Knowledge Based Anomaly Detection

Akara Prayote

Supervisor: Professor Paul Compton

A thesis submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

The School of Computer Science and Engineering
The University of New South Wales

November 2007

Abstract

Traffic anomaly detection is a standard task for network administrators, who with experience can generally differentiate anomalous traffic from normal traffic. Many approaches have been proposed to automate this task. Most of them attempt to develop a sufficiently sophisticated model to represent the full range of normal traffic behaviour. There are significant disadvantages to this approach. Firstly, a large amount of training data for all acceptable traffic patterns is required to train the model. For example, it can be perfectly obvious to an administrator how traffic changes on public holidays, but very difficult, if not impossible, for a general model to learn to cover such irregular or ad-hoc situations.

In contrast, in the proposed method, a number of models are gradually created to cover a variety of seen patterns, while in use. Each model covers a specific region in the problem space. Any novel or ad-hoc patterns can be covered easily. The underlying technique is a knowledge acquisition approach named Ripple Down Rules. In essence we use Ripple Down Rules to partition a domain, and add new partitions as new situations are identified. Within each supposedly homogeneous partition we use fairly simple statistical techniques to identify anomalous data. The special feature of these statistics is that they are reasonably robust with small amounts of data. This critical situation occurs whenever a new partition is added.

We have developed a two knowledge base approach. One knowledge base partitions the domain. Within each domain statistics are accumulated on a number of different parameters. The resultant data are passed to a knowledge base which decides whether enough parameters are anomalous to raise an alarm. We evaluated the approach on real network data. The results compare favourably with other

ORIGINALITY STATEMENT

‘I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, or substantial proportions of material which have been accepted for the award of any other degree or diploma at UNSW or any other educational institution, except where due acknowledgement is made in the thesis. Any contribution made to the research by others, with whom I have worked at UNSW or elsewhere, is explicitly acknowledged in the thesis. I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the project's design and conception or in style, presentation and linguistic expression is acknowledged.’

Signed …….....

Date …….....

techniques, but with the advantage that the RDR approach allows new patterns of use to be rapidly added to the model.

We also used the approach to extend previous work on prudent expert systems - expert systems that warn when a case is outside its range of experience. Of particular significance we were able to reduce the false positive to about 5%.

Acknowledgments

I am extremely grateful to my advisor, Professor Paul Compton, for his constant support, encouragement, for proofreading numerous drafts of this thesis, and helping me throughout my study. Without him, this thesis would never have happened. Not only does he guide my research, he also supports me when I am down and lost. What he has done for me is beyond what words can say.

I am also grateful to Royal Thai Government for funding a scholarship and the University of New South Wales for a UIPA scholarship. I would also like to thank the Faculty in the department of Computer and Information Science, King Mongkut's Institute of Technology North Bangkok, for letting me take leave for the study.

I would like to thank Peter Linich, the network manager in School of Computer Science and Engineering, UNSW, for his support in providing audit data sets and educating me about the network intrusion detection task.

I am thankful to Woralak Kongdenfha, Thanong Poonteerakul, Apisit Numprasanthai, Pat Leelarasamee, Pattrawut Tippimon, Chatchai Boonjaroen, Zhalaing Cheung and all my friends in the UNSW Badminton club for being good friends, keeping me company and never letting me feel lonesome.

I am especially impressed by my beloved Ms. Tippyarat Tansupasiri for her ultimate understanding, patience and support. Her contribution of ideas to my study is no less than to my life. Not only has Tippayarat patiently and strongly been waiting for me to return for five years, she has always made me smile, no matter what situation I have encountered. She is a real angel in my life.

Finally, my greatest debt is owned to my parents, my brother, my grandmother, and every one in my family for their love, patience and support, always.

Contents

1 Introduction **1**

1.1 Overview of Thesis 3

2 Network Attack and Intrusion Detection Systems **7**

2.1 Network Attacks 7

2.1.1 Vulnerabilities 9

2.1.2 Intentions of Attacks 11

2.1.3 Mechanisms of Attacks 12

2.2 Defense Against Attacks 16

2.2.1 Configuration Management 16

2.2.2 Firewalls 17

2.2.3 Encryption 17

2.2.4 Intrusion Detection System 18

2.2.5 Other Research 22

2.3 Generic Intrusion Detection System 22

2.3.1 Audit Collection and Storage 23

2.3.2 Reference Data Generation 24

2.3.3 Intrusion Detection 25

2.3.4 Policy Execution 28

2.4 Intrusion Detection Systems: Examples 28
2.4.1 Haystack 28
2.4.2 MIDAS 29
2.4.3 IDES 30
2.4.4 NIDES 31
2.4.5 Wisdom and Sense 32
2.4.6 EMERALD 33
2.4.7 NSM 35
2.4.8 Hyperview 35
2.4.9 USTAT 36
2.4.10 IDIOT 37
2.4.11 Snort 37
2.5 Traffic Volume Anomaly Detection 38
2.5.1 Traffic Volume: An Alternative Audit Data 39
2.5.2 Intrusion Detection on Traffic Volume 41
2.5.3 Discussion on Traffic Volume Anomaly Detection 45
2.6 Limitations and Questions of Intrusion Detection Systems 46

