DATA REDUCTION AND DATA CLASSFICATION IN AN INTRUSION DETECTION SYSTEM

By

SRILATHA CHEBROLU

Bachelor of Technology

Shadan College of Engineering and Technology

Hyderabad, India

2001

Submitted to the Faculty of the
Graduate college of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
December, 2003

DATA REDUCTION AND DATA CLASSFICATION

IN AN INTRUSION DETECTION SYSTEM

Thesis Approved:

[illegible]

Thesis Adviser

[illegible]

[illegible]

[illegible]

Dean of the Graduate College

PREFACE

Intrusion detection is a critical component of secure information systems. The purpose of this study is to identify important input features in building an intrusion detection system which is least computational expensive and to build an intrusion detection system which is effective. Since elimination of useless or insignificant inputs leads to a simplification of the problem, faster and accurate results will result. Feature ranking and selection or data reduction, therefore, is an important issue for data classification in intrusion detection systems. Independent component analysis algorithm, and principal component analysis algorithm, is used for data reduction in intrusion detection systems. For effective intrusion detection, we use Bayesian belief network classifier, and classification and regression trees classifier for data classification.

ACKNOWLEDGEMENTS

I wish to express my sincere appreciation to my advisors, Dr. Johnson Thomas and Dr. Ajith Abraham for their intelligent supervision, constructive guidance, inspiration and friendship. My sincere appreciation extends to my committee member Dr. Debao Chen, whose guidance, assistance, encouragement, and friendship are also invaluable.

Moreover, I would also like to give my special appreciation to: My parents for their support, encouragement at times of difficulties and blessings throughout my life. My sister, Saritha Devi Chebrolu for her love and laughter she brought to my life.

Finally, I would like to thank Dr. Johnson Thomas and Department of Computer Science for supporting during these years of study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 Background

Security and Intrusion Detection

Computer or Network security has been studied as a discipline since the early 1970s. It refers to measures and controls that protect an information system against denial of service and unauthorized (accidental or intentional) disclosure, modification, or destruction of information systems and data. A secure computer or network system should provide the following services-data confidentiality, data and communications integrity, and assurance against denial-of-service. Data confidentiality service protects information or data against unauthorized disclosure. This service should protect release of a message's content to unauthorized users. Data and communications integrity service is concerned with the accuracy, faithfulness, non-corruptibility, and believability of information transfer between peer entities (including computers connected by a network). This service must ensure correct operation of the system hardware and firmware, and it should protect against unauthorized modification of data and labels. Denial-of-service is an important security service. A denial-of-service condition is said to exist whenever the system throughput falls below a pre-established threshold, or when access to a (remote) entity is unavailable. While such attacks are not completely preventable, it is often desirable to reduce the probability of such attacks below some threshold.

Though different computer or network systems may have different definitions of security, computer scientists have developed common security mechanisms to protect computer systems. Early attempts of the protection mechanisms include authentication or identification, encryption, access control, etc. The goal of these mechanisms is to prevent unauthorized users from compromising the data confidentiality, data and communications integrity, and assurance against denial-of-service. Thus they can be collectively called prevention-based techniques.

It has been noticed that the prevention-based techniques cannot assure the security of the systems being protected. For example, in 1998 the Internet worm brought down the majority of the Internet by taking advantage of vulnerabilities in rsh, fingerd, and sendmail. Even in the year 2000, the so-called Distributed Denial of Service (DDoS) attacks stopped several major commercial sites, including Yahoo and CNN, from functioning normally, though they were protected by prevention-based techniques. Indeed a deeper reason is that the processes with which human beings develop information systems are not completely error-proof: there may be bugs in the implementation of the systems, and, moreover, there may be errors in the design of the information systems.

Intrusion detection was proposed to complement the prevention-based security measures. An intrusion is defined to be a violation of the security policy of the system; intrusion detection thus refers to the mechanisms that are developed to detect the violation of the system security policy. Intrusion detection is based on the assumption that intrusive activities are noticeably different from normal system activities and thus detectable. Intrusion detection is not introduced to replace the prevention-based techniques such as authentication and access control; instead, it is intended to be used along with the existing security measures and detect the actions that bypass the security control of the system. Thus, intrusion detection is usually considered as a second line of defense for computer and network systems.

Intrusion detection is defined to be the problem of identifying users or hosts or programs that are using a computer system without authorization and those who have legitimate access to the system but are abusing their privileges (i.e. the "insider threat"). Generally, an intrusion would cause loss of integrity, denial of resources, or unauthorized use of resources.

Some specific examples of intrusions that concern system administrators include:

- Unauthorized modifications of system files so as to permit unauthorized access to either system or user information.
- Unauthorized access or modification of user files or information.
- Unauthorized modifications of tables or other system information in network components (e.g. modifications of router tables in an internet to deny use of the network).
- Unauthorized use of computing resources (perhaps through the creation of unauthorized accounts or perhaps through the unauthorized use of existing accounts).

Detecting attacks requires the use of a model of intrusion, namely, what should the IDS look for? The first model hypothesizes its detection upon the profile of a user's (or a group of users) normal behavior. It statistically analyzes parameters of the user's current session, compares them to the profile representing the user's normal behavior, and reports significant deviations to a system security officer. Here significant is defined as a threshold set by the specific model or by the system security officer. A typical Intrusion detection system may report the top ten most suspicious sessions to the system security officer. Because it catches sessions, which are not normal, it is referred to as an "anomaly" detection model.

The second type of model bases its detection upon a comparison of parameters of the user's session and the user's commands to a rule-base of techniques used by attackers to penetrate a system. Attack signatures (i.e. known attack methods) are what this model looks for in the user's behavior. Since this model looks for patterns known to cause security problems, it is called a "misuse" detection model.

Early IDS models were designed to monitor a single host. However, more recent models accommodate the monitoring of a number of hosts interconnected by a network, e.g. ISOA, IDES, and UC Davis' Network Security Monitor (NSM) and Distributed Intrusion Detection System (DIDS). Some of these systems (ISOA and IDES) transfer the monitored information (host audit trails) from the monitored hosts to a central site for processing. Others (NSM, DIDS) monitor the network flow as well, as part of their intrusion detection algorithms.

1.2 Current Problems

Most existing intrusion detection systems suffer from some of the following problems:

Current IDS lack data reduction procedures: The information used by the intrusion detection system is obtained from audit trails or from packets on a network. An audit trail is defined to be the information consisting of all user and system interactions. The audit trail contains all the data needed to perform intrusion detection. There might be more data than needed. Inspecting the sheer volume of audit information generated requires a large effort and it is computationally expensive.

Current IDS lack effectiveness: An IDS is effective if it has both high intrusion detection (i.e. true positive rate) and low false alarm (i.e. false positive) rate. The handcrafted rules and patterns, and the statistical measures on selected system measures are the codified "expert knowledge" in security, system design, and the particular intrusion detection approaches in use. Expert knowledge is usually incomplete and imprecise due to the complexities of the network systems.

In this thesis, we describe a mechanism that addresses above two problems and also has several other desirable characteristics.

1.3 Objectives of the Study

Our objective is to apply two techniques to build an Intrusion detection system in two important ways:

- Since Intrusion detection is done online, our task is to build a model which is least computationally expensive (so elimination of redundant variables is important). We achieve this by data reduction using Independent Component Analysis algorithm, principal component analysis algorithm, CBL2 algorithm on general Bayesian network classifier and CART algorithm.

- It is necessary to build efficient Intrusion detection systems for data modeling, classification and prediction. So we do data classification by building Bayesian Network Intrusion detection model and Classification and Regression Trees (CART) Intrusion Detection model. We train the IDS to classify attacks by using Bayesian Belief Network data mining algorithm and CART algorithm.

The dataset used for applying the above soft computing techniques is a sample of the network traffic and audit logs, which capture the actual behavior, in the forms of statistical summaries, of normal activities and intrusions. Therefore, the intrusion detection models built on this dataset can be more effective in distinguishing normal and intrusion activities.

1.4 Significance of the Study

Since the amount of audit data that an IDS needs to examine is very large even for a small network, data reduction is a necessary task. Also as intrusion detection is done online, data reduction is necessary to reduce the computational time and to maximize the scalability and fast re-training or tuning of an IDS. Data mining approaches are relatively new techniques for intrusion detection. Previous research in data mining approaches for intrusion detection model identified several types of algorithms as useful techniques. Classification is one of the data mining algorithms which have been investigated as a useful technique for intrusion detection models. There are several classification algorithms available. Successful application of few algorithms for intrusion detection in previous research motivated us to use Bayesian belief network and classification and regression trees as intrusion detection models. Also no one has done research in this area by using classification and regression trees and Bayesian belief network for data reduction and as intrusion detection models. Classification and regression trees algorithm is unique among other decision tree algorithms because of its stable performance and reliable results. Classification and regression trees is an excellent pre-processing algorithm to other data analysis techniques because classification and regression trees outputs can be used as inputs to improve the predictive accuracy of neural nets and logistic regression. Also Classification and regression trees can extract the most important variables from a very large dataset. Bayesian network PowerPredictor gave the best prediction accuracy on KDDCup 2001 Task one, from among 114 submissions all over the world. Bayesian belief network being selected as KDDCup 2001 Data Mining Competition Winner inspired us to use it as an intrusion detection model. In this thesis we compare performance accuracies of classification and regression trees and Bayesian belief network to know their advantages and disadvantages. Also we compare Bayesian BN classifier's performance and classification and regression trees performance on the original datasets and on reduced datasets.

Our ultimate objective is to build a model that is least computational expensive and to build classification and regression trees and Bayesian belief network intrusion detection models.

regression trees as intrusion detection models. Also no one has done research in this area by using classification and regression trees and Bayesian belief network for data reduction and as intrusion detection models. Classification and regression trees algorithm is unique among other decision tree algorithms because of its stable performance and reliable results. Classification and regression trees is an excellent pre-processing algorithm to other data analysis techniques because classification and regression trees outputs can be used as inputs to improve the predictive accuracy of neural nets and logistic regression. Also Classification and regression trees can extract the most important variables from a very large dataset. Bayesian network PowerPredictor gave the best prediction accuracy on KDDCup 2001 Task one, from among 114 submissions all over the world. Bayesian belief network being selected as KDDCup 2001 Data Mining Competition Winner inspired us to use it as an intrusion detection model. In this thesis we compare performance accuracies of classification and regression trees and Bayesian belief network to know their advantages and disadvantages. Also we compare Bayesian BN classifier's performance and classification and regression trees performance on the original datasets and on reduced datasets.

Our ultimate objective is to build a model that is least computational expensive and to build classification and regression trees and Bayesian belief network intrusion detection models.

Chapter 2

LITERATURE REVIEW

Dr. Dorothy Denning presented in 1986 what would become the pivotal paper in the area of computer system intrusion detection. In her paper, Dr. Denning suggested that if it can be assumed that exploitation of a computer system involves an abnormal use of the system, then security violations could be detected by looking for abnormal patterns of system usage. She went on to explain how profiles could be developed which described authorized user`s normal activities. Unauthorized activity would then be indicated by behavior not fitting these individual profiles. The profiles can be created by maintaining a record of each user's actions and can be periodically updated to reflect possible changes in the user's normal activities. Profiling has been used by many of the current IDSs.

In this chapter, we first review definitions of intrusion, intrusion detection and intrusion detection system, characteristics of an intrusion detection system, different intrusion detection methods and types of intrusion detection. Then we review different techniques employed in intrusion detection systems such as neural network techniques, support vector machine techniques, data mining techniques and then study intrusion detection in wireless ad-hoc networks. Then, finally we review data reduction in intrusion detection.

2.1 Intrusion Detection

Definition: Intrusion

Any set of actions that attempt to compromise the integrity, confidentiality, or availability of a computer resource.

This definition disregards the success or failure of those actions, so it also corresponds to attacks against a computer system.

Definition: Intrusion Detection:

The problem of identifying actions that attempt to compromise the integrity, confidentiality, or availability of a computer resource.

Definition: Intrusion Detection System

A computer system (possibly a combination of software and hardware) that attempts to perform intrusion detection.

Most intrusion detection systems try to perform their task in real time. However, there are also intrusion detection systems that do not operate in real time, either because of the nature of the analysis they perform or because they are meant for forensic analysis (analysis of what has happened in the past on a system).

The definition of intrusion detection system does not include preventing the intrusion from occurring, only detecting it and reporting it to the operator. There are some intrusion detection systems that try to react when they detect an unauthorized action. This reaction usually includes trying to stop the damage, for example by terminating a network connection.

2.2 Characteristics of an intrusion detection system.

An intrusion detection system will have the following characteristics:

1. It must *run continually* with minimal human supervision.
2. It must be *fault tolerant*:

(a) The Intrusion detection system must be able to recover from system crashes, either accidental or caused by malicious activity.

(b) After a crash, the intrusion detection system must be able to recover its previous state and start its operation unaffected.

3. It must *resist subversion*:

   (a) For an attacker to disable or modify the intrusion detection system there must be a significant difficulty.

   (b) The intrusion detection system must be able to monitor itself and detect if an attacker has modified it.

4. It must impose a *minimal overhead* on the systems where it runs to avoid interfering with their normal operation.

5. It must be *configurable* to accurately implement the security policies of the systems that are being monitored.

6. It must be *easy to deploy*: This can be achieved through portability to different architectures and operating systems, through simple installation mechanisms, and by being easy to use and understandable by the operator.

7. It must be *adaptable* to changes in system and user behavior over time. For example, new applications being installed, users changing from one activity to another or new resources being available can cause changes in system use patterns.

8. It must be able to *detect attacks*:

   (a) The intrusion detection system must not recognize any legitimate activity as an attack (false positives).

(b) The intrusion detection systems must not fail to recognize any real attacks (false negatives). It must be difficult for an attacker to mask his actions to avoid detection.

(c) The intrusion detection system must report attacks as soon as they occur.

(d) The intrusion detection system must be general enough to detect different types of attacks.

2.3 Different Intrusion detection methods

There are two types of intrusion detection methods: misuse detection and anomaly detection.

2.3.1 Anomaly Detection

By definition, anomalies are not normal. Anomaly detection assumes that an intrusion will always reflect some deviations from normal patterns. Anomaly detection can be divided into static and dynamic. A static anomaly detector is based on the assumption that there is a portion of system being monitored that should remain constant. Usually, static detectors only address the software portion of a system and are based on the assumption that the hardware need not be checked. System administration tools check physical component configurations and report change, so such tools will not be treated here. The static portion of a system is the code for the system and the constant portion of data upon which the correct functioning of the system depends. Static portions of the system can be represented as a binary bit string or a set of such strings (such as files). If the static portion of the system ever deviates from its original form, an error has occurred or an intruder has altered the static portion of the system. Static anomaly detectors are meant for checking data integrity. Tripwire [Kim 93, Kim 94] and Self-Nonself [Forrest94] are examples of IDS that perform static anomaly detection.

Dynamic anomaly detectors such as NIDES [Anderson95a, Anderson95b, Javitz93, and Lunt93] or Pattern Matching [Hofmeyer97} must have a definition of behavior to classify as normal or anomalous. Frequently, system designers employ the notion of event. Behavior is defined as a sequence of distinct actions that cause events that are recorded in audit records. Since audit records of operating system only record events of interest, then the only behavior that can be observed is that which results in an event in an audit record. Events may occur in a sequence. In some cases such as with distributed systems, partial ordering of events is sufficient. In still other cases, the order is not directly represented; only cumulative information, such as cumulative processor resource used during a time interval, is maintained. In this case, thresholds are defined to separate normal resource consumption from anomalous resource consumption.

NORMAL
UNCERTAIN
ANOMALOUS

Figure 2.1: Anomalous behavior must be distinguished from normal behavior.

2.3.2 Misuse Detection

Misuse detection is based on the knowledge of system vulnerabilities and the known attack patterns. Misuse detection is concerned with catching intruders who are attempting to break into a system by exploiting some known vulnerability. Ideally, a system security administrator would be able to aware of all the known vulnerabilities and would eliminate them.

The term intrusion scenario is used as a description of a known kind of intrusion; it is a sequence of events that would result in an intrusion without some outside preventive intervention. An intrusion detection system continually compares recent activity to the intrusion scenarios to make sure that someone or combinations of someone's are not attempting to exploit known vulnerabilities. To perform this, each intrusion scenario must be described or modeled in some way. Generally, intrusion scenarios are quite specific.

The main difference between the misuse techniques is in how they describe or model the bad behavior that constitutes an intrusion. Initial misuse detection systems used rules to describe the events indicative of intrusive actions that a security administrator looked for within the system. Large numbers of rules can be difficult to interpret if the rules are not grouped by intrusion scenarios since making modifications to the rule set can be difficult if the affected rules are spread out across the rule set. To overcome these difficulties, alternative intrusion scenario representations are developed. These new rule organizational techniques include model-based rule organization and state transition diagrams. Better rule organization allowed the intrusion scenarios to be described in a more expressive,and understandable way for the misuse detection system user.

