based on one of two criteria:

1. **Simple majority vote:**

   The final output is simply the majority vote of ensemble network members.

2. **Simple averaging of ensemble network members:**

The final output is the average of all network ensemble member outputs, as follows:

$$z_i = \frac{1}{n} \sum_{i=1}^{n} y_i \tag{4.6}$$

4.4 Simulation and Results

In this section we simulate our proposed GMDH-based intrusion detection system. From the resulting ROC curves, the best threshold/cut off value was found to be 0.422. At this point on the curve, a higher true positive rate (TPR) is achieved, versus the lowest possible false positive rate (FPR).

4.4.1 Monolithic abductive models

For monolithic abductive models, the simulation was performed using all features from the training set, and different CPM values, i.e. , $CPM = 0.1, 0.5, 1, 2, 5$. The results for this simulation are shown in Table 4.1. In Figure 4.3, the ROC curve for all models using different CPM values are shown. From the ROC curves, it is evident that they all yield similar detection rates. The areas under the curve are almost equal to 99% for all cases. Figure 4.4 shows the Precision-Recall curves for different abductive networks synthesized using different model complexity values. These curves are almost the same for models with CPM = 0.1 and 5. The other models with CPM = 0.5, 1 and 2 are the same. They slightly outperform previous models in terms of recall.

We also ran the simulation with the first 14 and 20 GMDH top ranked features,

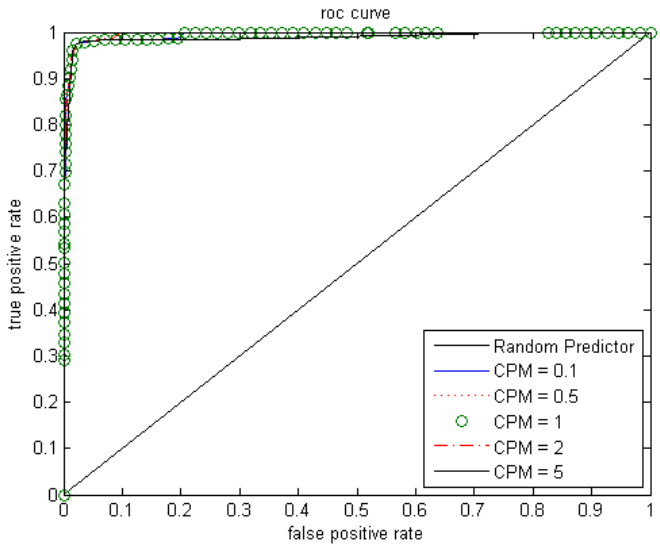


Figure 4.3: Comparison of the ROC curve for five abductive network classifiers: when $CPM = 0.1$, $CPM = 0.5$, $CPM = 1$,$CPM = 2$ and $CPM = 5$.

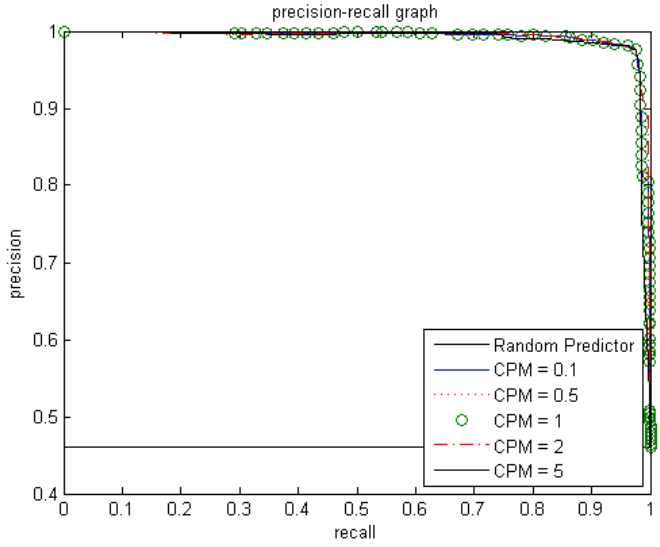


Figure 4.4: Comparison of the Precision-Recall curve for five abductive network classifiers: the optimum monolithic model when $CPM = 0.1$, $CPM = 0.5$, $CPM = 1$,$CPM = 2$ and $CPM = 5$.

FN	TN	FP	TP	TP	FAR	DR	CPM
57	2769	65	2358	0.022	0.976	0.1	
57	2769	65	2358	0.022	0.976	0.5	
60	2777	57	2355	0.020	0.975	1	
56	2773	61	2359	0.021	0.976	2	
57	2769	65	2358	0.022	0.976	5	

Table 4.1: Performance results of different abductive network models synthesized using different CPM values.

FN	TN	FP	TP	FAR	DR	No. Of Features
112	2753	81	2303	0.028	0.953	14
74	2775	59	2341	0.020	0.969	20

Table 4.2: Performance results of two abductive network models synthesized using 14 and 20 top ranked GMDH features.

CPM = 1, and the number of layers as 4. The results were not as good as using the full feature set, but were comparable to an extent. However, it was noticed that the training time for this case was much less than when using the full feature set. Table 4.2 shows the results of using the top 14 and 20 features, respectively. In Figures 4.5 and 4.6, we show ROC and PR curves for abductive network models synthesized using first 14 and 20 features respectively.

Subsequently, we ran the simulation using the best feature set, i.e. features $4, 19, 36, 62, 67, 69, 70, 72, 55$. The results were better than running the training over the entire dataset in terms of both the training time and the accuracy. Table 4.3 shows the results of monolithic abductive network models synthesized using the full training set and 38% randomly selected instances from the training set.

CPM	Accuracy	Specificity	Sensitivity	TP	FP	TN	FN
1	0.9779	0.9785	0.9772	2360	61	2773	55
1	0.9783	0.9774	0.9793	2365	64	2770	50

Table 4.3: Performance results of different abductive network models synthesized using the best features and CPM = 1, trained on a) full training set, and b) 38% random portion of the data.

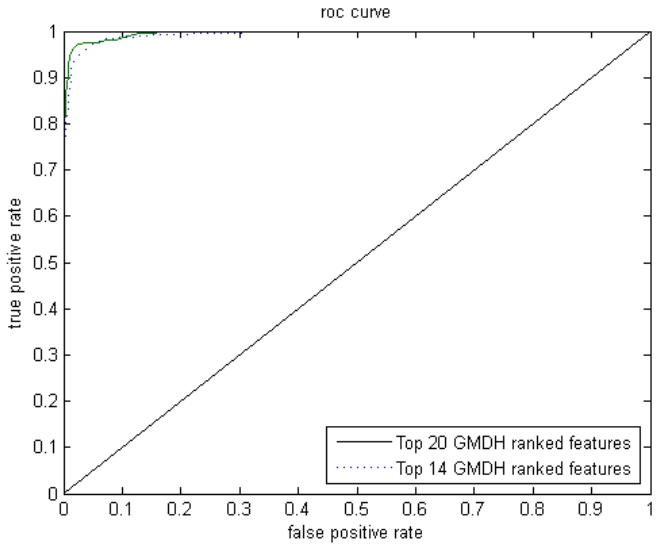


Figure 4.5: Comparison of the ROC curve for two abductive network classifiers synthesized using 14 and 20 top ranked GMDH features.