3 Knowledge Based Systems 49

3.1 Knowledge Base 50
3.2 Knowledge Elicitation 51
3.2.1 Knowledge Elicitation Methodologies 53
3.2.2 Difficulties in Knowledge Elicitation 55
3.3 Ripple Down Rules 58
3.3.1 Ripple Down Rule and Cognitive Psychology 58

3.3.2 Ripple Down Rule Essentials 59
3.3.3 RDR Terminology 62
3.3.4 RDR Structure 64
3.3.5 Construction RDR 71
3.3.6 RDR Applications 73
3.3.7 RDR Problems and Discussion 76

4 Detecting Outliers from Homogeneous Data 79

4.1 Outliers in Statistics 80
4.1.1 Accommodation of Outliers 82
4.1.2 Discordancy Test 82
4.1.3 Discussion 82
4.2 Outlier Detection 83
4.2.1 Distribution-Based Approach 84
4.2.2 Depth-Based Approach 84
4.2.3 Distance-Based Approach 85
4.2.4 Density-Based Approach 85
4.2.5 Problems and Discussion 86
4.3 Detecting Outliers while Learning 87
4.4 Simulations 91
4.4.1 Simulations on Noise Free Data Sets 92
4.4.2 Simulations on Outlier Contaminated Data Set 93
4.5 The Effect of A Normal Distribution 103
4.6 Conclusion 106

5 Network Traffic Anomaly Detection 108

5.1 Manual Network Anomaly Detection 111
5.1.1 Audit Traffic Data 111
5.1.2 Actions on Anomalies 112
5.2 Using RDR to Automate Traffic Anomaly Detection 113
5.2.1 Network Traffic Behaviour Profiling 113
5.2.2 Traffic Anomalies Detection 115
5.2.3 Discussion on RDR-based Anomaly Detection 116
5.3 System Implementation 117
5.3.1 System Architecture 118
5.3.2 Periodic Monitoring Cases 120
5.3.3 Knowledge Bases 127
5.3.4 Coding and GUI 128
5.4 Traffic Anomaly Detection Experiment 130
5.4.1 Traffic Data Sets 131
5.4.2 Experiments 137
5.4.3 Results And Discussion 138
5.4.4 Building Up Knowledge Bases 139
5.4.5 A Comparison with Another Method 145
5.5 Conclusion 146

6 Prudent Knowledge Bases **147**

6.1 Introduction 147
6.2 A Review on RDR with Prudence 149
6.2.1 Audit Data Sets 149
6.2.2 Simulation 151

6.2.3 Previous Results 152
6.3 Probabilistic Profile for Continuous Attribute 154
6.3.1 Data Set 155
6.3.2 Simulation of OEBA-based Prudence 155
6.3.3 Results and Discussion 156
6.4 Probabilistic Profile for Categorical Attribute 159
6.4.1 Outlier Estimation for Categorical Attributes 160
6.4.2 Simulation on OECA-based Prudence 161
6.5 Discussions of Probabilistic Profiles 164
6.6 An Extension on the Correlation between Attributes 167
6.6.1 Correlations between Attributes 167
6.6.2 Correlation and the Algorithm OEBA 169
6.6.3 Simulation of Correlation Profile on Uncorrelated Attributes 171
6.6.4 Simulation of OEBA and Correlations 172
6.7 A Study on Warning Characteristics 173

7 Summary **174**

7.1 Thesis Summary 175
7.2 Discussion 178
7.3 Future Research 180
7.3.1 An Interim Outlier Detection Algorithm 180
7.3.2 Combining Multiple Tests 180
7.3.3 Redundancy of Partitions 181
7.3.4 Correlations Between Classifications 181

List of Figures

2.1 Generic tasks of intrusion detection systems. 23

2.2 An example of graphical visualization of traffic volume. 40

3.1 The architecture of an expert system. 50

3.2 An example of SCRDR knowledge base in binary tree structure. . . 66

3.3 An example of SCRDR in composite rules structure. 67

3.4 An example of MCRDR knowledge base. 70

4.1 Distance between bands of outliers and valid data. 94

4.2 Java code of data set generator with outlier contamination. 95

4.3 False negative rates of different contamination levels at the distance of 0.01. 98

4.4 False negative rates of different contamination levels at the distance of 0.05. 99

4.5 False negative rates of different contamination levels at the distance of 0.10. 100

4.6 False negative rates of different contamination levels at the distance of 0.50. 101

4.7 False negative rates of different contamination levels at the distance of 1.00. 102

4.8 False positive rates for normally distributed data. 105

5.1 A system architecture of RDR-based network traffic anomaly detection. 119

5.2 An example of network traffic and its medians. 125

5.3 An example of two series with same median but different fluctuation. 126

5.4 GUI of the main window 129

5.5 GUI of the profile construction Window. 129

5.6 GUI of the rule construction window 130

5.7 Traffic series T_1 132

5.8 Traffic series T_2 133

5.9 Traffic series T_3 134

5.10 Traffic series T_4 135

5.11 Traffic series T_5 136

5.12 System states. 137

5.13 The number of profiles, decision rules in the two KBS, and the number of false positive cases added to profiles against seen cases. . . . 143