Misuse detection systems use the rules to look for events that possibly fit an intrusion scenario. The events may be monitored live by monitoring system calls or later using audit records. Although most systems use audit records, they would be fundamentally the same if they were collecting live system information.

2.3.3 Advantages and disadvantages of Anomaly detection and Misuse detection

The main disadvantage of misuse detection approaches is that they will detect only the attacks for which they are trained to detect. Novel attacks or unknown attacks or even variants of common attacks often go undetected. In a time when new security vulnerabilities in software are discovered and exploited every day, the reactive approach embodied by misuse detection methods is not feasible for defeating malicious attacks.

The main advantage of anomaly detection approaches is the ability to detect novel attacks or unknown attacks against software systems, variants of known attacks, and deviations of normal usage of programs regardless of whether the source is a privileged internal user or an unauthorized external user. The disadvantage of anomaly detection approaches is that well-known attacks may not be detected, particularly if they fit the established profile of the user. Once detected, it is often difficult to characterize the nature of the attack for forensic purposes. Another drawback of many anomaly detection approaches is that a malicious user who knows he or she is being profiled can change his or her profile slowly over time to essentially train the anomaly detection method to learn his or her malicious behavior as normal. Finally a high false positive rate may result for a narrowly trained detection algorithm, or conversely, a high false negative rate may result for a broadly trained anomaly detection approach.

2.4 Types of Intrusion Detection Systems

There are two types of intrusion detection systems: host-based intrusion detection systems and network-based intrusion detection systems. Host-based systems base their decisions on information obtained from a single host (usually audit trails), while network-based systems obtain data by monitoring the traffic in the network to which the hosts are connected.

2.4.1 Host-based Intrusion Detection

A generic intrusion detection model proposed by Dr. Dorothy Denning (1986) works as a rule-based pattern matching system which includes the following six components:

1. Subjects: A subject is the "initiator" of an action being performed on the host, e.g., a user or the host itself.
2. Objects: An object is the "receptor of an action e.g., a system device or a system file.
3. Audit records: An audit record represents an action initiated by the subject and that occurred on the object. Some quantitative measurements on the action are also included in the audit record, e.g., CPU usage time or I/O activity.
4. Profiles: A profile is the "signature or description of normal activity" of a subject or a group of subjects concerning an object or a group of objects, e.g., a profile on the CPU usage of a user session or a profile on the CPU usage of a program. Many statistical models can be included to calculate these quantitative.
5. measurements in these profiles. Some examples include the mean and standard deviation model, Markov process model, and time serial model.
6. Anomaly records: An anomaly record is used to record an anomalous event that has been detected.

7. Activity rules: An activity rule explains what action will be taken under some conditions. For example, when a new audit record is created, the corresponding profile will be updated automatically.

So, intrusion detection tasks are conducted by checking the similarity between the current audit record and the corresponding profiles. If the current audit record deviates from the normal patterns, it will be considered an anomaly.

On the basis of the SRI's IDES (Lunt and Jagannathan 1988) Dr. Denning's model has been proposed. SRI's IDES has two components: the statistical anomaly detector and the expert system (Mukherjee, Heberlein, and Levitt 1994). Based on Denning's model, the first component is used to detect anomalies by applying statistical methods, i.e. the normal patterns are constructed by use of statistical analysis and the anomaly intrusions are detected by assuming that there will be always some differences between normal patterns and intrusions. The expert system component of SRI's IDES is constructed as a rule-based system and is used to detect the intrusions whose patterns are already known.

2.4.2 Single Host Intrusion Detection Systems

After the publication of Dr. Denning's paper several IDSs have been developed. These systems use profile and rule-based approaches to identify intrusive activity. The system's audit trail is used in almost all systems to provide the information on individual user activities.

2.4.2.1 Clyde Digital Systems' AUDIT

Clyde Digital System's AUDIT package is one of the first systems designed to detect unauthorized activity on a computer system. Original designed to address the 'insider threat' (i.e. detecting unauthorized activity performed by authorized users), raised interest in countering intruders. The developers realized that many of the actions they implemented to identify unauthorized activity performed by authorized users would be the same type of actions needed to identify external intruders. The AUDIT package is designed for use on a VAX/VMS system.

AUDIT was designed to detect five categories of activity considered harmful to the computer system or the data it contains if they were performed in an unauthorized manner. These five activities would also be considered detrimental if performed by an intruder. The five categories are:

- Denial of service: irresponsible or inoperable system.
- Information Loss: destroyed data.
- Disinformation: altered data.
- Information Compromise: released data.
- Resource Exploitation: inappropriately used resources.

The AUDIT package has a certain limited analysis capability. Since recording all actions and the keystrokes for all users can produce a tremendous amount of data, AUDIT has the ability to limit the data to certain users, times of the day, or specific programs. The analysis that AUDIT performs results in three reports: a summary report which lists the activity of "high risk" users; a security event report detailing the events that caused users to be listed as 'high risk'; and a supporting data report which includes additional information justifying the conclusions of the other reports.

2.4.2.2 MIDAS

MIDAS stands for the Multics Intrusion Detection and Alerting System. MIDAS was designed to detect intrusive activity on the Multics operating system and runs on a stand-alone Symbolic Lisp machine.

MIDAS addresses five general threat areas; break-ins, masquerading, penetrations, misuse, and Trojan horses/viruses. Misuse is a combination of leakage and denial of service. Identification of an instance of one of these threats is based on the heuristic rules used by the expert system on the Symbolic machine. There are four types of heuristics rules used in MIDAS. Immediate rules are those that make no use of historical information but instead represent activities that by them are suspicious. Anomaly rules use statistical user profiles to detect when current user behavior departs from expected patterns of behavior. The profiles include information such as the user's usual access time and access location, expected typing rate, and the usual commands executed. System-wide state rules use a system-wide profile maintained by MIDAS, which characterizes the normal global state of the system. Finally, sensitive path rules attempt to match the current user's sequence of commands with a command sequence for a known or postulated type of attack.

2.4.2.3 COMPUTERWATCH

COMPUTERWATCH is an add-on audit trail analysis tool developed by AT&T to work with the UNIX System V/MLS operating system. This specific version of UNIX has been evaluated and certified to the Trusted Computer System Evaluation Criteria B1 level of security. This includes a distinct advantage for this package since it means that a certain minimal set of audit records can be expected. It also means, however, that a large amount of data will be produced by the system when the audit trail features are invoked. In order to minimize the amount of data generated by its audit trails, System V/MLS uses a binary format which reduces the average size of an audit record to just sixteen bytes.

Two different types of reports are generated: a canned report which uses a set of rules to detect potential anomalies, and a user specified report which allows the security manager to perform the analysis. The canned reports/queries include listings of failed and successful logins, failed file accesses, failed and successful attempts to gain super user privileges, and other similar events. COMPUTERWATCH is, not designed to be a real-time, stand-alone intrusion detection tool but rather an interactive tool to be used by a system manager to perform different types of analysis on the systems audit trail.

2.4.2.4 DISCOVERY

The DISCOVERY package is an expert system-based intrusion detection system developed by TRW. It is not a general purpose intrusion detection package but is instead a unique package designed to be used for computer services that are sold to an organization other than the owner of the computer system itself. DISCOVERY is designed to recognize the patterns in the usage of authorized users and to store these patterns in profiles for each user. The DISCOVERY package does not operate in real-time but is instead designed to produce a report at the end of each workday for the security manager, which lists any suspicious activities that occurred.

DISCOVERY allows the security manager to select the variables to be monitored and to set the levels atom lithography which suspicions will be raised. DISCOVERY not only checks the selected variables against the stored user profile, it also compares user activities against canned scenarios of intruder access. The user profiles are updated daily.

A limitation of discovery is that it only analyzes correct interactions with the system and does not check specific or patterns of errors. Because of this, DISCOVERY ignores an extremely valuable source of information on possible intrusive activities.

2.4.2.5 IDES

IDES stands for the Intrusion Detection Expert System and it was developed atom lithography SRI International for the States Navy SPAWAR organization. IDES falls uses a profile-based approach to determine when a user's current actions fall outside of an established norm. The profile data kept is updates daily and is weighted so that the most recent observations are given a greater influence on the determination of user norms than are older values. This allows IDES to adapt to changes in user behavior that might occur as a result of new assignments or responsibilities. Like MIDAS, IDES performs its analysis on stand-alone system in real-time. IDES, however, uses a Sun Workstation (as its platform) linked to the monitored host.

An additional feature of IDES is that it is designed to be host-system independent. The receiver is the only component that would need to be modified to adapt IDES to monitor a different host. This assumes, however, that while the new host's audit records may not be in the same format as what is expected by IDES, it nonetheless has the necessary audit capability. One disadvantage of IDES, however, is the need for a separate system to perform the intrusion detection. While this adds a certain extra level of security, the cost is prohibitive unless applied to a network instead of single hosts.

2.4.2.6 Haystack

The Haystack system was developed for use by the United States Air Force on their standard base-level Unisys 1100/60 mainframe computer system. Like the Clyde Digital Systems AUDIT package, Haystack was initially intended to detect threats from insiders (i.e., authorized users) who were exceeding their authority. This was initially believed to be the biggest threat for the specific system the Air Force wanted monitored. Later, as both Haystack and the connectivity of Air Force systems evolved, the goal of the package was extended to include detecting intrusive activity from outsiders.

Haystack is designed to detect six different categories of intrusive behavior. These six categories are listed below:

- Attempted/successful break-ins
- Masquerading
- Leakage
- Denial of Service
- Malicious Use
- Penetration of the Security Control System

The first two categories can be detected through observations that indicate the user's current activities fall outside of an established norm for that particular user or for the group the user is a member of. The next three categories can be detected through abnormal usage of the system and its resources. The final category is detected by observing the use of certain privileged commands or services.

As mentioned before, one of the problems with the use of user profiles is the ability for users to change their profile over a period of time by slowly modifying their behavior. Haystack solves this problem by the use of a group profile. Each user is assigned to a specific group, which is made up of other users with similar responsibilities and authority. While it is possible for a user to change an individual profile, the group profile is not as easily modified and can thus be used to detect abnormal behavior. Even though group profile approach alleviates some of the problems associated with individual user profiles, the approach taken in Haystack to implement them was imperfect. Users are often hard to assign to groups, which has resulted in numerous single-user groups at sites using Haystack. A better methodology needs to be implemented in Haystack for assigning users to groups.

An interesting issue that arose during the development of Haystack illustrates one of the problems with analysis of audit trails. The discussion surrounds the event horizon, which is the method, used to determine the number of events or records the system considers when performing its analysis. An event horizon of one is equivalent to an analysis based only on the current record and the stored user profiles. An event horizon of four means that the analysis will consider the current record and the three previous records. The problem with an event horizon of one however is that certain trends or patterns, which indicate the existence of intrusive activity, might be missed since only the current record is considered. The problem with event horizons greater than one, however, is their computational intensity. The larger the horizon the better the analysis but the more expensive it is computationally. Haystack employs an event horizon of one.

The output of haystack consists of two reports: summary report which provides an overview of what processing has occurred on the monitored system in addition to a list of new users and users whose actions were considered 'suspicious' and the detailed report which lists all anomalous events for each user.

2.4.3 Network-Based Intrusion Detection

With the proliferation of computer networks, more and more individual hosts are connected into LANs of small scale or WANs of large scale. However, the hosts, as well as the networks, are exposed to intrusions due to the vulnerabilities of network devices and network protocols. The TCP/IP protocol can be also exploited by network intrusions such as IP spoofing, port scanning, and so on. So, network-based intrusion detection has become important and is designed to protect a computer network as well as all of its hosts. The installation of a network-based intrusion detection system can also decrease the burden of the intrusion detection task on every individual host.

To detect network-based intrusions, Heberlein et al proposed a network security monitor (NSM), which has a hierarchical architecture composed of the following five layers (from lowest to highest):

1. Packet catcher: It will monitor network traffic, catch every packet, and send it to the next layer.

2. Parser: It will analyze every incoming packet, summarize the security-related information into a four dimensional vector of <source address, destination address, service, connection ID>, and pass it to the next layer.
3. Matrix generator: A corresponding four-dimensional matrix is maintained. Since the connection ID is unique, every connection will be represented by one cell in
4. the matrix. A cell usually stores two measurements: the number of packets and the total data bytes transferred in one connection.
5. Matrix analyzer: Since the matrix actually represents the network traffic, the matrix analyzer will compare it with the normal patterns by use of a "masking" method. Anomaly intrusions will be detected because they will not be masked by normal patterns.
6. Matrix archiver: It will store the matrix atom lithography intervals, e.g., every fourteen minutes. These matrices can then be used to construct normal patterns of network traffic.

NSM detects network anomalies by monitoring network traffic. For misuse detection, LANL's (Los Alamos National Laboratory) NADIR (Network Anomaly Detection and Intrusion Reporter) is built as a rule-base expert system through "audit analysis and consultation with security experts"(Mukherjee, Heberlein, and Levitt 1994).

2.4.4 Network Intrusion Detection Systems

Early intrusion detection efforts were concentrated on developing a single-host system. Today's computing environment is increasingly becoming heavily networked. Actually, there are certain types of attacks, which are considered as normal by the single-host intrusion detection systems, but if looked at from a networked perspective, might be indicative of intrusive activity.

A Single-host IDS would normally not become suspicious if a single failed logon occurred. If, however, a series of single logon failures, all to the same account but on different systems, was observed, this might indeed be indicative of an individual attempting to gain access to the network and its computer systems.

2.4.4.1 NSM

The Network Security Monitor (NSM) was developed atom lithography the University of California, Davis, and was originally based on simple traffic analysis. It is designed to detect intrusive behavior in a network by exploiting the broadcast nature of a network in order to perform an analysis on the packet traffic. NSM's goal is to address three problems that exist in audit trail based IDSs. First networks are often of a heterogeneous nature. There may be several different types of systems connected to the net, each with different audit trail formats and features. Since there is little standardization among system audit trails, basing a network IDS on audit trail analysis would require comparison of possibly many different audit trail formats. A second problem with audit trails is that often they are simply turned off because they are expensive in terms of both storage space and CPU time. Finally, the audit trails themselves become a target of intruders who know that the audit trail may reveal their activities. By depending on an analysis of the network traffic instead of audit trails, NSM avoids these problems.

2.4.4.2 DIDS

The Distributed Intrusion detection System (DIDS) was developed by the United States Air Force by the same individuals at the University of California, Davis that were responsible for NSM. The purpose of DIDS is to monitor a heterogeneous network consisting of a series of monitored and unmonitored hosts.

2.4.4.3 NADIR

NADIR, the Network Anomaly Detection and Intrusion Reporter, is in use on the Los Alamos National Laboratory's Integrated Computing Network (ICN). Unlike several of the other systems; NADIR is not designed to be used in a variety of different environments but instead is designed to be used on the unique configuration that exists in the ICN.

2.5 Different Approaches toward Intrusion Detection

Here we talk about different approaches used by intrusion detection systems to represent knowledge on a system and analyze audit information in order to detect an intrusion. We shall concentrate on the most well known ones such as Artificial Intelligence techniques which include Artificial Neural Networks and Support Vector machines, Data Mining techniques and Wireless Ad hoc Networks in intrusion detection systems.

2.5.1 Artificial Intelligence

Two artificial intelligent techniques are studied: Artificial Neural Networks (ANNs) and Support Vector Machines (SVMs).

2.5.1.1 Artificial Neural Networks

An artificial Neural Network consists of a collection of treatments to transform a set of inputs to a set of searched outputs, through a set of simple processing units, or nodes and connections between them. Subsets of the units are input nodes, output nodes, and nodes between input and output form hidden layers; the connection between two units has some weight, used to determine how much one unit will affect the other. Two types of architecture of Neural Networks can be distinguished:

1. Supervised training algorithms, where in the learning phase, the network learns the desired output for a given input or pattern. The well-known architecture of
2. supervised neural network is the Multi-Level Perceptron (MLP); the MLP is employed for Pattern Recognition problems.
3. Unsupervised training algorithms, where in the learning phase, the network learns without specifying desired output. Self-Organizing Maps (SOM) are popular unsupervised training algorithms; a SOM tries to find a topological mapping from the input space to clusters. SOM are employed for classification problems.

The most important property of a Neural Network is to automatically learn/retrain coefficients in the Neural Network from the data inputs and data outputs. When applying the Neural Network approach to Intrusion Detection, we first have to expose Neural Networks to normal data and then to attacks to automatically adjust coefficients of the Neural Networks during the training phase. Performance tests are then finally conducted with real network traffic and attacks.

Neural Networks have been largely employed with success for complex problems such as Pattern Recognition, hand-written character recognition, Statistical Analysis.

Application of Neural Networks in Misuse Detection

Rule-based intrusion detection has been considered as effective if the exact characteristics of the attack are known. However, network intrusions are constantly changing because of individual approaches taken by attackers and regular changes in the software and hardware of the targeted systems. Because of the infinite variety of attacks and attackers even a dedicated effort to constantly update the rule base of an expert system can never hope to accurately identify the variety of intrusions.