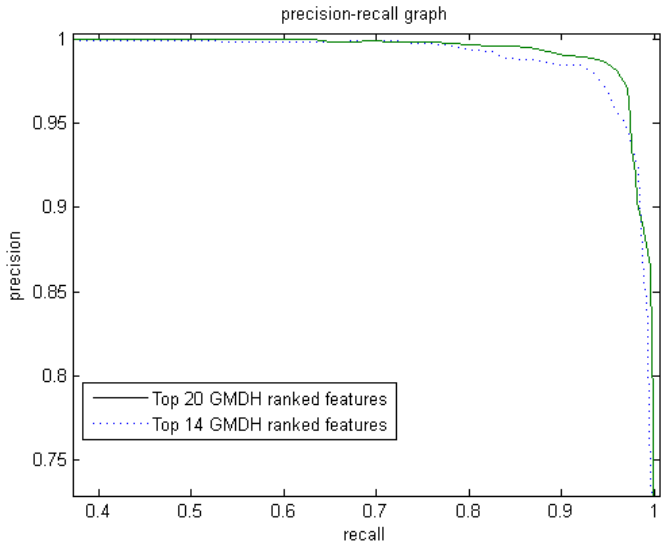


Figure 4.6: Comparison of the Precision-Recall curve for two abductive network classifiers synthesized using 14 and 20 top ranked GMDH features.

FN	TN	FP	TP	FAR	DR	Model
60	2777	57	2355	0.020	0.975	Monolithic
73	2786	48	2342	0.016	0.970	Averaged ensemble
63	2773	61	2352	0.021	0.973	Vote Ensemble

Table 4.4: Performance results of ensemble network individual models synthesized using model complexity of 1.

4.4.2 Ensemble abductive models

From the above simulation, we observed that the best model was synthesized for $CPM = 1$. So, we choose this value, and built an ensemble network of three different abductive network models, using $CPM = 1$, and all features. The results of the ensemble network were compared both in terms of ROC as well as Precision-Recall curves. The comparison ROC curve is shown in Figure 4.7. The curve indicates a slightly improved result when the ensemble network is used. The area under the curve for ensemble is 0.9963, whereas for the monolithic, it is 0.993. Table 4.4 shows the performance measures of each committee member network. Precision-Recall curves in Figure 4.8 shows that ensemble networks are always better than monolithic models, except for certain threshold values, when the precision of monolithic is higher than ensemble networks, albeit with the same recall values.

4.4.3 Abductive networks using Feature Selection

After performing simulation runs with all features selected, we synthesized abductive networks using the top 14 selected features, from the different feature selection algorithms studied in Chapter 3. The resulting networks are compared using ROC and Precision-Recall curves, as shown in Figures 4.9 and 4.10. It

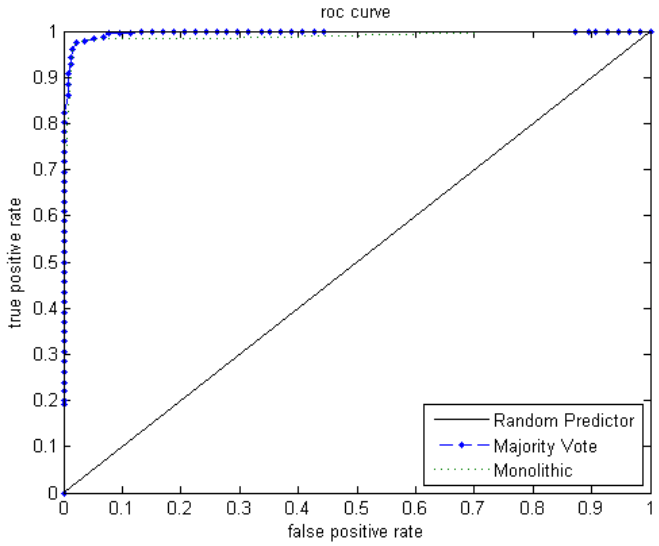


Figure 4.7: Comparison of the ROC curve for two abductive network classifiers: the optimum monolithic model when CPM = 1 and 3 member network ensemble using simple averaging.

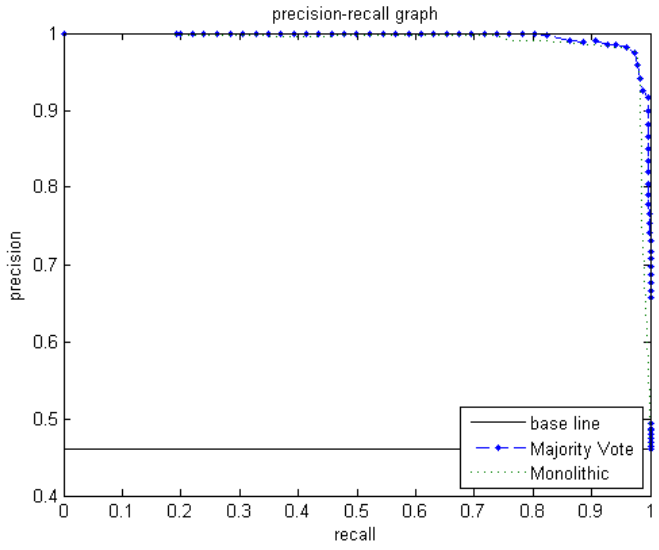


Figure 4.8: Comparison of the Precision-Recall curve for two abductive network classifiers: the optimum monolithic model when CPM = 1 and 3 member network ensemble using simple averaging.

FN	TN	FP	TP	FAR	DR	Model
112	2753	81	2303	0.028	0.953	GMDH-GMDH
152	2712	122	2263	0.043	0.937	GMDH-IG
59	2602	232	2356	0.082	0.975	GMDH-GR

Table 4.5: Performance results of different network models synthesized using top 14 features selected using different feature selection algorithms

can be observed here that abductive networks synthesized using the GMDH top-ranked features outperform abductive networks synthesized using the entire feature set, and different ranking methods. The abductive networks synthesized using gain ratio top-ranked features outperform abductive networks synthesized using top features of information gain feature ranking method. The PR curve shows that networks synthesized using information gain have better recall values vs. the some precision values. However, this is not important because it does not occur in the best area of PR graph.

The area under the curve for the abductive network model synthesized using the top 14 GMDH features is 0.993, whereas the area under the curve for the abductive network model synthesized using the top 14 Gain Ratio features is 0.990, and the area under the curve using the top 14 Information Gain features is 0.983. Table 4.5 shows the performance measures of each committee member network.

4.5 Summary

In this chapter, we provided a detailed description of the GMDH implementation of the intrusion detection system. Results of simulation of the monolithic

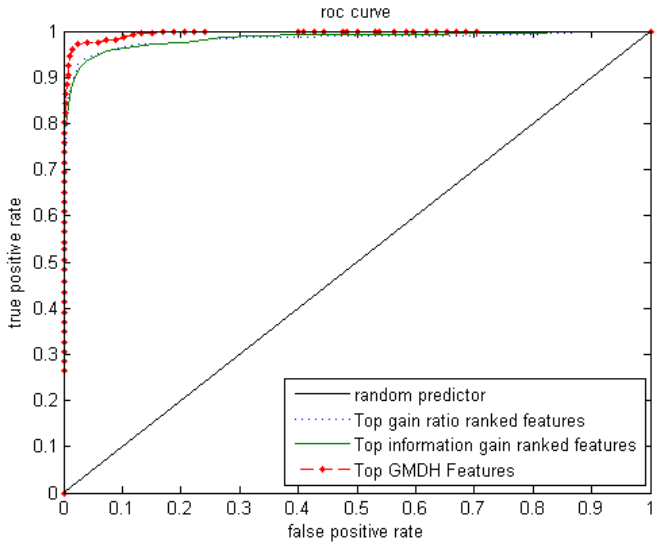


Figure 4.9: Comparison of the ROC curve for three abductive network classifiers: network model synthesized using top 14 GMDH ranked features, top 14 GR ranked features and top 14 IG ranked features.

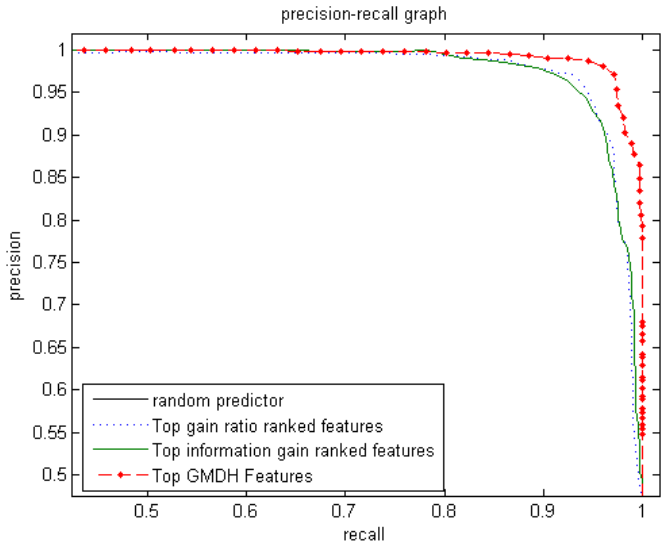


Figure 4.10: Comparison of the PR curve for three abductive network classifiers: network model synthesized using top 14 GMDH ranked features, top 14 GR ranked features and top 14 IG ranked features.

abductive network models were discussed. Subsequently, the results of simulation of ensemble abductive network models were discussed. Thirdly, the results of simulation of abductive network models synthesized using top selected features, were elaborated upon. Ensemble networks using simple averaging combiner shows slight improvement over monolithic networks. The feature selection proved to yield higher detection rates, and quicker training times.

CHAPTER 5

AVERAGED ONE-DEPENDENCE ESTIMATOR-BASED INTRUSION DETECTION

Naïve Bayesian has been applied successfully in many fields of machine learning. Its simplicity and high classification accuracy has attracted many researchers in diverse fields to use it for their classification applications. With its success, it still suffers from the attribute independence problem. Many proposals to alleviate this issue, while retaining the simplicity of Naïve bayesian classification have been proposed in the literature. Zheng et. al. proposed lazy Bayesian rules to alleviate the attribute independence problem of Naïve Bayesian classifier [31]. Lazy Bayesian Rules create a conjunctive rule to select the most significant subset of

training examples and then induce the Naïve Bayesian classifier using this subset. This method outperforms Naïve Bayesian in terms of minimizing bias, at the cost of increasing the variance and classification time. Figure 5.1 [32, 33] shows a visual example of trade-offs between bias and variance. Keogh et. al. proposed a method to alleviate the attribute independence problem of Naïve Bayesian by allowing each attribute to depend on a maximum of one other attribute besides the class attribute [34]. Their initial proposal was called One Dependence Estimator (ODE). However, their approach suffered from high delays for model selection because of the need to check all $m*(m-1)$ alternatives of m attribute dependencies, while building the model (during the training phase). Each model is evaluated using leave one out validation method, which is time consuming as well. It is an evaluation method that runs n times on a data set of n instances by using n-1 training instances and 1 test instance. Keogh et al. proposed another method that overcomes the delay associated with ODE [34]. Their method is called the Super Parent One Dependence Estimator (SPODE). SPODE solves some of the problems of ODE, by allowing all attributes to depend only on one other attribute called the super parent. There are two configurations of SPODE, namely, Full SPODE and Partial SPODE. Partial SPODE outperforms full SPODE because in reality not all attributes depend on the same attribute. However, recognizing some dependencies makes it outperform Naïve Bayes.

AODE eliminates model selection problem in ODE and SPODE by averaging all full SPODEs.

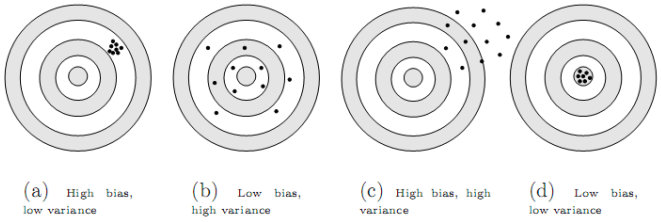


Figure 5.1: Bias and Variance [33].

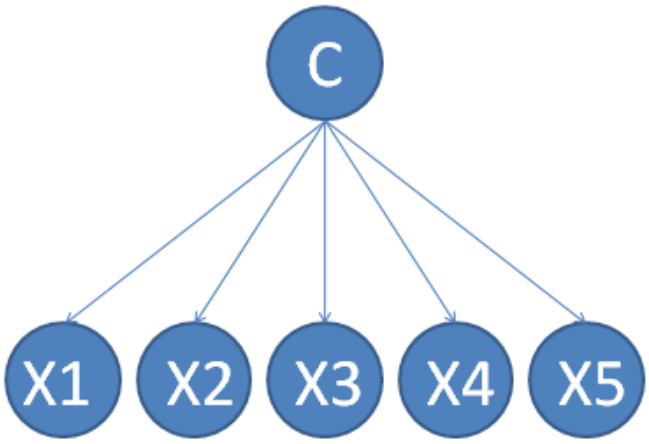


Figure 5.2: Naïve Bayesian [35].

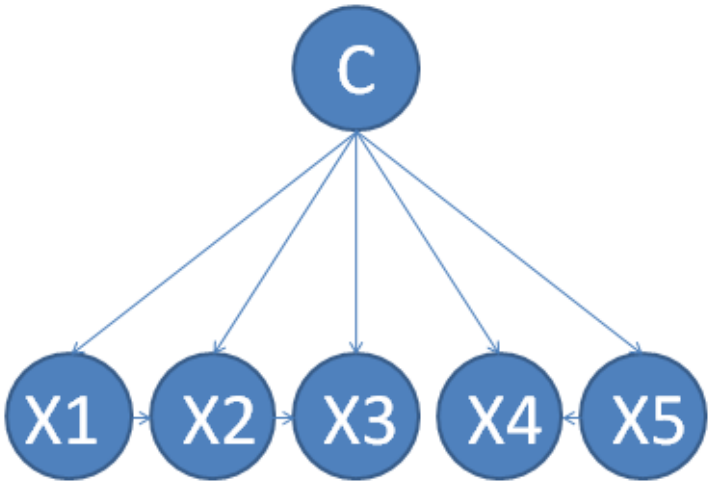


Figure 5.3: One Dependence Estimator [35].