Rapidly changing nature of network attacks requires a flexible defensive system that is capable of analyzing the enormous amount of network traffic in a manner, which is less structured than rule-based systems. A neural network-based misuse detection system could potentially address many of the problems that are found in rule-based systems.

Advantages of Neural Network-based Misuse Detection Systems

The first advantage in the utilization of a neural network in the detection of instances of misuse would be the flexibility that the network would provide. Even if the data is incomplete or distorted, neural network would be capable of analyzing the data from the network. In the same way, the network would possess the ability to conduct an analysis with data in a non-linear fashion.

Another advantage of neural networks is its inherent speed. Because the protection of computing resources requires the timely identification of attacks, the processing speed of the neural network could enable intrusion responses to be conducted before irreparable damage occurs to the system.

The most important advantage of neural networks in misuse detection is the ability of the neural network to "learn" the characteristics of misuse attacks and to identify instances that are unlike any which have been observed before by the network. Also neural network might be trained to recognize known suspicious events with a high degree of accuracy.

Disadvantages of Neural Network-based Misuse Detection Systems

Primarily there are two reasons why neural networks have not been applied to the problem of misuse detection in the past. The first reason relates to the training requirements of the neural network. Because the ability of the artificial neural network to identify indications of an intrusion is completely dependent on the accurate training of the system, the training data and the training methods that are used are very important.

The training routine requires a very large amount of data to ensure that the results are accurate. Also the training of a neural network for misuse detection purposes, may require thousands of individual attacks sequences, and it is very difficult to obtain large quantities of sensitive information.

The most significant disadvantage of applying neural networks to intrusion detection is the "black box" nature of the neural network. Unlike expert systems which have hard-coded rules for the analysis of events, neural networks adapt their analysis of data in response to the training which is conducted on the network. The connection weights and transfer functions of the various network nodes are usually frozen after the network has achieved an acceptable level of success in the identification of events.

Support Vector Machines

Suppon vector machines, or SVMs, are learning machines that plot the training vectors in high dimensional feature space, labeling each vector by its class. SVMs classify data by determining a set of support vectors, which are members of the set of training inputs that outline a hyper plane in the feature space.

SVMs provide a generic mechanism to fit the surface of the hyper plane to the data through the use of a kernel function. The user may provide a function (e.g., linear, polynomial, or sigmoid) to the SVMs during the training process, which selects support vectors along the surface of this function. The number of free parameters used in the SVMs depends on the margin that separates the data points, but not on the number of input features. These SVMs do not require a reduction in the number of features in order to avoid over fitting which is an advantage in intrusion detection. Another primary advantage of SVMs is the low expected probability of generalization errors.

There are other reasons that we use SVMs for intrusion detection. The first reason is speed; because real-time performance is of primary importance to intrusion detection systems, any classifier that can potentially run "fast" is worth considering. The second reason is scalability; SVMs are relatively insensitive to the number of data points and the classification complexity does not depend on the dimensionality of the feature space, so they can potentially learn a larger set of patterns and scale better than neural networks.

Data Mining

Across all industry sectors and scientific research areas, the amount of data collected and warehoused is growing at an explosive rate. However, it is believed hat less than 10% of the stores data has ever been retrieved and analyzed. The reason is that it is easy and cheap to store the data but difficult and expensive to make good use of the vast amount of data. Since manual approaches are obviously impractical given the sheer volume of data and the demand for fast analysis results, new techniques are being discovered to intelligently assist humans in discovering useful knowledge from the database. These techniques are the subject of the growing field of knowledge discovery in databases (KDD). KDD can be defined as "the nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data". Data mining is a particular step in the process in which specific algorithms are applied to extract patterns from data.

Data Mining: What is it?

Data mining is pattern finding. Data miners are experts at using specialized software to find regularities (and irregularities) in large data sets. Here are a few specific things that data mining might contribute to an intrusion detection project:

- Remove normal activity from alarm data to allow analysts to focus on real attacks
- Identify false alarm generators and "bad' sensor signatures
- Find anomalous activity that uncovers a real attack
- Identify long, ongoing patterns (different IP address, same activity)

To perform these tasks, data miners use one or more of the following techniques:

- Data summarization with statistics, including finding outliers
- Visualization: presenting a graphical summary of the data
- Clustering of the data into natural categories

- Association rule discovery: defining normal activity and enabling the discovery of anomalies.
- Classification: predicting the category to which a particular record belongs

Data Mining Algorithms

There are a wide variety of data mining algorithms, drawn from the fields of statistics, pattern recognition, machine learning, and databases.

Classification: classifies a data item into one of several pre-defined categories. These algorithms normally output "classifiers", for example, in the form of decision trees or rules. An ideal application in intrusion detection would be to gather sufficient "normal" and "abnormal" audit data for a user or a program, then apply a classification algorithm to learn a classifier that can label or predict new unseen audit data as belonging to the normal class or the abnormal class.

Link analysis: determines relations between fields in the database records. Correlations of system features in audit data, for example, the correlation between command and argument in the shell command history data of a user, can serve as the basis for constructing normal usage profiles.

Sequence analysis: models sequential patterns. These algorithms can discover what time-based sequences of audit events are frequently occurring together. These frequent event patterns provide guidelines for incorporating temporal and statistical measures into intrusion detection models. For example, patterns from audit data containing network-based denial-of-service (DOS) attacks suggest that several per-host and per-service measures should be included.

The Data mining process of Building Intrusion Detection Models

With the recent rapid development in KDD, we have gained a better understanding of the techniques and process frameworks that can support systematic data analysis on the vast amount of audit data. The process of using data mining approaches to build intrusion detection models is explained here.

Raw (binary) audit data is first processed into ASCII network packet information (or host event data), which is in turn summarized into connection records (or host session records) containing a number of within-connection features, e.g., service, duration, flag (indicating the normal or error status according to the protocols), etc. Data mining programs are then applied to the connection records to compute the frequent patterns i.e. association rules and frequent episodes, which are in turn analyzed to construct additional features for the connection records. Classification programs are then used to inductively learn the detection model. This process is of course iterative. For example, poor performance of the classification models often indicates that more pattern mining and feature construction is needed.

2.6 Data Reduction for Intrusion Detection

Intrusion Detection Systems (IDS) have become important and widely used tools for ensuring network security. Since the amount of audit data that an IDS needs to examine is very large even for a small network, classification by hand is impossible. Analysis is difficult even with computer assistance because extraneous features can make it harder to detect suspicious behavior patterns. Complex relationships exist between the features, which are difficult for humans to discover. IDS must therefore reduce the amount of data to be processed. This is very important if real-time detection is desired. Therefore, some form of data reduction is required for IDSs. Reduction can occur in one of several ways. Data that is not considered useful can be filtered, leaving only the potentially interesting data. Data can be grouped or clustered to reveal hidden patterns; by storing the characteristics of the clusters instead of the data, overhead can be reduced. Finally, some data sources can be eliminated using feature selection.

2.6.1 Data Filtering

The purpose of data filtering is to reduce the amount of data directly handled by the IDS. Some data may not be useful to the IDS and thus can be eliminated before processing. This has the advantage of decreasing storage requirements and reducing processing time. However, filtering may throw out useful data, and so must be done carefully.

2.6.2 Feature Selection

In complex classification domains, some data may hinder the classification process. Features may contain false correlations, which hinder the process of detecting intrusions. Further, some features may be redundant since the information they add is contained in other features. Extra features can increase computation time, and can impact the accuracy of IDS. Feature selection improves classification by searching for the subset of features, which best classifies the training data.

2.6.3 Data Clustering

Clustering can be performed to find hidden patterns in data and significant features for use in detection. Clustering can also be used as a reduction technique by storing the characteristics of the clusters instead of the actual data.

A Number of experiments have been performed to measure the performance of support vector machines and neural networks in intrusion detection, using the DARPA data for intrusion evaluation[MTTJ02]. Classifications were performed on the binary (normal/attack) as well as five-class classifications (normal, and four classes of attacks). It has been demonstrated that a large number of the (41) input features are unimportant and may be eliminated, without significantly lowering the performance of the IDS[MTTJ02]. Also the results showed that both SVMs and neural networks deliver highly accurate (99% and higher) performance, with SVMs showing slightly better performance[MTTJ02].

Further, when reduction was performed to reduce the 41 features to the 13 most significant, both SVMs and neural networks again were able to train to deliver accurate results for binary classification. In terms of the five-class classification, they found using only 19 most important (of the 41) features; the change in accuracy was statistically insignificant. But the reduction in features was expected to reduce the cost of detection and the overhead of the intrusion detection as a whole.

Chapter 3

DATA REDUCTION AND DATA CLASSIFICATION FOR INTRUSION DETECTION MODELS

3.1 Independent Component Analysis

Definition of ICA:

Let us assume that we have n linear mixtures $x_1 \ldots x_n$, of n independent components $x_j = a_{j1}s_1 + a_{j2}s_2 + a_{j3}s_3 + \ldots + a_{jn}s_n$ for all j.

We assume that the mixture x_j and each independent component s_k are random variables, so that we can drop the time index t and s_k need not be a proper time signal. The observed values x_j (t) (for e.g., the microphone signals of the cocktail party problem) are then a sample of this random variable. Using the vector notation

$$x \quad As$$

which can also be written as

$$x \cdot \Sigma a_j s_i$$

the statistical model in above equation is called independent component analysis or ICA model [Hyvarinen]. The ICA model is a generative model, which means that it describes how the observed data are generated by a process of mixing the components s_i. The independent components are latent variables, meaning that they cannot be directly observed. Also the mixing matrix is assumed to be unknown. All we observe is the random vector x, and we estimate both A and s using it. This is done under as the following general assumptions as possible.

Assumptions

- The number of sensors is greater than or equal to the number of sources $N \geq M$.
- The sources s (t) are at each time instant mutually independent.
- At most one source is normally distributed.
- No sensor noise or only low additive noise signals are permitted.
- The independent component must have nongaussian distributions.

Definition:

Independent Variables:

Consider two scalar-valued random variables y_1 and y_2. The variables y_1 and y_2 are said to be independent if information on the value of y_1 does not give any information on the value of y_2, and vice versa. Above, we noted that this is the case with the variables s_1 and s_2 but not with the mixture variables x_1, x_2.

Technically, independence can be defined by the probability densities. Let us denote by $p(y_1, y_2)$ the joint probability function (pdf) of y_1 and y_2. Let us further denote the marginal pdf of y_1 as $p_1(y_1)$, i.e. the pdf of y_1 is considered alone:

$$p_1(y_1) = \int p(y_1, y_2)\, dy_2,$$

and similarly for y_2. Then we define that y_1 and y_2 are independent if and only if the joint pdf is factorizable in the following way:

$$p(y_1, y_2) = p_1(y_1)\, p_2(y_2)$$

This definition extends naturally for any number n of random variables, in which case the joint density must be a product of n terms.

Also uncorrelated variables are only partly independent. Two random variables y_1 and y_2 are said to be uncorrelated, if their covariance is zero:

$$E\{y_1 y_2\} - E\{y_1\} E\{y_2\} = 0$$

If the variables are independent, they are uncorrelated but in the other way, uncorrelated ness does not imply independence.

Principles of ICA algorithm

1. **Preprocessing for ICA** – Before applying an ICA algorithm on the dataset, it is usually very useful to do some preprocessing for the dataset. We discuss below some preprocessing techniques that make the problem of ICA estimation much simpler and better.

Centering: The most basic and necessary preprocessing is to center x, the observed variable, i.e. subtract its mean vector $m = E\{x\}$ so as to make x a zero-mean variable.

This preprocessing is made to simplify the ICA algorithms: It does not mean that the mean could not be estimated. After estimating the mixing matrix A with centered data, we complete the estimation by adding the mean vector of s back to the centered estimates of s.

Whitening: Another useful preprocessing strategy in ICA is to first whiten the observed variables. This means that before the application of ICA algorithm (and
after centering), we transform the observed vector x linearly so that a new vector $\tilde{x}$ which is white, i.e. its components are uncorrelated and their variances equal unity.

Whitening reduces the number of parameters to be estimated. Instead of having to estimate the n^2 parameters that are elements of the orthogonal matrix A, we only need to estimate the new, orthogonal maxing matrix $\tilde{A}$. An orthogonal matrix contains n (n-1)/2 degrees of freedom. Whitening solves half of the problem of ICA. Because whitening is a very simple and standard procedure, much simpler than any ICA algorithms, it is a good idea to reduce the complexity. Also it is quite useful to reduce the dimension of the dataset at the same time as we do the whitening.

2. **"Nongaussian is independent"** – The key to estimating the ICA model is measuring the nongaussianity. According to the classical statistical theory, random variables are assumed to have Gaussian distributions.

3. **Measures of nongaussianity** – To use nongaussianity in ICA estimation, we must have a quantitative measure of a random variable, say y. To simplify, let us assume that y is centered (zero-mean) and has variance equal to one. Preprocessing the dataset before this makes this simplification possible.

Kurtosis: The classical measure of nongaussianity is kurtosis or the fourth-order cumulant. The kurtosis of y is classically defined by

$$\text{kurt}(y) = E\{y^4\} - 3(E\{y^2\})^2$$

Actually, since we assumed that y is of unit variance, the right-hand side simplifies to $E\{y^4\} - 3$. This shows that kurtosis is simply a normalized version of the fourth moment $E\{y^4\}$. For a Gaussian y, the fourth moment equals $3(E\{y^2\})^2$. Thus, kurtosis is zero for a Gaussian random variable. For most (but not all) no Gaussian random variables, kurtosis is nonzero.

Kurtosis can be both positive and negative. Random variables that have a negative kurtosis are called sub Gaussian, and those with positive kurtosis are called super Gaussian.

Negentropy: A second very important measure of nongaussianity is negentropy. Negentropy is based on the information-theoretic quantity of (differential) entropy.

Entropy is the basic concept of information theory. The entropy of a random variable is defined as the degree of information that the observation of the variable gives. The more random i.e. unpredictable and unstructured the variable is, the larger its entropy.

Entropy H is defined for a discrete random variable Y as

$$H(Y) = -\sum P(Y = a_i) \log P(Y = a_i)$$

where the a_i are the possible values of Y. This definition is generalized for continuous-valued random variables and vectors, in which case it is often called differential entropy. The differential entropy H of a random vector y with density f(y) is defined as:

$$H(y) = -\int f(y) \log f(y)\, dy.$$

A fundamental result of information theory is that a Gaussian variable has the largest entropy among all random variables of equal variance. This means that entropy could be used as a measure of nongaussianity.

To obtain a measure of nongaussianity that is zero for a Gaussian variable and always nonnegative, a slightly modified version of the definition of differential entropy, called negentropy is used. Negentropy J is defined as follows:

$$J(y) = H(y_{gauss}) - H(y)$$

where y_{gauss} is a Gaussian random variable of the same covariance matrix as y. From the above-mentioned properties, negentropy is always non-negative, and it is zero if and only if y has a Gaussian distribution.

4. **Minimization of Mutual Information** – Mutual information is the natural information-theoretic measure of the independence of random variables. In this approach that is an alternative to the model estimation approach, we define the ICA of a random vector x as an invertible transformation i.e. s = Wx where W is the inverse matrix of A, where the matrix W is determined so that the mutual information of the transformed components s_i is minimized.

5. **Maximum Likelihood estimation** – Another very popular approach for estimating the ICA model is maximum likelihood estimation, which is closely connected to infomax principle. It is equivalent to minimization of mutual information.

3.2 Principal Component analysis

Principal component analysis is applied in situations where, the dimension of the dataset is large, and also the components of the dataset are highly correlated (redundant).

Principal component analysis is useful in this situation to reduce the dimension of the input vectors. This algorithm has three effects.

- It orthogonalizes the components of the input vectors (so that they are uncorrelated with each other);
- It orders the resulting orthogonal components (principal components) so that those with the largest variation come first;

- It eliminates those components that contribute the least to the variation in the data set.

3.3 Independent Component Analysis and Principal Component Analysis for Intrusion Detection Data Reduction:

The intrusion detection data is obtained from audit trails or from packets on a network. The audit trail sometimes contains more data than needed for intrusion detection. Inspecting large amounts of audit data generated requires large effort and it is computationally expensive. ICA and PCA algorithms solve this problem by data reduction. The ICA and PCA algorithms analyze whether the 41 variables of the DARPA dataset, are really important for intrusion detection to classify attacks, are independent and try to minimize the variables, which are correlated thereby reducing the computational time and extraneous work.

3.4 Bayesian Belief Network

The Bayesian network is a powerful knowledge representation and reasoning algorithm under conditions of uncertainty. A Bayesian network B = (N, A, Θ) is a directed acyclic graph (DAG) (N, A) where each node n ∈ N represents a domain variable (e.g. a dataset attribute or variable), and each arc $a \in A$ between nodes represents a probabilistic dependency among the variables, quantified using a conditional probability distribution (CP table) $\theta_i \in \Theta$ for each node n_i (see Pearl 1988). A BN can be used to compute the conditional probability of one node, given values assigned to the other nodes.