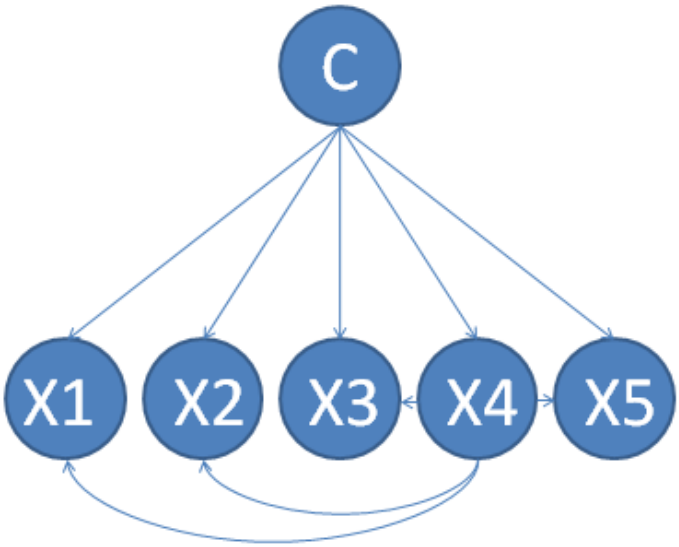


Figure 5.4: Super Parent One Dependence Estimator [35].

5.1 Naïve Bayes

In the Naïve Bayes algorithm, each attribute has only one parent, which is the class attribute, as shown in Figure 5.2. Naïve bayes predicts from a training sample of classified objects, the class of a sample $x = x_1$,x_2...,x_n,where x_i is the value of the i_th attribute. We can minimize error by selecting $argmax_y\ P(y \mid x)$, where $y \in \{c1, ...c_k\}$ are the k classes. It is intended to seek an estimate $\hat{P}(y \mid x)$ of $P(y \mid x)$ and perform classification by selecting $argmax_y\ \hat{P}(y \mid x)$.

Nave Bayesian prediction requires each conditional probability to be non zero. Otherwise, the predicted probability will result in a zero.

$\hat{P}(X \mid y) = \prod_{k=1}^{n} P(x_k \mid y)$

In order to overcome this, the probability estimation is done by one of the following

three methods:

$$Original : P(x_i \mid C) = \frac{N_{ic}}{Nc} \tag{5.1}$$

$$Laplace : P(x_i \mid C) = \frac{N_{ic} + 1}{N_c + c} \tag{5.2}$$

$$m - estimate : P(x_i \mid C) = \frac{N_{ic} + m * p}{N_c + m} \tag{5.3}$$

where

c: number of classes

p: prior probability, and

m: parameter

Naïve Bayes classifies by selecting $argmax_y(\hat{P}(y) \prod_{i=1}^{n} \hat{P}(x_i \mid y))$

where:

$\hat{P}(y)$ and $\hat{P}(x_i \mid y)$ are estimates of respective probabilities derived from the sample training set, with possible corrections such as laplace, as mentioned earlier.

5.2 AODE

AODE (averaged,one-dependence estimators) is a Bayesian method that solves the problem of attribute-independence assumption in Naïve Bayes. The algorithm outperforms ODE and SPODE in terms of accuracy with less computation time [36]. AODE achieve this by eliminating model selection. The algorithm solves the independence assumption of Naïve Bayes by averaging all models that have a single attribute as a parent to all others. For quality, the algorithm will restrict parents to attributes having more than thirty values.

Moreover, the algorithm gives more accurate classification than Naïve Bayes on datasets with non-independent attributes. At training time, as illustrated by the algorithm in Table 5.1, a three dimensional frequency joint table, indexed by class values for one dimension and attribute values for the other two dimensions, is generated. Subsequently, at classification time an m-estimate is used to produce conservative estimates of conditional probabilities from joint frequencies, using the table generated during training.

The probability for a given data row of a dataset to belong to a given class y using AODE is given by:

$$\hat{P}(y,x) = \frac{(\sum_{i:1\leq \imath \leq norF(xi)\geq m} \hat{P}(y,x_i) \prod_{j=1}^{n} \hat{P}(x_j \mid y,x_i))}{i : 1 \leq \imath \leq norF(xi) \geq m} \tag{5.4}$$

AODE classifies by selecting $argmax_y(\sum_{i:1\leq i\leq norF(xi)\geq m} \hat{P}(y,x_i) \prod_{j=1}^{n} \hat{P}(x_j \mid y,x_i))$, and can be extended to provide direct class probability estimates by normalizing the numerators of previously obtained probabilities, across all classes.

The time complexity of training is given by $\mathrm{O}(n^2 i)$, where i is the number of training instances and n is the number of attributes. Training time complexity is linear with respect to the training set size. Thus, AODE is an efficient algorithm for classification from large datasets.

The time complexity of classifying an object is $\mathrm{O}(n^2 c)$, where n is number of attributes and c is the number of classes. The space requirement for the algorithm

INPUTS: training set X^*, Y^*,
number of attributes k, and
number of classes m

OUTPUTS: joint frequency vector $freq$,
class frequency vector $cfreq$,
attribute frequency vector $afreq$,
attribute-value frequency vector $vfreq$, and
item count $count$

Initialize frequencies
$count \leftarrow 0$
Initialize all elements of $freq$, $cfreq$, $afreq$, and $vfreq$ to 0

Accumulate frequencies
FOR EACH $x, y \in X^*, Y^*$
  $count \leftarrow count + 1$
  $cfreq[y] \leftarrow cfreq[y] + 1$
  FOR $i \leftarrow 1$ **TO** k
    IF x_i is known
      $afreq[i] \leftarrow afreq[i] + 1$
      $vfreq[x_i] \leftarrow vfreq[x_i] + 1$
      FOR $j \leftarrow 1$ **TO** k
        IF x_j is known
          $freq[y, x_i, x_j] \leftarrow freq[y, x_i, x_j] + 1$
        END IF
      END FOR
    END IF
  END FOR
END FOR

Table 5.1: Training time algorithm [36].

is the space needed to store the joint frequency table, given by $O((nv)^2c)$ where v is the average number of attribute values.

5.3 Simulation Analysis

The first Naïve Bayes simulation using a training set of 15747 instances and a test set of 5249 was performed, to observe an accuracy of 94.19%, as can be seen from Figure 5.5. Then, the simulation with the top most prominent features

obtained using the GMDH algorithm was carried out, and an accuracy of 99.30% was noticed. Our third simulation run was with the top 16 ranked features, as acquired from information gain and chi-square, to obtain an accuracy of 92.11%. Using information gain ratio, an accuracy of 94.01% was observed. We provide a detailed performance comparison in Table 5.2, for the Naiive Bayes simulation on the dataset.

From Figure 5.5, we see that using the top ranked gain ratio features to build the Naïve Bayes model helps achieve the highest detection rate of all feature selection algorithms. AUC using top ranked gain ratio features is 98.73%, followed by GMDH best feature set, which yields an area of 97.95%. Using all features, an accuracy of 97.78% is observable, and information gain yields an area of 96.03%. Figure 5.5 shows that the precision-recall graph for some threshold values using information gain for feature ranking, always outperform all other models built using the other feature ranking methods/approaches, in terms of precision and recall. NB models build using all features gives second best results, but, for some thresholds NB models built using GMDH feature selection outperform others in terms of recall. Naïve Bayes models built using GMDH outperform NB models built using top ranked gain ratio features, in term of recall. On the other hand, NB models built using top ranked gain ratio features outperform NB models built using GMDH best features in terms of precision.

The ROC and Precision-Recall graphs in Figures 5.9 and 5.10 show that the best AODE models outperform Naïve Bayes. Moreover, the curve shows that the

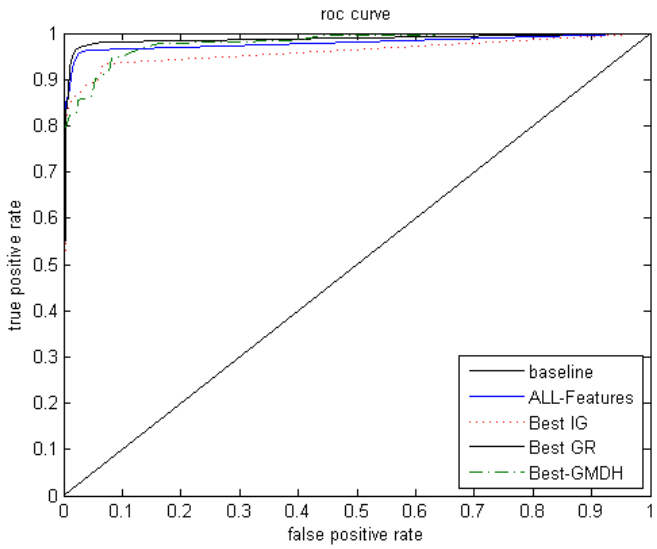


Figure 5.5: Comparison of the ROC curve for different feature sets used in building the NB model.

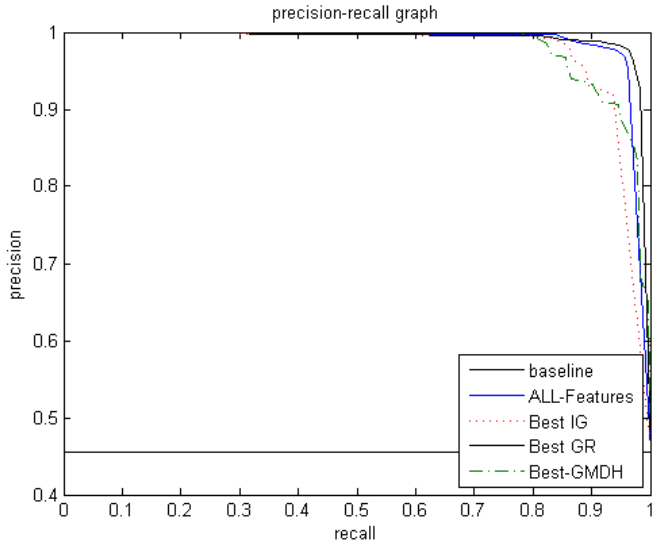


Figure 5.6: Comparison of the Precision-Recall curve for different AODE models built using different feature sets.

FN	TN	FP	TP	FAR	DR	model
377	2830	25	2017	0.008	0.843	NB-GR
392	2833	22	2002	0.007	0.836	NB-IG
433	2781	74	1961	0.025	0.819	NB-GMDH
274	2824	31	2120	0.010	0.885	NB-Full

Table 5.2: Performance results for the NB implementation for intrusion detection.

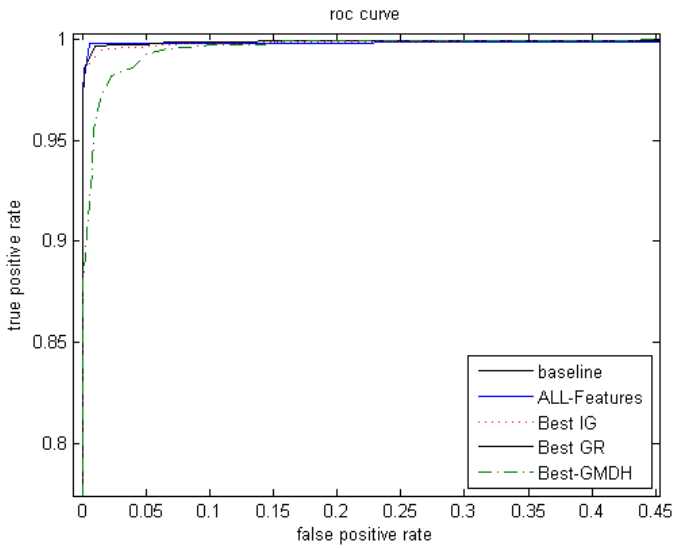


Figure 5.7: Comparison of the ROC curve for different AODE models built using different feature sets.

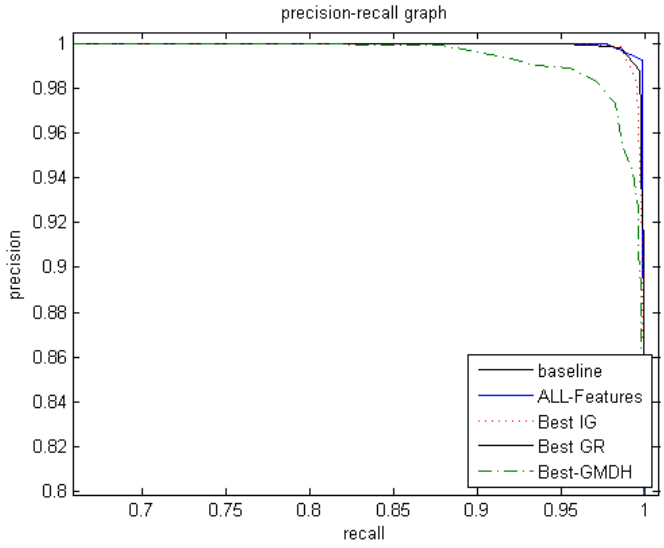


Figure 5.8: Comparison of the Precision-Recall curve for different AODE models built using different feature sets.

FN	TN	FP	TP	FAR	DR	model
13	2841	14	2381	0.005	0.994	AODE-GR
14	2838	17	2380	0.006	0.994	AODE-IG
138	2818	37	2256	0.012	0.942	AODE-GMDH
11	2852	3	2383	0.001	0.995	AODE-Full

Table 5.3: Performance results for the AODE implementation for intrusion detection.