Many Bayesian network structure-learning algorithms have been developed. These algorithms generally fall into two groups, search & scoring based algorithms and dependency analysis based algorithms. Although some of these algorithms can give good results on some benchmark data sets, there are still several problems such as Node ordering requirement, lack of efficiency, lack of publicly available learning tools.

In order to resolve these problems, two algorithms such as Algorithm A and Algorithm B have been developed in the area of Bayesian network structure learning. Algorithm A deals with a special case where the node ordering is given, which requires O (N^2) CI tests and is correct given that the underlying model is DAG faithful. Algorithm B deals with the general case and requires O (N^4) CI tests and is correct given that the underlying model is monotone DAG faithful. Based on these two algorithms, Bayesian network learning system, called Bayesian Network PowerConstructor has been developed.

Major advantage of Bayesian networks over many other types of predictive models, such as neural networks and decision trees, is that unlike those "black box" approaches, the Bayesian network structure represents the inter-relationships among the dataset attributes. Human experts can easily understand the network structures and if necessary can easily modify them to obtain better predictive models. By adding decision nodes and utility nodes, BN models can also be extended to decision networks for decision analysis.

Several other advantages of Bayesian networks are explicit uncertainty characterization, fast and efficient computation, and quick training. They are highly adaptive and easy to build, and provide explicit representation of domain specific knowledge in human reasoning framework. Also Bayes networks offer good generalization with limited training data, easy maintenance when adding new features or new training data.

Feature Selection in Bayesian Belief Network:

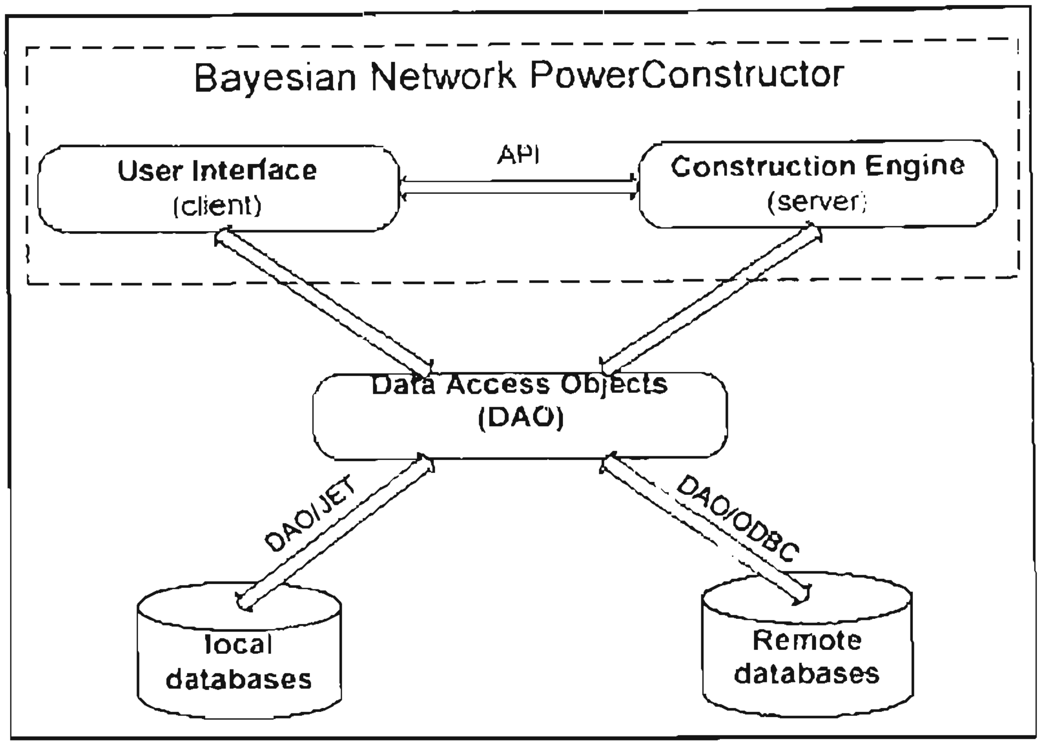
Reducing over fitting by considering only a subset of the features is called feature selection. A general Bayesian network classifier learning is that we can get a set of features that are on the Markov blanket of the class node. The Markov blanket of a node n is the union of n's parents, n's children and the parents of n's children. This subset of nodes shields n from being affected by any node outside the blanket. When using a BN classifier on complete data, the Markov blanket of the class node forms feature selection and all features outside the Markov blanket are deleted from the BN.

Bayesian Belief Network Software:

Bayesian Belief Network software includes Bayesian network PowerConstructor (BN PowerConstructor), Bayesian network PowerPredictor (BN PowerPredictor), and Data Preprocessor.

Bayesian Network PowerConstructor:

It has two components, a user-friendly interface and a construction engine. It runs under 32-bit windows systems (i.e., Windows 95, Windows 98 and Windows NT) on PCs. The system takes as input a database table and constructs a Bayesian network (both structure and parameters) as output. It also supports domain knowledge as additional input.


Bayesian Network PowerConstructor
User Interface (client)
API
Construction Engine (server)
Data Access Objects (DAO)
DAO/JET
DAO/ODBC
local databases
Remote databases

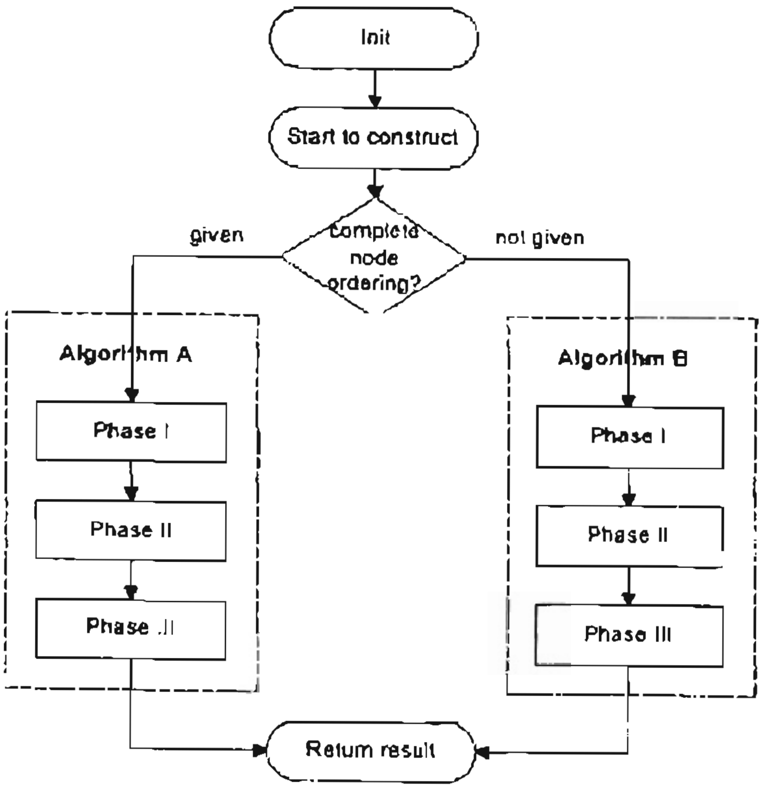
[CBL98] Figure 3.1. System structure of PowerConstructor

The structure of the system is shown in figure1, from which we can see that the two components of the system (user interface and construction engine) are in a client-server structure and both of them are connected to different kinds of databases through a standard interface, Data Access Objects.

The interface part of PowerConstructor is an executable file (BNPC.EXE). It first gathers the input information from the user using a five-step wizard. This information includes database formats, database location, data set name, domain knowledge etc. Next, the user interface calls the construction engine. When the construction engine finishes the process, the result is passed back as a parameter to the user interface.

Also we can see from the figure1 that both the user interface and the construction engine are connected to the data access objects, which provides a standard interface to access databases. To access local desktop databases like MS-Access, FoxPro, and Paradox, we use DAO/JET interface, which provides the database operation functions using the Microsoft Jet database engine. To access remote databases, we use DAO/ODBC direct interface, which passes commands directly to the remote database servers for processing. Because most of the workload is moved to the high-performance database server, this method can speed up the Bayesian network learning process and save lot of resources in the local computer.

The working mechanism of the construction engine is shown in the figure 3:



[CBL98] Figure 3.2. Working mechanism of the construction engine

Features of the PowerConstructor:

User Interface:

- Wizard-like interface: it gathers necessary input information through 5 simple steps.
- Online help: Online help is available for each step.
- Graphical belief network editor: it is available for modifying BN structure after the learning process.

Construction Engine:

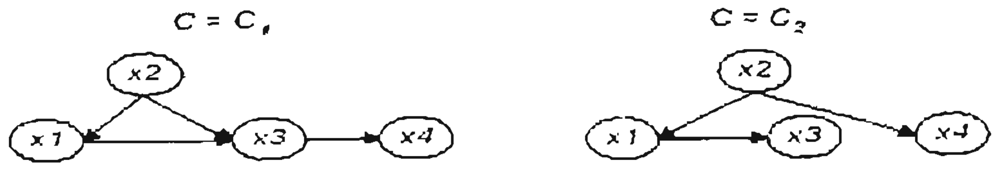
- Accessibility: The system supports most of the popular desktop database and spreadsheet formats, including Ms-Access, dBase, FoxPro, Paradox, Excel and text file formats. It also supports remote database servers like Oracle, SQL-server through ODBC.
- Reusability: The construction engine is an ActiveX DLL, so it can be easily integrated into other belief network, data mining or knowledge base systems for windows 95/98/NT.
- Efficiency: In theory it requires CI (conditional independence) tests to the complexity of O (N^4) without node ordering and O (N^2) when node ordering is given. (N is the number of attributes). In practice, the complexity is about O (N^2) even without node ordering.
- Supporting domain knowledge: Complete ordering. Partial ordering, direct causes and effects, forbidden links and root & leaf nodes can be used to constrain the search space and therefore speed up the construction process.

- Supporting large datasets: Running time is linear to the number of cases.
- Supporting condensed datasets, which has a frequency fields that contains the number of appearances of the current entry in the data.
- Connectivity: The resulting Belief network (both structure and parameters) can be exported to other belief network systems.

Bayesian Network PowerPredictor:

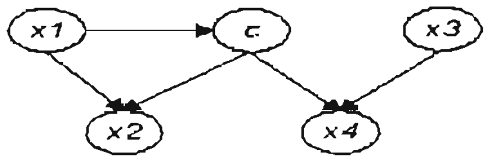
This is a data mining system for data modeling/classification/prediction. This system is an extension of our BN learning system (BN PowerConstructor) to BN based classifier learning and using. It learns general Bayesian network classifiers and Bayes multi-net classifiers from training data and uses these classifiers to classify new data. The system can also perform feature subset selection automatically.

Bayesian Multi-net: A Bayesian multi-net consists of the prior probability distribution of the class node and a set of local networks, each corresponding to a value that the class node can take. Bayesian multi-net allows the relations among the features to be different i.e., for different values the class node takes the features can form different local networks with different structures. This means, the class node can be also viewed as a parent of all the feature nodes since each local network is associated with a value of the class node. It is called as unrestricted BN classifier as it does not impose any restrictions on the relationships among attributes.



[CBL98] Figure 3.3. A simple Bayesian Multi-net

General Bayesian Network: It is another kind of unrestricted BN classifier. GBN treats the class nodes as an ordinary node and it is not necessary that class node is a parent of all the feature nodes.



[CBL98] Figure 3.4. A simple GBN

Comparison: On comparing Bayesian multi-nets and General Bayesian network, it is observed that GBNs assume that there is a single probabilistic dependency structure for the entire dataset; by contrast, multi-nets allow different probabilistic dependencies for different values of the class node. So, GBN classifiers work better when there is a single underlying model of the dataset and multi-net classifier work better when the underlying relationships among the features are very different for different classes.

Functions of BN PowerPredictor:

1. Learn a new general BN classifier or Bayesian multi-net classifier (with or without auto feature subset selection) from training data.
2. Modify an existing BN based classifier.
3. Use an existing BN based classifier to classify new data.

Features of the BN PowerPredictor:

User Interface:

- Wizard-like interface: it gathers necessary input information from the given training dataset to learn a BN classifier and then measure the classification accuracy on the given test dataset.
- Online help: Online help is available for each step.
- Graphical belief network editor: it is available for modifying BN classifier's structure.

Classifier Learning Engine:

- Wrapper algorithm. The system can automatically learn classifiers of different types and different complexities and choose the best one. (Performance measure: prediction accuracy and rate of complexity)
- Feature subset selection. The system can automatically perform feature subset selection.
- Two classification modes. The classification can be performed in either batch mode (a data set) or interactive mode (an instance).
- Efficiency. The system is based on fast BN learning algorithms.
- Supporting domain knowledge. Complete ordering, partial ordering, direct causes and effects, forbidden links and root & leaf nodes can be used to constrain the search space and therefore speed up the learning process.
- Supporting condensed data sets, which have 'frequency' fields that contain the number of appearances of the current entry in the data.
- Supporting misclassification cost table definition.

Data PreProcessor: It is a tool used with BN PowerConstructor and BN PowerPredictor for pre-processing the training data. The training data sets are processed by this data preprocessor before BN PowerPredictor can use them.

Functions:

1. Converting data from other desktop database formats to Microsoft JET/Access (*.MDB) format (as required by BN PowerPredictor).
2. Detecting and discretizing data fields that contain continuous data.
3. Dividing the training data into internal training set and internal test set (as required by BN PowerPredictor).

Features:

- Wizard-like interface. It gathers necessary input information through 4 simple steps.
- Accessibility. It supports most of the popular desktop database and spreadsheet formats, including: Ms-Access, dBase, FoxPro, Paradox, Excel and text file formats. It also supports remote database servers like ORACLE, SQL-SERVER through ODBC.
- Continuous field detection. Automatically detect fields with continuous values.

3.5 Classification and Regression Trees

CART is an acronym for Classification and Regression Trees, a decision tree procedure introduced in 1984 by world-renowned UC Berkeley and Stanford statisticians, Leo Breiman, Jerome Friedman, Richard Olshen, and Charles Stone. A decision tree is a flow chart or diagram representing a classification system or predictive model. The classification and regression trees methodology solves number of performance, accuracy, and operational problems that still cannot be solved by many decision-tree methods.

The Classification and regression trees methodology is technically called as binary recursive partitioning. The process is binary because parent nodes are always split into exactly two child nodes and recursive because the process is repeated by treating each child node as a parent. The key elements of a Classification and regression trees analysis are a set of rules for:

1. splitting each node in a tree;
2. deciding when tree is complete; and
3. assigning a class outcome to each terminal node (or predicted valuc for regression).

To split a node into two child nodes, CART always asks questions that have a "yes" or "no" answer. CART's method is to look at all possible splits for all variables included in the analysis. For example, consider the DARPA data set with 5092 cases and 41 variables. CART considers up to 5092 times 41 splits for a total of 208772 possible splits.

The default splitting rule used in CART is the GINI rule, essentially a measure of how well the splitting rule separates the classes contained in the parent node. (Alternative splitting criteria are also available). Once a best split is found, CART repeats the search process for each child node, continuing recursively until further splitting is impossible or stopped. Splitting is impossible if only one case remains in a particular node or if all the cases in that node are exact copies of each other or if a node has too few cases.

Once a terminal node is found, CART decides how to classify all cases falling within it. Because each node has the potential for being a terminal node, CART makes a class assignment for every node whether it is terminal or not.

Instead of attempting to decide whether a given node is terminal or not, CART proceeds by growing trees until it is not possible to grow them any further. Once CART has generated a maximal tree, it examines smaller trees obtained by pruning away branches of the maximal tree. Unlike other methods, CART does not stop in the middle of the tree-growing process, because there might still be important information to be discovered by drilling down several more levels.

Once the maximal tree is grown and a set of sub-trees are derived from it, CART determines the best tree by testing for error rates or costs. The misclassification error rate is calculated for the largest tree and also for every sub-tree. The best sub-tree is the one with the lowest or near-lowest cost, which may be a relatively small tree.

Advantages of CART compared to other decision tree algorithms:

- Reliable pruning strategy- CART's developers determined that no stopping rule could be relied on to discover the optimal tree, so they introduced the notion of over-growing trees and then pruning back; this idea ensures that important

- structure is not overlooked by stopping too soon. Other decision tree algorithms use problematic stopping rules.
- Powerful binary-split approach- CART's binary decision tress are more sparing with data and detect more structure before too little data is left for learning. Other decision-tree approaches use multi-way splits that fragment the data rapidly, making it difficult to detect rules that require broad ranges of data to discover.
- Automatic self-validation procedures- in the search for patterns in databases it is essential to avoid the trap of overfitting or finding patterns that apply only to the training data. CART's embedded test procedures ensure that the patterns found will hold up when applied to new data.
- Adjustable misclassification penalties help avoid the most costly errors.
- Multiple tree, committee-of-expert methods increase the precision of results, and;
- Alternative splitting criteria make progress when other criteria fail.