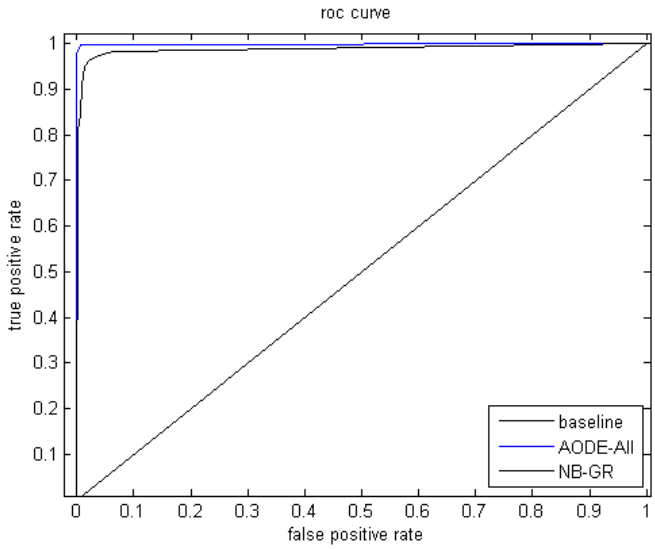


Figure 5.9: Comparison of the ROC curve for the best AODE model and best NB model.

whole AODE curve is always close to a 100% true positive rate and close to a zero false positive rate. This indicates how accurate the AODE model has proven to be, for intrusion detection.

5.4 Result Comparison

In Figures 5.11 and 5.12, a comparison of the AODE detector and the Naïve Bayes detector is shown. The AODE model outperforms NB and GMDH abductive network models, with the curve being close to unity for the True Positive Rate, and zero for the False Positive Rate. This is followed by GMDH abductive networks, and Naïve Bayes, giving an area of 99.93%, 99.66% and 98.73% respectively with standard errors of 0.0, 0.001 and 0.002.

While comparing results from the previous chapter, with those given in this

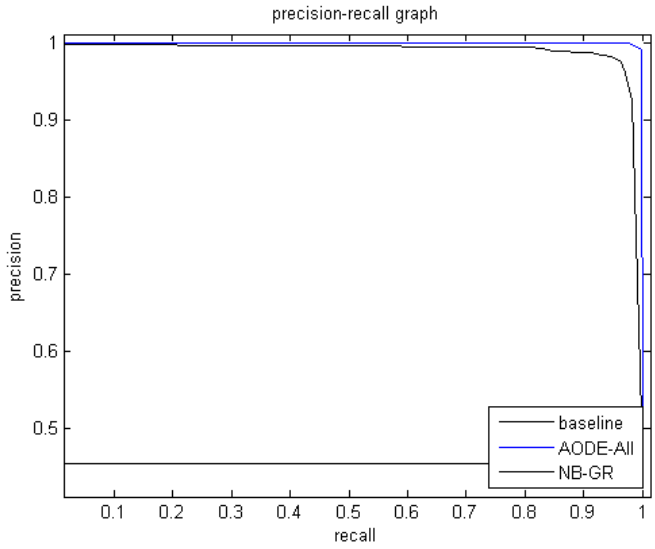


Figure 5.10: Comparison of the Precision-Recall curve for the best AODE model and best NB model.

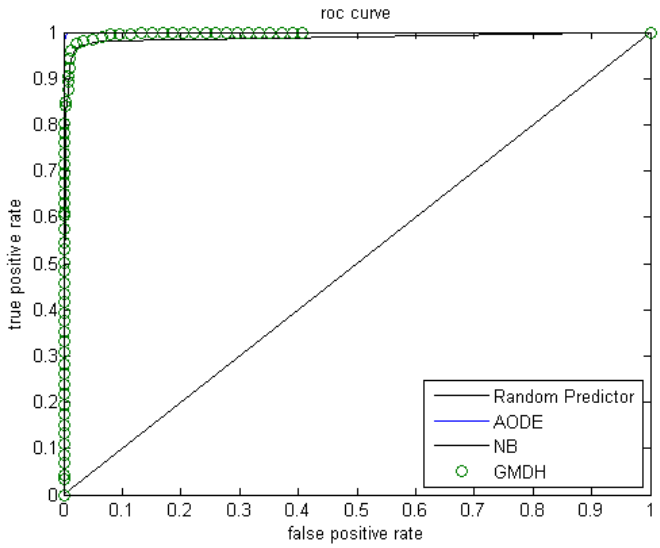


Figure 5.11: Comparison of the ROC curve between the best AODE model, best NB model and best GMDH model.

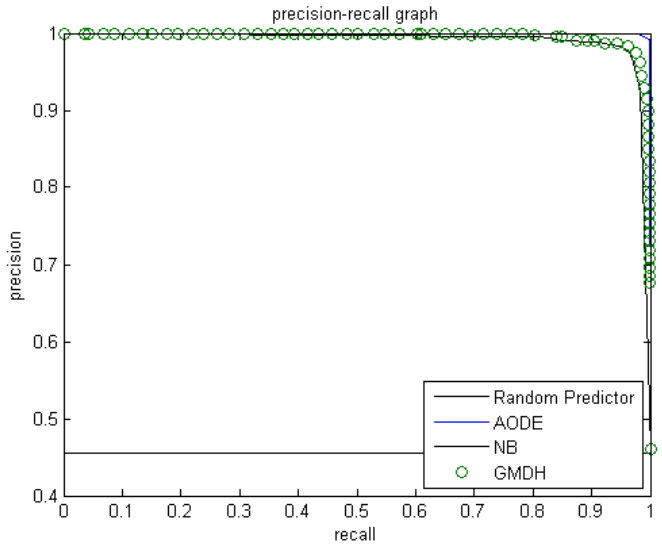


Figure 5.12: Comparison of the Precision-Recall curve between the best AODE model, best NB model and best GMDH model.

chapter, it can be seen that AODE outperforms GMDH and NB in terms of detection rate and false alarm rate, with the highest observed Detection Rate of 99.54% and False Alarn Rate of 00.1%. The GMDH detector yields a DR of 97.72% which comes second after AODE. However, it gives the worst FAR of 2.25%. NB detector gives a DR of 88.55% and FAR of 1.08%. This shows that statistical inference detectors perform better than GMDH, in terms of minimizing the false alarms.

As expected, results shows that filter feature selection methods produce better results than the GMDH-based wrapper method when used with Naïve Bayes and AODE. This is because filter methods are more generic and are based on statistical inference. On the other hand, from the results we can also see that the GMDH based wrapper methods yield better results when used with GMDH abductive networks. This is because GMDH-based wrapper methods incorporate the GMDH

algorithm in its process of feature selection.

5.5 Summary

In this chapter, we proposed an AODE implementation for intrusion detection. We performed simulations using various feature selection techniques, and both Naïve Bayes and AODE for classification. The results were compared with those obtained previously from the GMDH intrusion detection system. It was noticed that AODE outperforms GMDH and Naïve Bayes in terms of detection rates, false alarm rates, as well as training time. It was also observed that GMDH outperforms Naïve Bayes in terms of detection and false alarm rates.

CHAPTER 6

CONCLUSION

This chapter concludes the thesis by outlining the contributions achieved. Moreover, it discusses some of the open problems for future work.

6.1 Results Comparison

Table 6.1 shows that our AODE approach outperforms all other methods in terms of false alarms rate and comes second only to ESOM, in terms of detection rates. However, the difference is negligible. GMDH comes second after AODE/ESOM in terms of detection rate, and fourth in terms of false alarms.