3.6 Variable importance in Classification and Regression Trees:

CART provides predictor ranking i.e. variable importance based on the contribution predictors make to the construction of the decision tree. Predictor rankings are relatively specific to the tree; by changing the tree we get different rankings. Importance is determined by the role of each predictor either as a main splitter or as a surrogate. Surrogate splitters are defined as back-up rules that closely mimic the action of primary splitting rules. Suppose that, in a given model, CART splits data according to household income. If a value for income is not available, CART might substitute education level as a good surrogate.

Variable importance, for a particular predictor is the sum across all nodes in the tree of the improvement scores that the predictor has when it acts as a primary or surrogate (but not competitor) splitter. Say for node i, if the predictor appears as the primary splitter then its contribution toward the importance as

importance_contribution_node_i = improvement

Instead if the predictor appears as the n'th surrogate instead of the primary predictor, then the expression is:

importance_contribution_node_i = (p ^ n) * improvement

in which p is the "surrogate improvement weight" which is a user controlled parameter which is equal to1.0 by default and can be set anywhere between 0 and 1. Thus we can specify that the surrogate splits contribute less towards a predictor's improvement than primary splits.

3.7 Classification and Regression trees and Bayesian Belief Network as Intrusion Detection Models

Intrusion detection can be considered as a classification problem where each connection or user is identified as one of the attack types or normal based on some existing data. CART and Bayesian belief network can solve this classification problem of intrusion detection as they learn the model from the data set and can classify the new data item into one of the classes specified in the data set. CART and Bayesian belief network can be used for misuse intrusion detection as they learn a model based on the training data and can predict the future data as one of the attack types or normal based on the learned model.

CART and Bayesian belief network give very good classification accuracy, even on smaller data sets (i.e. data sets after reduction), which is very useful as smaller data sets reduce the computational time for real-time intrusion detection. CART and Bayesian belief network are fast. This makes the system useful in real-time intrusion detection. CART and Bayesian Belief network construct easily interpretable models, which is useful for a security officer to inspect and edit. Generalization accuracy of CART and Bayesian belief network is another useful property for intrusion detection model. There will always be some new attacks on the system which are small variation of known attacks after the intrusion detection model is built. The ability to detect these new intrusions is possible due to the generalization accuracy of the CART and Bayesian belief network.

3.8 Audit data reduction for intrusion detection

Srinivas Mukkamala etal [MTTJ02] has done several simulations, including binary classifications (normal and attack) and five-class classifications (normal, and four classes of attacks) on the DARPA data set. He used the method of deleting one feature at a time to rank the importance of each feature towards the overall efficiency and effectiveness. He used neural networks for ranking the effectiveness. Considering performance as the basis he discovered that 19 features with feature names protocol_type, dst_bytes, num_compromised, root_shell, su_attempted, num root, num_file_creations, is_host_login, is_guest_login, count, srv_ count, srv_serror_ rate, srv_rerror_rate, diff_srv_rate, dst_host_same_src_port_rate, dst_host_srv_diff host_rate, dst_host_serror_rate, dst_host_srr_serror_rate and dst_host_rerror_rate which are labeled as 2, 6, 13, 14, 15, 16, 17, 21, 22, 23, 24, 26, 28, 30, 36, 37, 38, 39, 40 were important for detecting the attack and normal patterns for five-class classification.

3.9 Ensemble Approach

Empirical observations show that different classifiers provide complementary information about the patterns to be classified. Although for a particular problem one classifier works better than the other, a set of misclassified patterns would not necessarily overlap. This different information combined together yields better performance than individual classifiers. The idea is not to rely on single classifier for decision, instead different classifiers individual information is considered to take the final decision. We call this approach of combining different classifiers as ensemble approach. The effectiveness of the ensemble approach depends on the accuracy and diversity of the base classifiers.

Various techniques are developed for the ensemble approach [Die00], [KHDM98]. One technique is to use different training models for different base classifiers and then combine their outputs, another one uses different subset of features for different base classifiers and combines their outputs. In this approach we use the same data set as well as feature set for all the base classifiers and combine them to give the final output of the ensemble approach. We used the highest scored class as the final output among the base classifiers outputs. When the highest scored class fails, then the next highest scored class is given preference. The architecture of the ensemble approach is depicted in figure 3.5.

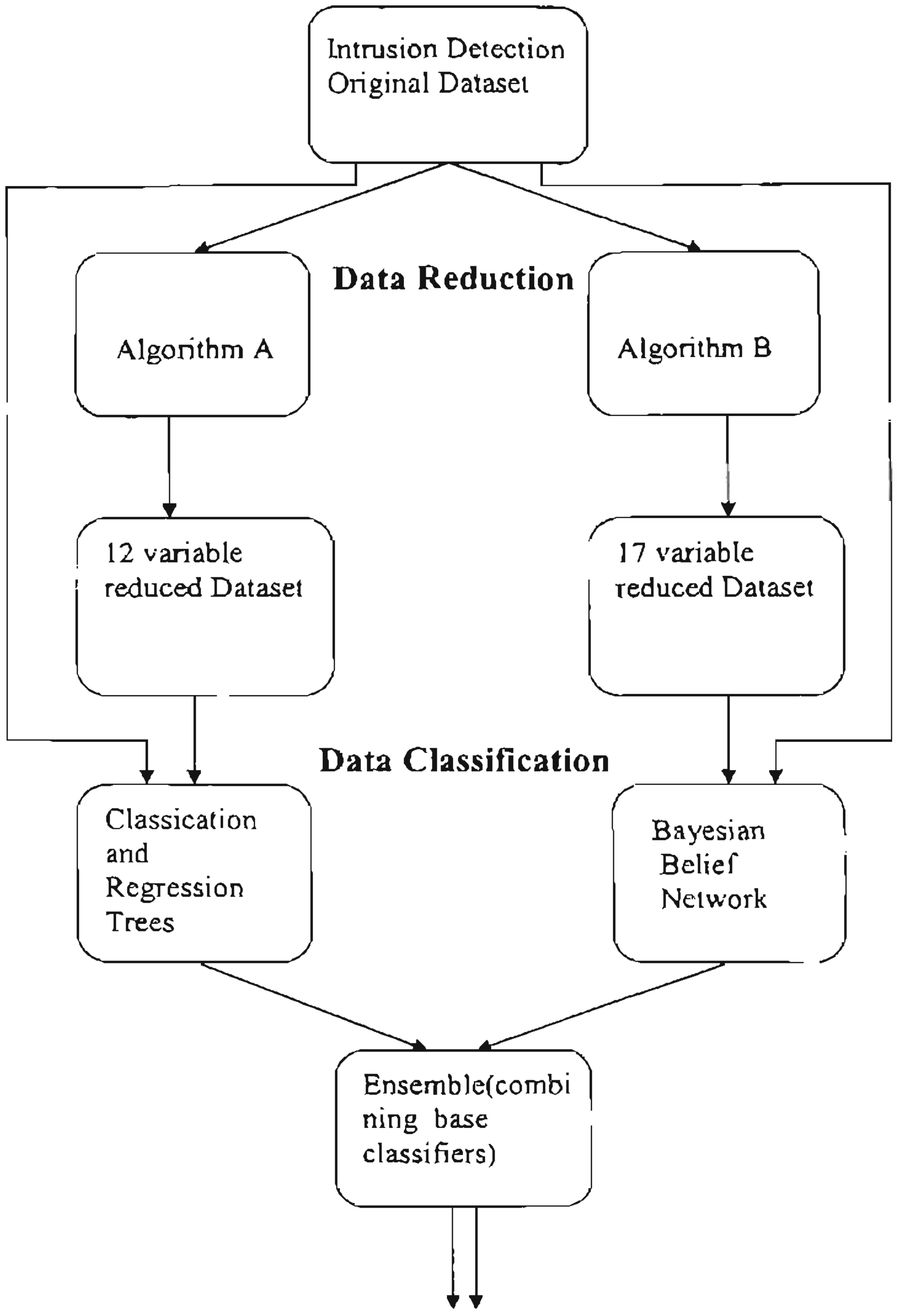

Intrusion Detection Original Dataset
Data Reduction
Algorithm A
Algorithm B
12 variable reduced Dataset
17 variable reduced Dataset
Data Classification
Classication and Regression Trees
Bayesian Belief Network
Ensemble(combining base classifiers)

Figure 3.5. Ensemble Approach Design

Algorithm A: CART Algorithm

Algorithm B: CBL2 Algorithm (Bayesian Network algorithm) on General Bayesian Network Classifier.

Chapter 4

EXPERIMENTATION SETUP AND PERFORMANCE EVALUATION

4.1 Intrusion Data

The KDD Cup 1999 Intrusion detection contest data [KDD99] is used in our experiments. This data was prepared by the 1998 DARPA Intrusion Detection Evaluation program by MIT Lincoln Labs [MIT]. Lincoln labs set up an environment of a local-area network (LAN) simulating a typical U.S. Air Force LAN and they operated the LAN as if it were a true Air Force environment. They acquired nine weeks of raw TCP dump data. The raw data was processed into connection records, which are about five million connection records. The data set contains 24 attack types. All these attacks fall into four main categories.

1. **Denial of Service (DOS):** In this type of attacks an attacker makes some computing or memory resources too busy or too full to handle legitimate requests, or denies legitimate users access to a machine. Examples are Apache2, Back, Land, Mailbomb, SYN Flood, Ping of death, Process table, Smurf, Teardrop.
2. **Remote to User (R2L):** In this type of attacks an attacker who does not have an account on a remote machine sends packets to that machine over a network and

exploits some vulnerability to gain local access as a user of that machine. Examples are Dictionary, Ftp_write, Guest, Imap, Named, Phf, Sendmail, Xlock.

3. **User to Root (U2R):** In this type of attacks an attacker starts out with access to normal user account on the system and exploits vulnerabilities to gain root access to the system. Examples are Eject, Load module, Ps, Xterm, Perl, Fdformat.
4. **Probing:** In this type of attacks an attacker scans a network of computers to gather information or find known vulnerabilities. An attacker with a map of machines and services that are available on a network can use this information to look for exploits. Examples are Ipsweep, Mscan, Saint, Satan, Nmap.

The original data contains 744 MB data with 4, 940,000 records. The data set has 41 attributes for each connection record plus one class label. Some features are derived features which are useful in distinguishing normal connection from attacks. These features are either nominal or numeric. Some features examine only the connections in the past two seconds that have the same destination host as the current connection, and calculate statistics related to protocol behavior, service, etc. These are called same host features. Some features examine only the connections in the past two seconds that have the same service as the current connection and are called same service features. Same host and same service features are together called time-based traffic features of the connection level records. Some other connection records were also sorted by destination host, and features were constructed using a window of 100 connections to the same host instead of a time window. These are called host-based traffic features. R2L and U2R attacks don't have any sequential patterns like DOS and Probe because the former attacks have the attacks embedded in the data packets whereas the later attacks have many connections in a short amount of time. So some features that look for suspicious behavior in the data packets like number of failed logins are constructed and these are called content features.

4.2 Experimentation Setup and Results Analysis

Our experiments have three phases namely data reduction, training phase and testing phase. In the data reduction phase, important variables for real-time intrusion detection are selected by feature selection. In the training phase, the system constructs a model using the training data to give maximum generalization accuracy on the unseen data. The test data is tested with the constructed model to detect the intrusion in the testing phase. The original data set has some 24 attack types which belong to four classes as described in section 4.1. The data set for our experiments contains 11982 records, which are randomly generated from the data set described in section 4.1. The 41 features are duration, protocol-type, service, flag, src_bytes, dst_bytes, land, wrong_fragment, urgent, hot, num_falied_logins, logged_in, num_compromised, root_shell, su_attempted, num_root, num_file_creations, num_shells, num_access_files, num_outbound_cmds, is_host_login, is_guest_login, count, srv_count, serror_rate, srv_serror_rate, rerror_rate, srv_rerror_rate, same_srv_rate, diff_srv_rate, srv_diff_host_rate, dst_host_count, dst_host_srv_count, dst_host_same_srv_rate, dst_host_diff_srv_rate, dst_host_same_src_port_rate, dst_host_srv_diff_host_rate, dst_host_serror_rate, dst_host_srv_serror_rate, dst_host_rerror_rate, dst_host_srv_rerror_rate and are labeled in order as A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z, AA, AB, AC, AD, AF, AG, AH, AJ, AJ, AK, AL, AM, AN, AO and the class label is named as AP. This data set has five different classes, random generation of data include the number of data from each class proportional to size, except that the smallest class is completely included. This data set is again divided into training data with 5092 records and testing with 6890 records. All the intrusion detection models are trained and tested with the same set of data. As the data set has five different classes we perform a 5-class classification. The normal data belongs to class1, probe belongs to class2, denial of service (DOS) belongs to class3, user to root (U2R) belongs to class4 and remote to local (R2L) belongs to class5. We used Independent component analysis algorithm and principal component analysis algorithm initially for data reduction. And then we used that reduced data set for classifying attacks. We found that ICA and PCA algorithms failed to perform feature selection carefully and it threw out useful data. So we used Classification and Regression Trees and Bayesian Belief Network for both data reduction and data classification. We arranged the data in text files for ICA and PCA algorithms present in MATLAB Software. For classification and regression trees and for Bayesian belief network algorithms data is saved on spreadsheets like Excel and some data base programs. We used AMD Athlon 1.67 GHz processor with 992 MB of RAM for our experiments.

4.2.1 Independent Component Analysis

We used the data set described in section 4.1 for evaluating the performance of ICA in data reduction. We did scaling and normalization on the intrusion detection data set. The objective is to separate the independent components that are uncorrelated with other features. On the scaled data set, it gave 2 independent components among 41 features. And then we used this reduced data set to check the performance accuracy. First the General Bayesian Network classifier is constructed using the training data and then testing data is tested with the constructed classifier to classify the data into normal or any of the remaining four attacks. Table 4.1 summarizes the results of the test data. It shows the training and testing times of the classifier in seconds for each of the five classes and their accuracy in percentage terms. As can be seen from the results, the accuracy of detection is very low.

	Training time (sec)	Testing time (sec)	Accuracy (%)
Normal	2.45	0.12	36.733
Probe	1.09	0.14	23.812
DOS	1.32	0.06	56.323
U2R	1.06	0.09	74.776
R2L	1.43	0.08	67.355

Table 4.1 Performance of General Bayesian Network Classifier on the ICA reduced 2 variable data set

On the normalized data set, it gave 39 independent components among 41 features. And then we used this reduced data set to check the performance accuracy. Table 4.2 summarizes the results of the test data. It shows the training and testing times of the classifier in seconds for each of the five classes and their accuracy in percentage terms.

	Training time (sec)	Testing time (sec)	Accuracy (%)
Normal	42.45	15.12	89.442
Probe	41.29	14.34	84.999
DOS	51.12	11.46	67.861
U2R	29.16	10.09	96.227
R2L	31.33	14.18	87.770

Table 4.2 Performance of General Bayesian Network Classifier on the ICA reduced 39 variable data set

4.2.2 Principal Component Analysis

We used the data set described in section 4.1 for evaluating the performance of PCA in data reduction. We eliminated the components of the data set that contribute less than one percent to the variation in the data set. So, the data set is left with 17 variables out of 41 variables. Then we used this reduced data set to check the performance accuracy. The General Bayesian Network classifier is trained using the training data and the trained network is then applied on the test data to classify the data into normal or attack patterns. The results are summarized in the Table 4.3. The training and testing times of the classifier are shown in seconds for each of the five classes and their accuracy is shown in percentage terms. The results show that the PCA reduced dataset yields poor accuracy.

	Training time (sec)	Testing time (sec)	Accuracy (%)
Normal	22.45	10.12	31.270
Probe	21.29	14.34	34.520
DOS	21.12	11.46	48.920
U2R	19.16	8.09	70.360
R2L	11.33	12.18	52.070

Table 4.3 Performance of General Bayesian Network Classifier on PCA reduced 17 variable data set

4.2.3 Bayesian Belief Network

We used the data set described in section 4.1 to evaluate the performance of Bayesian belief network. General Bayesian network classifier is constructed using the training data and then the classifier is used on the test data set to classify the data as an attack or normal. Table 4.4 shows the results by using the original 41 variable data set on the general Bayesian network classifier. Training time and testing time are shown in seconds for each of the five classes and their accuracy is shown in percentage terms.

	Training time (sec)	Testing time (sec)	Accuracy (%)
Normal	42.14	19.02	99.571
Probe	49.15	21.04	99.430
DOS	54.52	23.02	99.691
U2R	30.02	15.23	64.000
R2L	47.28	12.11	99.112

Table 4.4 Performance of Bayesian belief network on the 41 variable original intrusion detection data set.

We have done feature selection in Bayesian belief network and found out that 17 variables of the intrusion detection data set forms the Markov blanket of the class node as explained in section 3.4. These 17 variables are A, B, C, E, G, H, K, L, N, Q, V, W, X, Y, Z, AD, AF and these are considered the most important variables for intrusion detection by general Bayesian network classifier. Table 4.5 shows the performance of Bayesian belief network on the reduced data set i.e. the data set consisting of only A, B, C, E, G, H, K, L, N, Q, V, W, X, Y, Z, AD, AF variables and the class variable. It shows the training time and testing time in seconds and the performance accuracy in percentage terms.

	Training time (sec)	Testing time (sec)	Accuracy (%)
Normal	23.29	11.16	99.643
Probe	25.07	13.04	98.571
DOS	28.49	14.14	98.168
U2R	14.13	7.49	60.000
R2L	21.13	13.57	98.934

Table 4.5 Performance of Bayesian Belief Network on the 17 variable reduced data set.

Comparison of Bayesian belief network performance on the original data set and on the reduced data set:

Bayesian belief network performance is compared by using the original 41 variable data set and the 17 variable reduced data set. The training times and testing times for each classifier are decreased when the 17 variable data set is used. Moreover, by using the 17 variable data set there is a slight increase in the performance accuracy of Normal class over the 41 variable data set. For the other classes, Probe, DOS, U2R and R2L there is slight decrease in the performance accuracy.

We therefore conclude that the intrusion detection model built on the 17 variable data set is less computationally less expensive compared to the model built on the 41 variable data set and the normal data is classified better with the reduced data set. From the table 4.6 we can conclude that the Bayesian Belief network performs better on the reduced data set in terms of the computational time and in terms of the accuracy for class Normal.

Class	41 variable data set			17 variable data set		
	Training time(s)	Testing time(s)	Accuracy (%)	Training time(s)	Testing time(s)	Accuracy (%)
Normal	42.14	19.02	99.571	23.29	11.16	99.643
Probe	49.15	21.04	99.430	25.07	13.04	98.571
DOS	54.52	23.02	99.691	28.49	14.14	98.168
U2R	30.02	15.23	64.000	14.13	7.49	60.000
R2L	47.28	12.11	99.112	21.13	13.57	98.934

Table 4.6 Performance comparison of Bayesian Belief Network with the original 41 variable data set and the reduced 17 variable data set.

The graph in figure 4.1 shows the comparison of training times for the 41 variable original data set and the 17 variable reduced data set. The x-axis represents the number of classes present in the data set. The y-axis represents the training time taken for each class in seconds. The graph shows that the training time taken for the original dataset is more than the training time taken for 17 variable reduced dataset which means that the computational time will be reduced if we use the reduced data set for real-time intrusion detection.

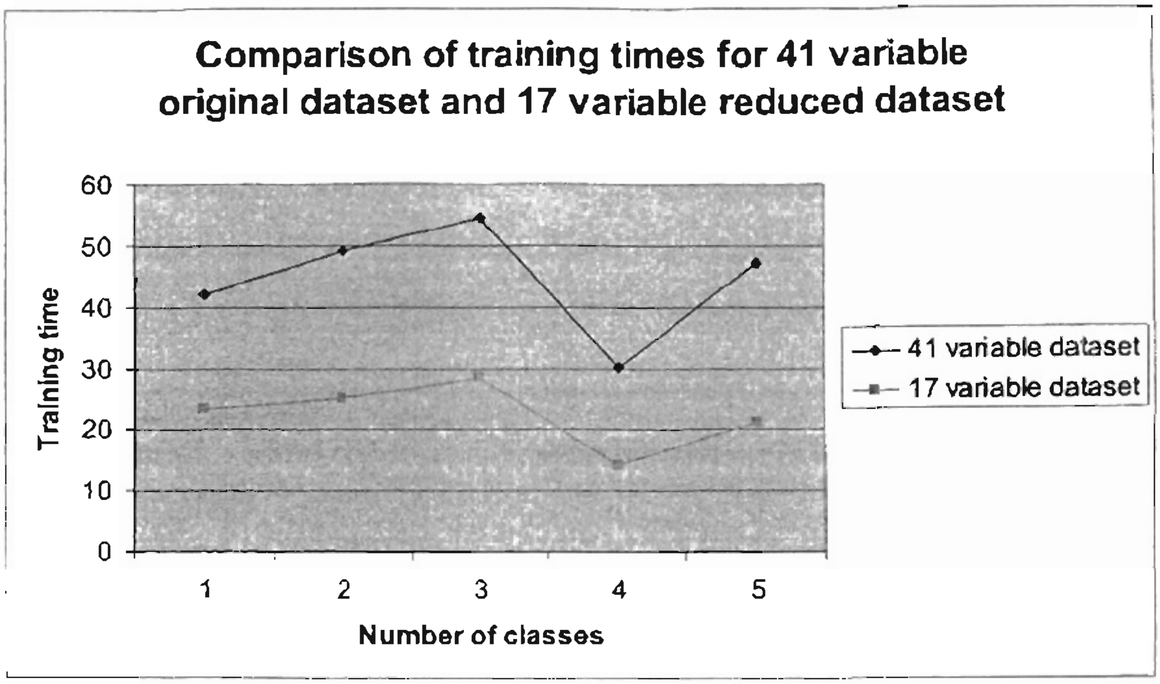


Figure 4.1 Comparison of Training times of Original dataset Vs 17 variable dataset

The graph in figure 4.2 shows the comparison of performance accuracies of the original data set and 17 variable reduced data set for the Normal class of data. Dataset of Normal class contains 1400 data points and as it is difficult to represent all of them in the graph 30 data points are used. The classification value of 1 in the graph represents a correct classification and value of 2 represents a misclassification. The graph shows that the reduced data set gives correct classification on 1 data point more compared to the original data set. Therefore, it is better to use the reduced data set for classification of normal data and also by using the reduced data set the computational time decreases thereby improving the performance.

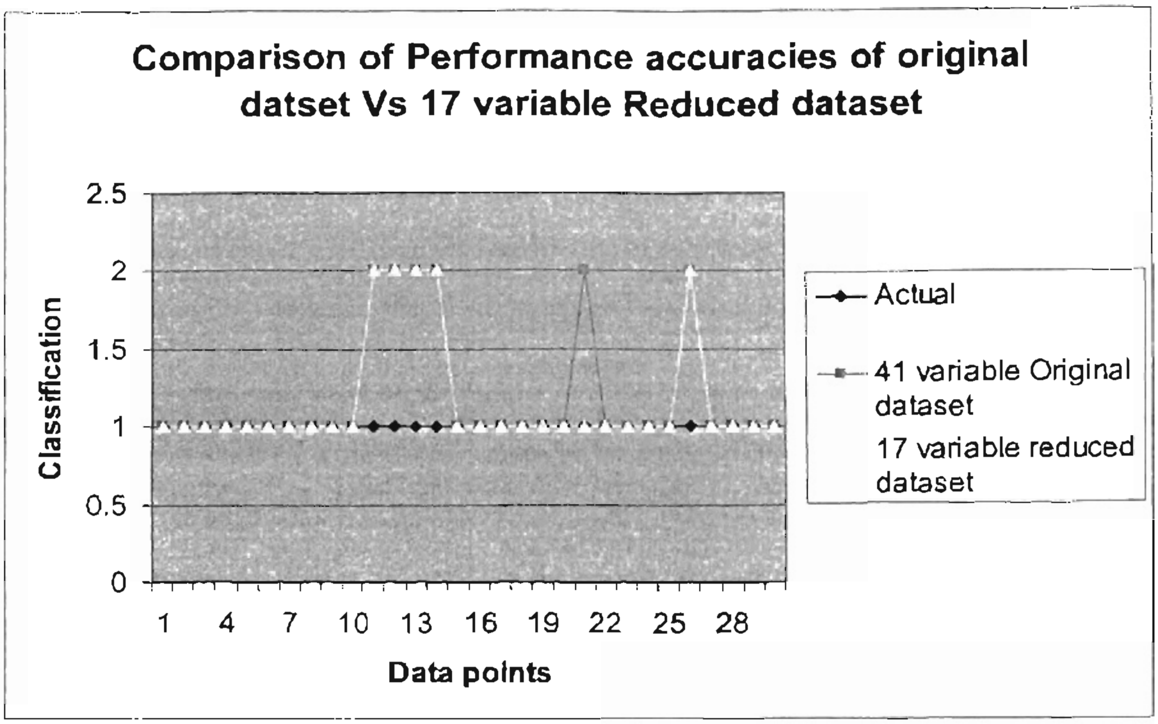


Figure 4.2 Comparison of Performance accuracies of Original dataset Vs reduced data set for the Normal class.

4.2.4 Classification and Regression Trees

We used the data set described in section 4.1 for measuring the performance of classification and regression trees. The Classifier is constructed on the training data and then the classifier is used on the test data to classify the data into normal or attack. Table 4.6 shows the performance of classification and regression trees on the 41 variable original data set. The training time and testing time for each classifier are shown in seconds and accuracy is shown in percentage terms.

	Training time (sec)	Testing time (sec)	Accuracy (%)
Normal	1.15	0.18	99.643
Probe	1.25	0.03	97.857
DOS	2.32	0.05	99.476
U2R	1.10	0.02	48.000
R2L	1.56	0.03	90.586

Table 4.7 Performance of Classification and Regression Trees on the 41 variable original data set

We decided the important variables by using the information provided by the Classification and regression trees predictor ranking. Predictor rankings are in terms of percentages. We eliminated the variables that have 0.00% rankings and considered only the primary splitters or surrogates as explained in section 3.6. This resulted in 12 variable data set that has C, E, F, L, W, X, Y, AB, AE. AF, AG and AI variables. Table 4.7 shows the performance accuracy of classification and regression trees on the 12 variable reduced data set. The training times and testing times for each classifier are shown in seconds and accuracy is shown in percentage terms.

	Training time (sec)	Testing time (sec)	Accuracy (%)
Normal	0.80	0.02	100.000
Probe	0.85	0.05	97.714
DOS	0.97	0.07	85.340
U2R	0.45	0.03	64.000
R2L	0.79	0.02	95.560

Table 4.8 Performance accuracy of classification and regression trees on the 12 variable data set.

Comparison of performance of classification and regression trees on the original data set and on the reduced data set:

Classification and regression trees performance accuracies are compared by using the 41 variable original data set and the 12 variable reduced data set. From table 4.8 we conclude that computational time is less for the 12 variable reduced data set which is beneficial for real-time intrusion detection. And also the Normal class is classified 100 percent correctly. Furthermore, the accuracies of classes U2R and R2L have increased by using the 12 variable data set. In the other classes such as Probe and DOS there is slight decrease in accuracy by using the 12 variable data set compared to the 41 variable data set. So Classification and regression trees therefore classify accurately on smaller data sets.

Class	41 variable data set			12 variable data set		
	Training time(s)	Testing time(s)	Accuracy (%)	Training time(s)	Testing time(s)	Accuracy (%)
Normal	1.15	0.18	99.643	0.80	0.02	100.000
Probe	1.25	0.03	97.857	0.85	0.05	97.714
DOS	2.32	0.05	99.476	0.97	0.07	85.340
U2R	1.10	0.02	48.000	0.45	0.03	64.000
R2L	1.56	0.03	90.586	0.79	0.02	95.560

Table 4.9 Performance comparison of classification and regression trees on 41 variable original data set and on the 12 variable reduced data set.

The graph in figure 4.3 shows the comparison of training times of original data set and 12 variable reduced data set. The x-axis represents the number of classes present in the data set and the y-axis represents the training time in seconds. The graph shows that the training time taken for original data set is more than the reduced data set. So by using reduced data set in real-time intrusion detection systems, the computational time will be reduced. So data reduction is necessary for real-time intrusion detection systems.

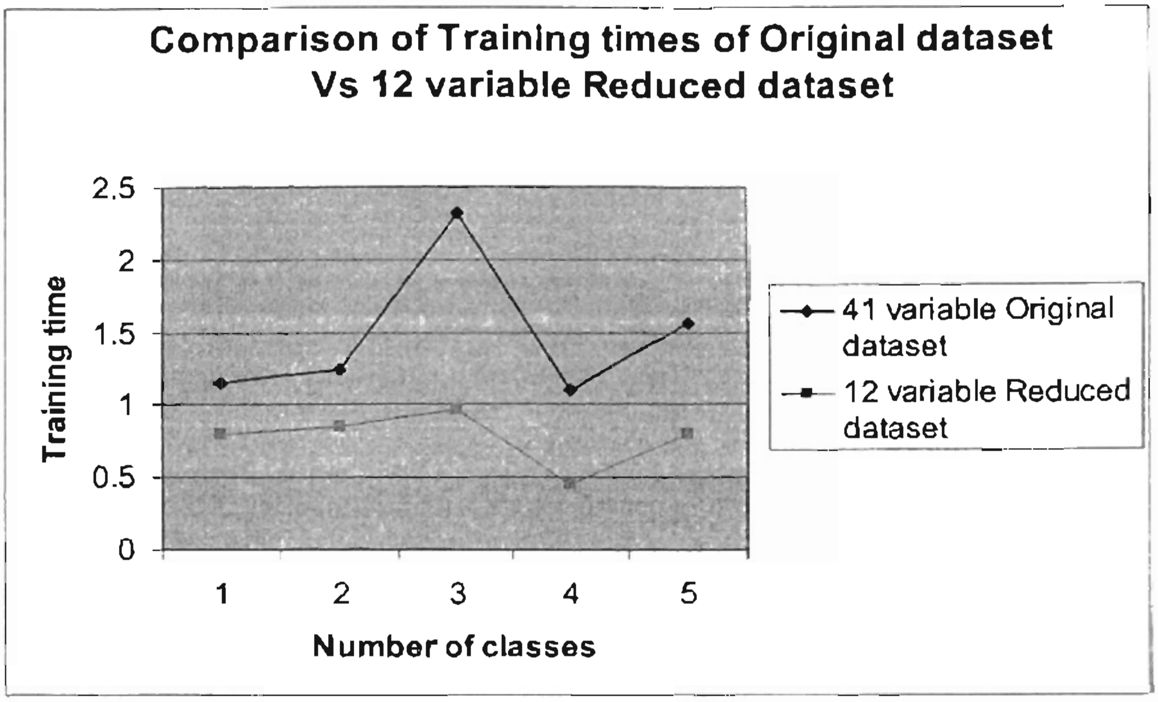


Figure 4.3 Comparison of Training times of Original dataset Vs 12 variable Reduced dataset

The graph in figure 4.4 shows the comparison of performance of original data set and 12 variable reduced data set in terms of accuracy for the U2R class of data. Data set of U2R class contains 25 data points and all the data points are represented in the graph. The classification value of 1 represents a correct classification and value of 2 represents a misclassification. The graph shows that 4 more data points are classified correctly by the reduced data set as compared to the original data set, which means that some variables in the original data set are hindering the process of detecting intrusions. Also using the reduced data set results in decreasing storage requirements and reduces the processing time.

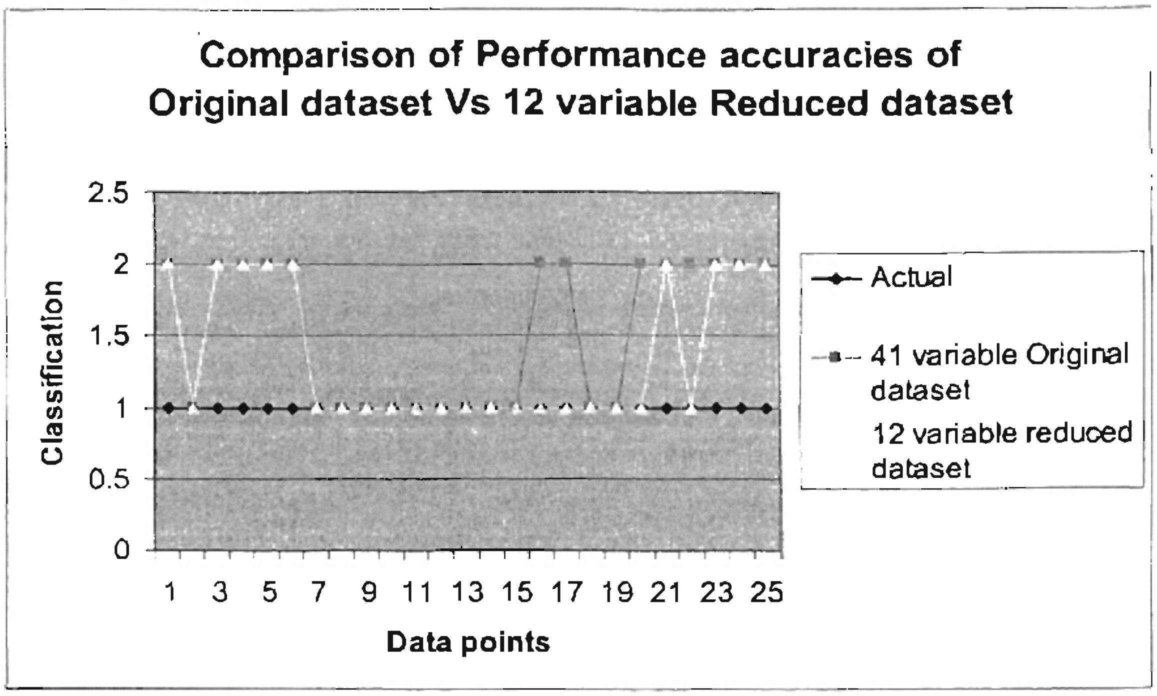


Figure 4.4 Comparison of Performance accuracies of Original dataset Vs 12 variable Reduced dataset for U2R class.