FAR	**DR**	**Method**
2%	96.07%	PCC [37]
2.25%	97.72%	GMDH**
00.1%	99.54%	AODE**
1.08%	88.55%	NB**
3.4%	99.56%	ESOM [38]
3.5%	93%	MLP [39]

Table 6.1: Performance Comparison between various intelligent techniques for intrusion detection.

As part of this thesis, the following contributions were made:

1. Different anomaly-based intrusion detection systems were studied.

2. Data reduction techniques were used to reduce the input feature set to the intrusion detection system.

3. Two techniques, namely GMDH-based and AODE-based, were proposed for intelligent classification of network traffic.

4. Simulations were performed, and the results were compared using evaluation measures explained in Section 3.5, to test the effectiveness of the proposed approaches for intrusion detection.

6.2 Future Work

Many issues in the *Intrusion Detection* problem are open for further research. Some of them are:

- **Alternative detection techniques:** Multi agent detection systems that can detect DDoS attacks efficiently.

- **Dataset problems:** The dataset still old and intrusions are emerging day by day. So, there should be a way to create recent new specialized datasets for attack types.

- **Information visualization techniques:** Information visualization is hot emerging science that can be utilized for intrusion detection. ArchSight

event correlation is an example of using information visualization for intrusion detection.

APPENDIX A

C# CODE FOR DATA HANDLING

A.1 Code for calculating accuracy for gmdh result files

```
using System.IO;
using System.Collections;
using System.Text.RegularExpressions;

public class GetROCFile
{
    public static void Main(string [] args)
    {
        float threshold = float.Parse(args[2]);
        DirectoryInfo root = new DirectoryInfo(args [0]);
        FileInfo[] files = root.GetFiles("*" + args[1] + ".TXT");
        string outputFile = args [1]+"\_allresults.txt";
        TextWriter tw = new StreamWriter(outputFile);
        int fileCount = files.Length;
        int kk = 1;
        float[,] outputs = new float[5249,fileCount+1];
```

```
foreach(FileInfo file in files)
{
    string inputFile = file.FullName;

    TextReader tr = new StreamReader(inputFile);
    string input = tr.ReadToEnd();
    tr.Close();
    int TP = 0;
    int TN = 0;
    int FP = 0;
    int FN = 0;

    Regex reg = new Regex(@"(?<in>[\d|\.]+)\t(?<in1>[\d|\.]+)\t(?<act>[\d|\.]+)\t(?<est>[\d|\.]+)\t[\^\\n]*");

    MatchCollection mc = reg.Matches(input);
    int ii = 0;
    foreach(Match m in mc)
    {
        float nu = float.Parse(m.Groups["est"].ToString());

        float act =  float.Parse(m.Groups["act"].ToString());
        outputs[ii,0] = act;
        outputs[ii,kk] = nu;
        ii++;
        if (act > 0.5 && nu >= threshold)
            TP++;
        else if (act > 0.5 && nu < threshold)
            FN++;
        else if(act<0.5 && nu <0.5)
            TN++;
        else if (act < 0.5 && nu >= threshold)
            FP++;
    }

    double acc = (TP+TN)/(1.0*(FP+TP+FN+TN));
    tw.WriteLine("acc = {0}, {1} {2} {3} {4} {5}",acc,TP,FP,TN,FN,file.Name);
    Console.WriteLine("acc = {0}, {1} {2} {3} {4}",acc,TP,FP,TN,FN);
    kk++;
}
float avg = float.Parse("0.0");

tw.Close();
TextWriter tw2 = new StreamWriter(args[1] + "\_major\_vote.txt");
int rowscount = outputs.GetLength(0);
int colscount = outputs.GetLength(1);
 int TP1 = 0;
int TN1 = 0;
int FP1 = 0;
int FN1 = 0;

for (int i = 0; i < rowscount; i++)
{
    float act1 = outputs[i, 0];
    for (int k = 1; k < colscount; k++)
    {
        avg = avg + outputs[i, k];
    }
    avg = avg / colscount;
    float nu = avg;
    if (act1 > 0.5 && nu >= threshold)
        TP1++;
    else if (act1 > 0.5 && nu < threshold)
        FN1++;
    else if (act1 < 0.5 && nu < threshold)
        TN1++;
    else if (act1 < 0.5 && nu >= threshold)
        FP1++;
    Console.WriteLine("{0} : {1} : {2}", act1, nu, nu >= threshold ? "n" : "p");
    avg = 0;
}
double acc1 = (TP1 + TN1) / (1.0 * (FP1 + TP1 + FN1 + TN1));
tw2.WriteLine("acc = {0}, {1} {2} {3} {4} {5}", acc1, TP1, FP1, TN1, FN1, "ALL");
Console.WriteLine("acc = {0}, {1} {2} {3} {4}", acc1, TP1, FP1, TN1, FN1);
kk++;
rowscount = outputs.GetLength(0);
colscount = outputs.GetLength(1);
TP1 = 0;
TN1 = 0;
FP1 = 0;
FN1 = 0;
```

```
        int pcount = 0,ncount = 0;
        for (int i = 0; i < rowscount; i++)
        {
            float act1 = outputs[i, 0];
            for (int k = 1; k < colscount; k++)
            {
                avg = avg + outputs[i, k];
                if (outputs[i, k] >= threshold)
                    pcount++;
                else
                    ncount++;
            }
            Console.WriteLine(pcount+":"+ncount);
            avg = avg / (colscount);
            float nu = avg;
            if (pcount > ncount)
                nu = 1.0F;
            else if (pcount < ncount)
                nu = 0.0F;
            pcount = 0; ncount = 0;
            if (act1 > 0.5 && nu >= threshold)
                TP1++;
            else if (act1 > 0.5 && nu < threshold)
                FN1++;
            else if (act1 < 0.5 && nu < threshold)
                TN1++;
            else if (act1 < 0.5 && nu >= threshold )
                FP1++;
            Console.WriteLine("{0} : {1} : {2}", act1, nu, nu >= threshold ? "n" : "p");

            avg = 0;
    }
    acc1 = (TP1 + TN1) / (1.0 * (FP1 + TP1 + FN1 + TN1));
    tw2.WriteLine("acc = {0}, {1} {2} {3} {4} {5}", acc1, TP1, FP1, TN1, FN1, "ALL");
    Console.WriteLine("acc = {0}, {1} {2} {3} {4}", acc1, TP1, FP1, TN1, FN1);
    kk++;
    tw2.Close();

    }
}
```

References

[1] L. Portnoy, E. Eskin, and S. Stolfo, "Intrusion detection with unlabeled data using clustering," in *Proceedings of ACM CSS Workshop on Data Mining Applied to Security, Philadelphia, PA*, 2001.

[2] W. Feng, "The case for TCP/IP puzzles," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 4, p. 327, 2003.

[3] D. Barbara, N. Wu, and S. Jajodia, "Detecting novel network intrusions using bayes estimators," in *First SIAM Conference on Data Mining*, 2001.