4.2.5 Comparing performances of General Bayesian network classifier and Classification and regression trees on 41 variable data set

The data set used here is the 41 variable original data set. Table 4.9 shows the performance comparisons of Bayesian belief network and classification and regression trees on the 41 variable data set. The training times and testing times are shown in seconds and accuracies are shown in percentage terms.

Class	Bayesian Belief Network			Classification and Regression Trees		
	Training time(s)	Testing time(s)	Accuracy (%)	Training time(s)	Testing time(s)	Accuracy (%)
Normal	42.14	19.02	99.571	1.15	0.18	99.643
Probe	49.15	21.04	99.430	1.25	0.03	99.857
DOS	54.52	23.02	99.691	2.32	0.05	99.476
U2R	30.02	15.23	64.000	1.10	0.02	48.000
R2L	47.28	12.11	99.112	1.56	0.03	90.586

Table 4.10 Comparison of performances of Bayesian BN and Classification and Regression trees on the 41 variable data set

From table 4.9 we can conclude that class Normal and class Probe are classified better by classification and regression trees algorithm. Whereas, classes DOS, U2R, and R2L are classified better by general Bayesian network classifier. Moreover, training times and testing times for each class are greater in the general Bayesian network classifier compared to classification and regression trees.

4.2.6 Comparing performances of General Bayesian network classifier and Classification and regression trees on 12 variable reduced data set

We used the 12 variable reduced data set that is obtained from the data reduction algorithm in classification and regression trees. Table 4.10 shows the performance comparisons of Bayesian belief network and classification and regression trees on the 12 variable reduced data set. The training times and testing times are shown in seconds and accuracies are shown in percentage terms.

Class	Bayesian Belief Network			Classification and Regression Trees		
	Training time(s)	Testing time(s)	Accuracy (%)	Training time(s)	Testing time(s)	Accuracy (%)
Normal	20.10	10.13	98.786	0.80	0.02	100.000
Probe	23.15	11.17	99.571	0.85	0.05	97.714
DOS	25.19	12.10	98.950	0.97	0.07	85.340
U2R	11.03	5.01	48.000	0.45	0.03	64.000
R2L	19.05	12.13	98.934	0.79	0.02	95.560

Table 4.11 Performance comparisons of Bayesian BN and Classification and Regression trees on the 12 variable reduced data set

From table 4.10 we can conclude that classification and regression trees classifies normal data very accurately. Also user-to-root attacks are classified better by · classification and regression trees compared to general Bayesian network classifier.

Probe attacks, denial-of-service attacks and remote-to-local attacks are classified better by the general BN classifier compared to classification and regression trees.

4.2.7 Comparing performances of General Bayesian network classifier and Classification and regression trees on 17 variable reduced data set

We used the 17 variable reduced data set that is obtained from feature selection in Bayesian belief network algorithm. Table 4.11 shows the performance comparisons of general BN classifier and classification and regression trees on the 17 variable reduced data set. The training times and testing times are shown in seconds and accuracies are shown in percentages.

Class	Bayesian Belief Network			Classification and Regression Trees		
	Training time(s)	Testing time(s)	Accuracy (%)	Training time(s)	Testing time(s)	Accuracy (%)
Normal	23.29	11.16	99.643	1.03	0.04	99.643
Probe	25.07	13.04	98.571	1.15	0.13	100.000
DOS	28.49	14.14	98.168	0.96	0.11	99.976
U2R	14.13	7.49	60.000	0.59	0.02	72.000
R2L	21.13	13.57	98.934	0.93	0.10	96.625

Table 4.12 Performance comparisons of general BN classifier and classification and regression trees on the 17 variable reduced data set.

From Table 4.11 we can conclude that normal data is classified by both intrusion detection systems to the same level. Probe attacks are classified more accurately by classification and regression trees. For class DOS and class U2R, classification and regression trees classify more accurately compared to general Bayesian network classifier. Class R2L is classified better by general Bayesian network classifier.

4.2.8 Performance comparisons of different reduced data sets

Table 4.12 and 4.13 shows the performance comparisons of classification and regression trees classifier and general Bayesian network classifier on 12, 17 and 19 variable reduced datasets. The 12 variable reduced data set is obtained from the data reduction in classification and regression trees. The 17 variable data set is obtained from the feature selection procedure in Bayesian BN whereas the 19 variable reduced dataset is the Srinivasan's etal [MTTJ02] reduced data set as explained in section 3.7.

	12 variable data set Accuracy(%)	17 variable data set Accuracy(%)	19 variable data set Accuracy (%)
Normal	100.000	99.643	95.500
Probe	97.714	100.000	96.857
DOS	85.340	99.976	94.312
U2R	64.000	72.000	84.000
R2L	95.560	96.625	97.691

Table 4.13 Performance comparisons of classification and regression trees on different reduced datasets.

From table 4.12 we can conclude that for Normal, Probe and DOS classes 12 and 17 variable reduced datasets are much better than the 19 variable reduced data set. For U2R class 19 variable reduced data set performs better compared to the other two. For R2L class there is slight increase in the performance by using 19 variable reduced data set compared to 17 and 12 variable reduced data set.

	12 variable data set Accuracy(%)	17 variable data set Accuracy(%)	19 variable data set Accuracy (%)
Normal	98.786	99.643	99.571
Probe	99.571	98.571	96.714
DOS	98.950	98.168	99.020
U2R	48.000	60.000	56.000
R2L	98.934	98.934	97.869

Table 4.14 Performance comparisons of general Bayesian network classifier on different reduced datasets

From table 4.13 we can conclude that Normal, Probe, DOS and R2L classes are classified better by using 12 and 17 variable reduced datasets compared to 19 variable reduced dataset. Only for U2R class, which has only 25 data points, does the 19 variable reduced data classify better.

4.2.9 Ensemble Approach on the 41 variable original data set

In this approach we first construct the general Bayesian network classifier and classification and regression trees classifier individually to obtain a very good generalization performance (Optimizing the model for performance on unseen data rather than the training data). Test data is passed through each individual model and the corresponding outputs are used to decide the final output as described in section 3.7. The performance of the ensemble approach is compared with the two individual models, which were used to build the ensemble approach model and presented in table 4.14. From the results we can conclude that ensemble approach gives better performance than the two individual separately used models.

	Bayesian Belief Network Accuracy(%)	Classification and Regression Trees Accuracy (%)	Ensemble Approach Accuracy (%)
Normal	99.571	99.643	99.714
Probe	99.430	99.857	99.857
DOS	96.691	99.476	99.930
U2R	64.000	48.000	72.000
R2L	99.112	90.586	99.470

Table 4.15 Performance of Ensemble Approach on the 41 variable reduced data set

The graph in figure 4.5 shows the R2L class data points for the three different models. Some 25 data points out of 563 data points are chosen to construct the graph. The classification value of 1 in the graph represents a correct classification and value of 2 represents a misclassification. The ensemble approach classifies most of them correctly by picking up all the classes which are correctly classified by the two classifiers. From the graph we can conclude that different classifiers misclassify different data points, so the ensemble approach basically exploits these differences in misclassification and improves the performance.

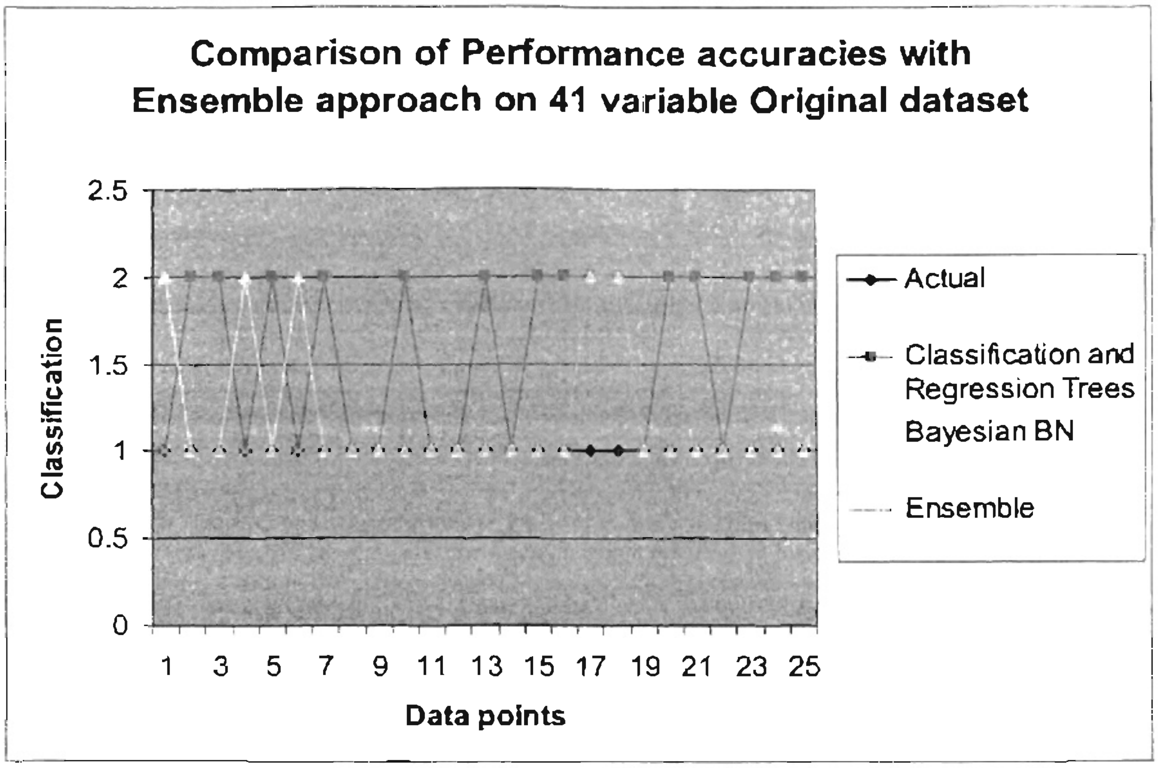


Figure 4.5 Performance of Ensemble Approach on the Original data set for R2L class.

4.2.10 Ensemble Approach on the 12 variable reduced data set

We used the ensemble approach on the 12 variable reduced data set that is obtained from the data reduction in classification regression trees algorithm. We first constructed general Bayesian network classifier and classification and regression trees classifier individually and then the test data is passed through each individual model and then the outputs of each classifier are combined to get the final output as explained in section 3.7. Table 4.15 shows the comparison of ensemble approach with the two classifiers, which were used to build ensemble approach. From table 4.15 we can conclude that the ensemble approach gives better performance compared to other models.

	Bayesian Belief Network Accuracy(%)	Classification and Regression Trees Accuracy (%)	Ensemble Approach Accuracy (%)
Normal	98.786	100.000	100.000
Probe	99.571	97.714	99.860
DOS	98.950	85.340	99.980
U2R	48.000	64.000	80.000
R2L	98.934	95.560	99.470

Table 4.16 Performance of Ensemble Approach on the 12 variable reduced data set

The graph in figure 4.6 shows the U2R class data points for three different models. All the 25 data points of U2R class are chosen for the graph. The classification value of 1 represents a correct classification and value of 2 represents a misclassification. The ensemble approach classifies most of them correctly by picking up all the classes which are correctly classified by two classifiers. We can observe from the graph that different classifiers misclassify different data points, so the ensemble approach basically improves the performance.

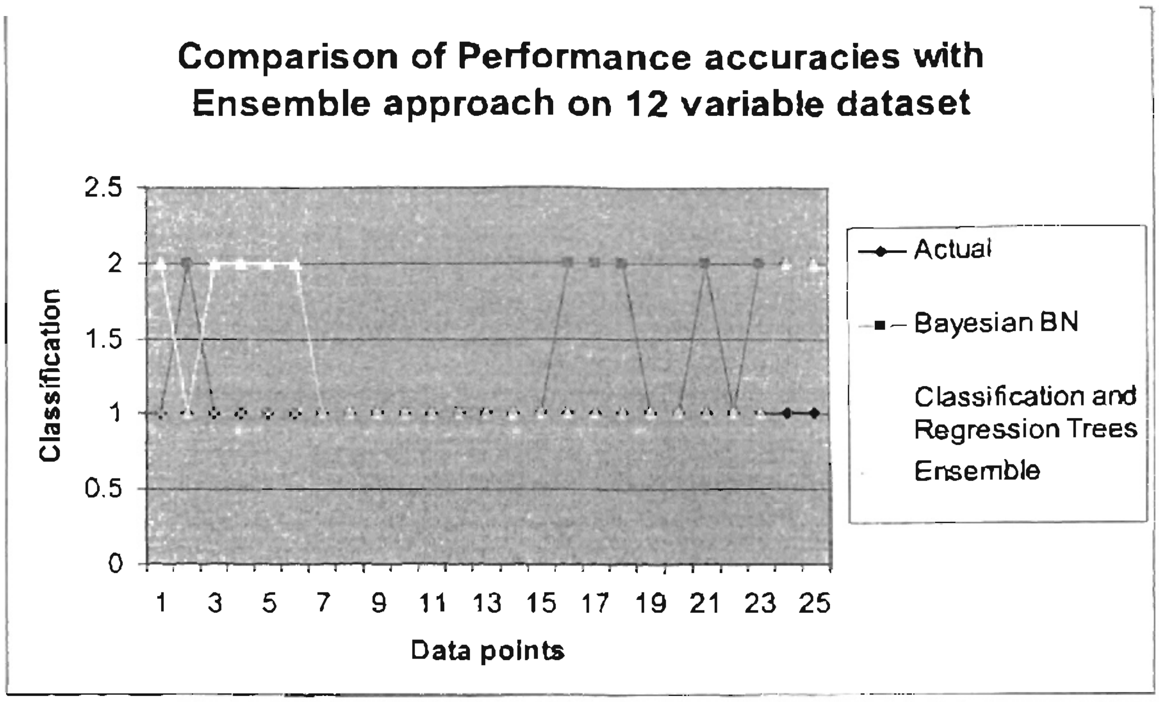


Figure 4.6 Performance of Ensemble approach on 12 variable reduced data set for U2R class.

4.2.11 Ensemble Approach on the 17 variable reduced data set

We used the ensemble approach on the 17 variable reduced data set that is obtained from feature selection procedure in the Bayesian belief network. We first constructed the general Bayesian network classifier and classification and regression trees classifier individually and then the test data is passed through each individual model and then the outputs of each classifier are combined to get the final output as explained in section 3.7. Table 4.16 compares two classifiers with the ensemble approach. From table 4.16 we can conclude that ensemble approach gives better performance compared to other models.

	Bayesian Belief Network Accuracy(%)	Classification and Regression Trees Accuracy (%)	Ensemble Approach Accuracy (%)
Normal	99.643	99.643	99.643
Probe	98.571	100.000	100.000
DOS	98.168	99.976	100.000
U2R	60.000	72.000	72.000
R2L	98.934	96.625	99.290

Table 4.17 Performance of Ensemble Approach on the 17 variable reduced data set

The graph in figure 4.7 shows the DOS class data points for three different models. Some 30 data points out of 4202 data points are chosen to construct the graph. The classification value of 1 represents the correct classification and value of 2 represents a misclassification. The ensemble approach classifies most of them correctly. From the graph we can observe that all the 30 data points are classified correctly by ensemble approach thereby improving the performance.

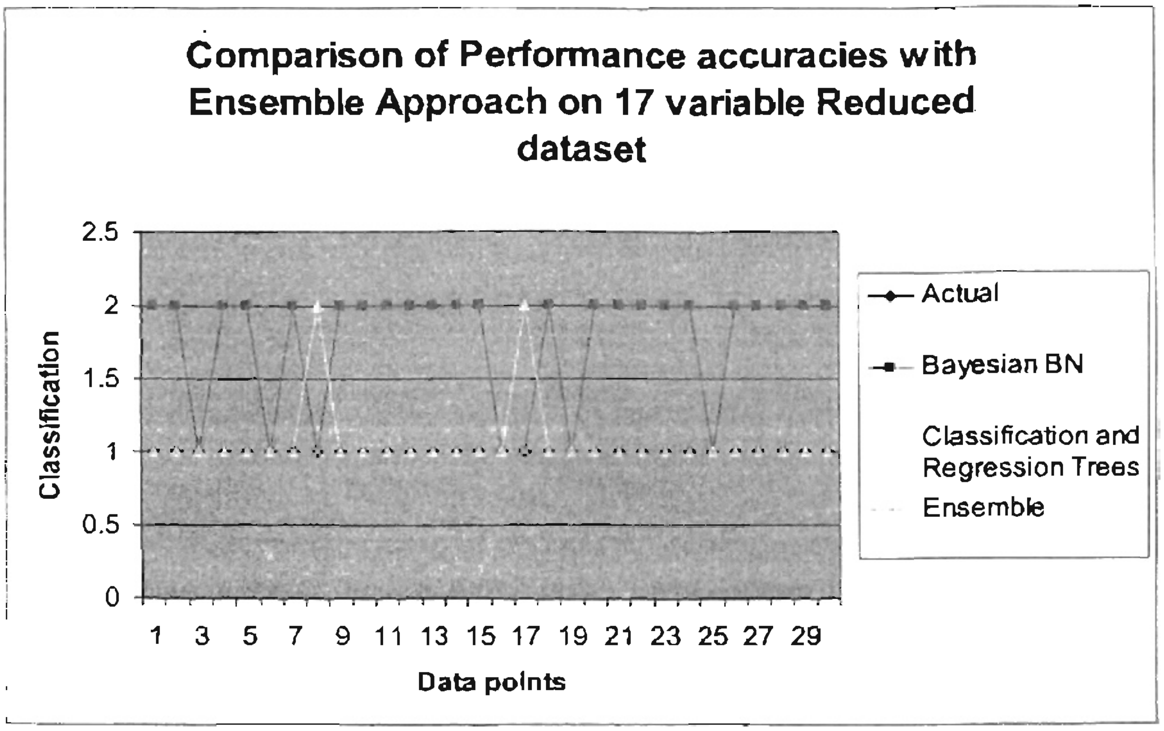


Figure 4.7 Performance of Ensemble approach on 17 variable reduced data set for DOS class.

4.2.12 Ensemble approach by using the ICA reduced data set to construct Bayesian Belief Network base classifier

We used the ensemble approach on the 2 variable reduced data set obtained by Independent component analysis algorithm, the 17 variable reduced data set obtained by Bayesian belief network algorithm and on the 12 variable reduced data set obtained by classification and regression trees algorithm. We first constructed the general Bayesian network classifier on the ICA 2 variable reduced data set, the general Bayesian network classifier on the 17 variable reduced data set and the classification and regression trees classifier individually and then the test data is passed through each individual model and then the outputs of each classifier are combined to get the final output as explained in section 3.7. Table 4.17 compares three classifiers with the ensemble approach. From table 4.17 we can conclude that ensemble approach gives better performance compared to other models.

	ICA reduced data set accuracy (%)	Bayesian BN Accuracy (%)	Classification and Regression trees Accuracy (%)	Ensemble Approach Accuracy (%)
Normal	36.733	99.643	100.000	100.000
Probe	23.812	98.571	97.714	100.000
DOS	56.323	98.168	85.340	98.253
U2R	74.776	60.000	64.000	84.000
R2L	67.355	98.934	95.560	98.970

Table 4.18 Performance of Ensemble Approach

Chapter 5

CONCLUSIONS

5.1 Conclusions

In this research we have investigated new techniques for intrusion detection and performed data reduction and evaluated their performance on the benchmark KDD Cup 99 Intrusion data. We first performed data reduction on the intrusion detection data set by using independent component analysis algorithm and principal component analysis algorithm. We next used the feature selection method in Bayesian belief network and then applied the data reduction algorithm present in classification and regression trees. Following this, we explored general Bayesian network classifier and classification and regression trees classifier as intrusion detection models. We compared the training times of the general Bayesian network classifier on the original data set and on the reduced data set. We also compared the performance accuracies of Bayesian belief network on the original data set and on the 17 variable reduced data set. In a similar fashion, we compared the training times of classification and regression trees classifier on the original data set and on the reduced data set and we compared the performance accuracies of classification and regression trees classifier on the original data set and on the 12 variable reduced data set. We then compared the performance accuracies of classification and regression trees and Bayesian BN classifier on the 41 variable data set, the 12 variable data set and on the 17 variable data set. We have done performance comparisons of different reduced data sets. Finally we designed the ensemble approach with Bayesian BN and classification and regression trees as base classifiers. We built 3 ensemble models with the original data set, 12 variable reduced data set and 17 variable reduced data set and compared their performance with base classifiers.

Data reduction using independent component analysis algorithm resulted in 2 different data sets. The 2 variable reduced data set was obtained by scaling the input data and the 39 variable reduced data set was obtained by normalizing the input data. The performance of general Bayesian network classifier with these reduced data sets was very low. The principal component analysis algorithm also gave a 17 variable reduced data set out of 41 variables. 17 variables contributed more than one percent to the variation in the data set. The performance of general BN classifier with this reduced data set was poor. We therefore used feature selection in Bayesian BN and data reduction algorithm present in classification and regression trees to reduce the data set.

The training times, testing times and performance of general BN classifier on the original data set is shown in Table 4.4. Feature selection in Bayesian BN resulted in 17 variable data set including A, B, C, E, G, H, K, L, N, Q, V, W, X, Y, Z, AD, AF variables. These 17 variables were different from the 17 identified by the PCA algorithm. The training times, testing times and performance of general BN classifier on this reduced data set is shown in Table 4.5. Comparison of training, testing times and accuracies of original data set and reduced data set are shown in Table 4.6. From the table 4.6 we can conclude that 55% savings in training time due to variable reduction for Intrusion detection systems maximizes the time performance and fast re-training of an IDS. Furthermore, accuracy of Normal class of data increased by 0.99% by using the reduced data set. The graph in figure 4.1 shows that use of a reduced data set decreases the computational time of real-time IDS. The graph in Figure 4.2 shows that for Normal class classification is better by using the reduced data set.

Table 4.6 shows the training time, testing time taken for each classifier in seconds and accuracy for each classifier in percentage terms. The data reduction algorithm present in classification and regression trees gave 12 variable reduced data set as output and it consists of C, E, F, L, W, X, Y, AB, AE, AF, AG and AI variables. Performance comparison of classification and regression trees on original data set and on the 12 variable reduced data set is shown in Table 4.8. From the table we can conclude that classification and regression trees gives better performance on the smaller training data as U2R class of data is classified more accurately by using the 12 variable reduced data set compared to the 41 variable original data set. From Table 4.8 we can conclude that 70% savings in training time due to variable reduction and 16% improvement in accuracy of U2R class. The graph in Figure 4.3 shows that the use of reduced data set reduced the cost of detection. The graph in Figure 4.4 shows that the use of reduced data set for classification reduces the overhead of the intrusion detection as a whole.

Comparison of performance accuracies of Bayesian BN and classification and regression trees on the original data set is shown in Table 4.9. For Normal and Probe class classification and regression trees classifier gives slightly better performance compared to the general Bayesian BN classifier. And for the DOS, U2R and R2L classes general BN classifier gives better performance. Table 4.10 shows the performance comparisons of Bayesian BN classifier and classification and regression trees classifier on the 12 variable reduced data set. From table 4.10 we can say that for Normal and U2R classes classification and regression trees performs better. For the other three classes general Bayesian BN classifier performs better. For the U2R class which has less data points general BN classifer performed better with the original data set. So we can conclude that general BN classifier performs better with more training data. Whereas the classification and regression trees classifier performed better for the U2R class with the reduced data set compared to the original data set. We can therefore say that classification and regression trees perform better with smaller training data. Table 4.11 shows the performance comparisons of general BN classifier and classification and regression trees on the 17 variable reduced data set. From table 4.11 we can say that for Normal class both classifiers classify to the same level. For Probe, DOS and U2R classes classification and regression trees classifier performs better. And for the R2L class general BN classifier performs better.

Table 4.12 shows the performance comparisons of classification and regression trees on different reduced datasets. For the Normal and Probe classes both 12 and 17 variable datasets performed better compared to the Srinivasan's etal [MTTJ02] reduced data set. For DOS class 17 variable reduced data set performed better than the Srinivasan's data set whereas the 12 variable data set performance is low compared to both. For the U2R class and R2L class srinivasan's dataset performed slightly better compared to the other two. Table 4.13 shows the performance comparisons of general Bayesian network classifier on different reduced datasets. For the normal class 17 variable data set performed better than the other two. For Probe class the 12 variable data set performed better than the other two.

For DOS class srinivasan's dataset performed better than the other two. For the U2R class the 17 variable data set performed better. And for the R2L class both 12 and 17 variable datasets performed better than Srinivasan's dataset.

The ensemble approach with the two base classifiers Bayesian BN and classification and regression trees was constructed and evaluated its performance with the two base classifiers on the original data set, 12 variable and 17 variable reduced datasets. The results in table 4.14, 4.15 and 4.16 suggest that ensemble approach is a good model for intrusion detection. The ensemble approach combines the complementary features of the base classifiers. The ensemble approach gave 100% accuracy for Normal class on the 12 variable reduced data set and 100% accuracy for Probe and DOS classes on the 17 variable reduced data set. The results in table 4.17 show the performance of the ensemble approach with the three base classifiers. The first base classifier was the Bayesian BN classifier constructed on the 2 variable ICA reduced data set, the second one was the Bayesian BN classifier constructed on the 17 variable reduced data set and the third one was the classification and regression trees classifier constructed on the 12 variable reduced data set. The results show that ensemble approach performs much better than the three base classifiers. This suggests that if proper base classifiers are chosen 100% accuracy might be possible for other classes too on different datasets.

BIBLIOGRAPHY

[BCHSTT01] E. Bloedorn, A.D. Christiansen, W. Hill, C. Skorupka, L.M. Talbot, J.Tivel. Data Mining for Network Intrusion Detection: How to Get Started. The MITRE Corporation, McLean, VA, Pages 1-9, 2001.

[BFOS84]L. Breiman, J. Friedman, R. Olshen, C.Stone. Classification and Regression Trees. WADSWORTH INTERNATIONAL GROUP, Belmont, California, Pages 93-126, 1984.

[Big96] J. Bigus. Data Mining with Neural Network. McGrawHill, 1996.

[Can98] J.Cannady. Artificial Neural Networks for Misuse Detection. School of Computer and Information Sciences, Nova Southeastern University, Fort Lauderdale, Pages 4-9, 1998.

[CBL98] J. Cheng, D. Bell, W. Liu. Learning Bayesian Networks from Data: An Efficient Approach Based on Information Theory. Technical Report, Pages 24-28, 1998.

[CG99] J. Cheng, R. Greiner, Comparing Bayesian Network Classifiers. (Proceedings of the fifteenth conference on uncertainty in artificial intelligence, 1999) Department of Computing Science, University of Alberta, Edmonton, Alberta T6G 2H1 Canada, Pages 2-6, UAI'99.

[CG01] J. Cheng, R. Greiner, Learning Bayesian Belief Network Classifiers: Algorithms and System. (Proceedings of the fourteenth Canadian conference on artificial intelligence,

2001) Department of Computing Science, University of Alberta, Edmonton, Alberta T6G 2H1 Canada, Pages 3-8, AI'2001.

[DBS92] H. Debar, M. Becker, D. Siboni. A Neural Network Component for an Intrusion Detection System. Proc. IEEE Symp. on Research in Computer Security and Privacy, pp.240-250, 1992.

[Den97] D.E. Denning. An Intrusion Detection Model. In IEEE Transactions on Software Engineering, February 1987.

[Die00] T. G. Dietterich. Ensemble Methods in Machine Learning. Proceedings of the 1st International Workshop on Multiple Classifier Systems, Cagliari, Italy, 2000.

[FRA94] Jeremy Frank. Artificial Intelligence and Intrusion Detection: Current and Future Directions. Division of Computer Science, University of California at Davis, CA, June 1994.

[GWC98] A.K Ghosh, J. Wanken, F. Charron. Detecting Anomalous and Unknown Intrusions Against Programs. Reliable Software Technologies, Sterling, VA, 1998.

[HO99] A. Hyvarinen, E. Oja. Independent Component Analysis: A Tutorial. Helinski University of Technology, Finland, Pages 8-18, April 1999.

[HYV99] A. Hyvarinen. Survey on Independent Component Analysis, Helinski University of Technology, Pages 7-15, 1999.

[Ilg92] K. Ilgun. USTAT: A Real-Time Intrusion Detection System for UNIX. Master Thesis, University of California, Santa Barbara, November 1992.

[KDD99] KDD cup 99 Intrusion detection data set. <http://kdd.ics.uci.edu/databases/kddcup99/kddcup.data_10_percent.gz>

[KHDM98] J. Kittler, M. Hatef, R. P. W. Duin, J. Matas. On Combining Classifiers. IEEE Trans. on Pattern Analysis and Machine Intelligence, 20(3), pp. 226-229, 1998.

[Lee99] Wenke Lee, A Data Mining Framework for Constructing Features and Models for Intrusion Detection Systems. PhD Thesis, Columbia University, Pages 227-259, 1999.

[LSM99] W. Lee, S. Stolfo, K. Mok. A Data Mining Framework for building Intrusion Detection Models. In Proceedings of the 1999 IEEE Symposium on Security and Privacy, Oakland, CA, May 1999.

[LSM00] W. Lee, S. Stolfo, and K. Mok. Adaptive Intrusion Detection: a Data Mining Approach. Journal Artificial Intelligence Review, Kluwer Academic Publishers, Pages 20-30, 2000.

[Luo99] J. Luo. Integrating Fuzzy logic with Data Mining Methods for Intrusion Detection. Master's Thesis, Department of Computer Science, Mississippi State University, Mississippi, Pages 9-13, 1999.

[MHL94] B. Mukherjee, L.T. Heberlein, K.N. Levitt. Network Intrusion Detection. IEEE 1994.

[MIT] MIT Lincoln Laboratory. <http://www.ll.mit.edu/IST/ideval/>

[MJS02] S. Mukkamala, G. Janoski, A. Sung. Intrusion Detection: Support Vector Machines and Neural Networks. Department of Computer Science, New Mexico Institute of Mining and Technology, Socorro, New Mexico, 2002.

[MJS02] S.Mukkamala, G. I. Janoski, A.H. Sung. Intrusion Detection Using Support Vector Machines. Department of Computer Science, New Mexico Institute of Mining and Technology, Socorro New Mexico, 87801 USA, 2002.

[MSA02] S. Mukkamala [1], A. H. Sung [1, 2], A. Abraham [3]. Identifying Key Variables for Intrusion Detection Using Soft Computing Paradigms, [1]Department of Computer Science, [2]Institute for Complex Additive Systems Analysis, New Mexico Tech, Socorro, New Mexico 87801, [3]Department of Computer Science, Oklahoma State University, 700 N Greenwood Avenue, Tulsa, OK 74106, 2002.

[MS03] S.Mukkamala, A. H. Sung. Identifying Significant Features for Network Forensic Analysis Using Artificial Intelligent Techniques. Dept of Computer Science, New Mexico Tech, Winter2003.

[MTTJ02] S. Mukkamala, G.R. Tadiparthi, N. Tummula, G. Janoski. Audit Data Reduction for Intrusion Detection. Dept of Computer Science, New Mexico Institute of Mining and Technology, Scorro, New Mexico, 2002.

[RLM98] J.Ryan, M. Lin, R. MIikkulainen. Intrusion Detection with Neural Networks. The University of Texas at Austin, Austin, Texas, 1998.

[Sie99] R. S. Sielken. Application Intrusion Detection. Master's Thesis, School of Engineering and Applied Science, University of Virginia, Pages 20-30, May 1999.

[SM03] A. H. Sung, S. Mukkamala: Identifying Important Features for Intrusion Detection Using Support Vector Machines and Neural Networks. Pages 209-217, SAINT 2003.

[Sun96] A. Sundaram. An Introduction to Intrusion Detection. ACM Cross Roads, Vol. 2, No. 4, April 1996.

[Unad00] S. Unadkat. ICA Applications in Data Mining. Masters Thesis, Pages 15-30, 2000.

[Zam01] D. Zamboni. Using Internal Sensors for Computer Intrusion Detection. PhD Thesis, Purdue University, Pages 2-22, August 2001.

[ZL00]Y. Zhang, W. Lee. Intrusion Detection in Wireless Ad-Hoc Networks. In Proceedings of the Sixth International Conference on Mobile Computing and Networking (MobiCom 2000), Boston, MA, August 2000.

VITA 9

Srilatha Chebrolu

Candidate for the Degree of

Master of Science

Thesis: DATA REDUCTION AND DATA CLASSIFICATION IN AN INTRUSION DETECTION SYSTEM

Major Field: Computer Science

Biographical:

Personal Data: Born in Kandukur, Andhra Pradesh, India, On July 1, 1979, the daughter of Venkateswarlu Chebrolu and Chinnammayi Chebrolu.

Education: Graduated from Simhapuri Public School, Nellore, India in May 1995;Passed out from Adarsha Junior College , Ongole, India in May 1997;received Bachelor of Technology from Shadan College of Engineering and Technology , Hyderabad, India in June 2001. Completed the requirements for the Master of Science degree with a major in Computer Science at Oklahoma State University in December, 2003.

Experience: Employed by Oklahoma State University, Department of Computer Science as a graduate teaching assistant for Advanced Operating systems; served as a Web Developer for the Oklahoma State University, Department of Mathematics, 2001 to present.