[4] S. Stolfo, W. Lee, P. Chan, W. Fan, and E. Eskin, "Data mining-based intrusion detectors: an overview of the Columbia IDS project," *ACM SIGMOD Record*, vol. 30, no. 4, p. 14, 2001.

[5] K. Faraoun and A. Boukelif, "Neural networks learning improvement using the k-means clustering algorithm to detect network intrusions," *International Journal of Computational Intelligence*, vol. 3, no. 2, pp. 161–168, 2006.

[6] J. Cannady, "Artificial neural networks for misuse detection," in *National information systems security conference*, pp. 368–81, 1998.

[7] A. Sung and S. Mukkamala, "Identifying important features for intrusion detection using support vector machines and neural networks," in *Proceedings of Symposium on Applications and the Internet, 2003.*, pp. 209–216, 2003.

[8] H. Kayacik, A. Zincir-Heywood, and M. Heywood, "On the capability of an SOM based intrusion detection system," in *Proceedings of the International Joint Conference on Neural Networks, 2003.*, vol. 3, 2003.

[9] L. DeLooze, "Attack characterization and intrusion detection using an ensemble of self-organizing maps," in *2006 IEEE Information Assurance Workshop*, pp. 108–115, 2006.

[10] O. Depren, M. Topallar, E. Anarim, and M. Ciliz, "An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks," *Expert Systems with Applications*, vol. 29, no. 4, pp. 713–722, 2005.

[11] A. Mitrokotsa and C. Douligeris, "Detecting denial of service attacks using emergent self-organizing maps," in *Proceedings of the Fifth IEEE International Symposium on Signal Processing and Information Technology, 2005.*, pp. 375–380, 2005.

[12] E. Gelenbe, "Stability of the random neural network model," *Neural Computation*, vol. 2, no. 2, pp. 239–247, 1990.

[13] E. Gelenbe, "Random neural networks with negative and positive signals and product form solution," *Neural Computation*, vol. 1, no. 4, pp. 502–510, 1989.

[14] G. Oke and G. Loukas, "A denial of service detector based on maximum likelihood detection and the random neural network," *The Computer Journal*, vol. 50, no. 6, p. 717, 2007.

[15] Z. Ghahramani and M. Beal, "Propagation algorithms for variational Bayesian learning," *Advances in Neural Information Processing Systems*, pp. 507–513, 2001.

[16] J. Cheng and R. Greiner, "Learning bayesian belief network classifiers: Algorithms and system," *Advances in Artificial Intelligence*, pp. 141–151.

[17] M. Tavallaee, E. Bagheri, W. Lu, and A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proceedings of the Second IEEE International Conference on Computational Intelligence for Security and Defense Applications*, pp. 53–58, IEEE Press, 2009.

[18] J. McHugh, "Testing intrusion detection systems: A critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory," *ACM Transactions on Information and System Security*, vol. 3, no. 4, pp. 262–294, 2000.

[19] I. Witten and E. Frank, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann Publisher, 2005.

[20] S. Doraisamy, S. Golzari, N. Norowi, M. Sulaiman, and N. Udzir, "A study on feature selection and classification techniques for automatic genre classification of traditional Malay music," in *Proceedings of the International Confer-*

ence on Music Information Retrieval, Philadelphia, PA, USA, pp. 331–336, 2008.

[21] M. Hall and L. Smith, "Practical feature subset selection for machine learning," *Computer Science*, vol. 98, pp. 4–6, 1998.

[22] R. Abdel-Aal, "GMDH-based feature ranking and selection for improved classification of medical data," *Journal of Biomedical Informatics*, vol. 38, no. 6, pp. 456–468, 2005.

[23] J. Davis and M. Goadrich, "The relationship between precision-recall and roc curves," in *Proceedings of the 23rd International Conference on Machine Learning*, p. 240, ACM, 2006.

[24] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006.

[25] C. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing*. MIT Press, 2000.

[26] V. Raghavan, P. Bollmann, and G. Jung, "A critical investigation of recall and precision as measures of retrieval system performance," *ACM Transactions on Information Systems (TOIS)*, vol. 7, no. 3, pp. 205–229, 1989.

[27] R. Bunescu, R. Ge, R. Kate, E. Marcotte, and R. Mooney, "Comparative experiments on learning information extractors for proteins and their interactions," *Artificial Intelligence in Medicine*, vol. 33, no. 2, pp. 139–155, 2005.

[28] M. Goadrich, L. Oliphant, and J. Shavlik, "Learning ensembles of first-order clauses for recall-precision curves: A case study in biomedical information extraction," *Inductive Logic Programming*, pp. 98–115.

[29] I. Al-Awal, "Recognition of handwritten arabic (indian) digits using abductive network," Jan 2010.

[30] S. Farlow, *Self-organizing Methods in Modeling: GMDH-type Algorithms*. CRC, 1984.

[31] Z. Zheng and G. Webb, "Lazy Bayesian rules," *Deakin University Computing Technical Report TR C*, vol. 98, 1998.

[32] Y. Yang, G. Webb, J. Cerquides, K. Korb, J. Boughton, and K. Ting, "To select or to weigh: A comparative study of model selection and model weighing for spode ensembles," *Machine Learning: ECML 2006*, pp. 533–544.

[33] H. Arsham, "Statistical Thinking for Managerial Decisions," 1994.

[34] E. Keogh and M. Pazzani, "Learning augmented Bayesian classifiers: A comparison of distribution-based and classification-based approaches," in *Proceedings of the Seventh International Workshop on Artificial Intelligence and Statistics*, pp. 225–230, 1999.

[35] Y. Yang, K. Korb, K. Ting, and G. Webb, "Ensemble Selection for Superparent-One-Dependence Estimators," *AI 2005: Advances in Artificial Intelligence*, pp. 102–112.

[36] G. Webb, J. Boughton, and Z. Wang, "Not so naive bayes: Aggregating one-dependence estimators," *Machine Learning*, vol. 58, no. 1, pp. 5–24, 2005.

[37] M. Shyu, S. Chen, K. Sarinnapakorn, and L. Chang, "A novel anomaly detection scheme based on principal component classifier," in *Proceedings of the IEEE Foundations and New Directions of Data Mining Workshop*, 2003.

[38] A. Mitrokotsa and C. Douligeris, "Intrusion Detection Using Emergent Self-organizing Maps," *Advances in Artificial Intelligence*, pp. 559–562.

[39] M. Sabhnani and G. Serpen, "Application of machine learning algorithms to KDD intrusion detection dataset within misuse detection context," in *Proceedings of International Conference on Machine Learning, Models, Technologies and Applications*, 2003.

Vita

Personal Information

- Name: AbdulRhman Salah Mohammed Shaheen
- Nationality: Palestinian
- Current Address: Saudi Arabia - Al-Khobar
- Permanent Address: Saudi Arabia - Riyadh
- Email: abdulrhman.shaheen@gmail.com

Education

- 2010: *Master* of Science in Computer Engineering KFUPM
- 2006: *Bachelor* of Science in Computer Engineering KFUPM

Work Experience

- 2007-Currently: Information Technology Center KFUPM
- 2006-2007: Universal Cold Store Dammam