6.1 False positives and false negatives produced from different thresholds. 163

6.2 Examples of different correlations between 2 variables. 168

6.3 Limits in different directions. 170

List of Tables

4.1 Simulation results of the OEBA algorithm on noise free data. 93

4.2 False positive rates for uniformly distributed data 97

4.3 Minimum threshold with 0 false negatives 103

4.4 False positive rates (FPR) of various threshold T of the OE and BA algorithms on data set following normal distribution. 104

5.1 University seasons of each data set 131

5.2 Anomalies in data sets 131

5.3 Data sets for training and testing in each system state 137

5.4 False positive and false negative rates of each test data sets in different system states 139

5.5 KA sessions for the first 1,000 cases. 141

5.6 KA sessions from case 2000 to case 3000. 142

5.7 The number of profiles, decision rules and profile updates for every 1,000 cases. 144

6.1 Final results of simulation of RDR with prudence 153

6.2 A comparison between original prudence and OEBA-based prudence on continuous attributes 157

6.3 Sources of incorrect warnings 158

6.4 Simulations on categorical attributes 162

6.5 Simulations when new rules are not added for false negatives 163

6.6 Three false negative cases. 165

6.7 Simulations on OEBA profiles for correlation 171

6.8 Simulations of the algorithm OEBA for correlation between attributes.172

Chapter 1

Introduction

Communications is now central in human lives, both for leisure and work. A variety of products/applications have been continuously introduced to facilitate communication. Underlying much of the success in communications technology are computer networks. From one device to another, from one area to another area, from one country to another country, from one continent to another continent, finally, the world is connected. Great effort has gone into the study of networks to improve their performance as much as we can. Recently, one of many concerns in network performance is *security*.

Many attempts have been put together to secure networks; i.e., protecting them from a variety of intrusion. There are different kinds of intrusion. Each of which attacks with different purposes. For example, one hacker attempts to gain an access to an organization's network for some information, while another might want to disrupt a network. They require different techniques to detect, to identify and to remedy.

A network intrusion detection system (NIDS) is one way to approach network security. A variety of architectures and techniques have been proposed to implement such a system. Basically, a NIDS monitors data within the network and takes further actions if an anomaly is detected. Typical audit data used by NIDSs are TCP connection packet data. Recently, there has been a community suggesting monitoring traffic volume instead of packet data. It has been demonstrated that

most attacks alter traffic volume and can be detected effectively.

Actually, detecting anomalies from traffic volume is not a new idea. For a number of years, it has been a standard task for network administrators, who with experience can generally differentiate anomalous traffic from normal traffic. The task can be viewed as the problem of detecting anomalies in time series data.

Time series have been widely studied in statistics. Essentially, it is data in a sequential form with significance in the order of occurrence. Early foci of the study were on behavioural characteristics and modeling of time series, which would later support a prediction process. Anomaly detection was started as a by-product of other time series analysis. But it has now become a major focus of such study. Early techniques were statistics based, that is, a distribution or model for the time series was estimated and any values deviating from the learned model were flagged as outliers. For example, a normal distribution was mostly used to estimate data distribution. Regression techniques, such as autoregressive integrated moving average (ARIMA) (Mills, 1990); and forecasting algorithms, such as Holt-Winters (Brockwell and Davis, 1996) were used to estimate time series. Later studies investigated clustering techniques as alternatives, while density-based approaches have recently been attracting more interest in the area.

The common thing among those techniques is that they essentially rely on a universal model of time series. The model is meant to capture the generic characteristics of a time series. Any data behaving differently from the model are marked as outliers. Under this paradigm, when novel characteristics or ad hoc events occur, they are simply flagged as outliers. Furthermore, statistical or learning methods do not easily accommodate individual special cases.

In contrast, this thesis investigates a mechanism to partition a problem space into smaller subspaces of homogeneous traffic and learn a model for each subspace. The partitioning can be conducted at any time when the system is in routine use via the knowledge acquisition methodology named Ripple Down Rules; novel events can be added to a knowledge base when required.

The study also investigates a technique called *Prudence*. A prudent expert sys-

tem is one which knows when it has reached the limits of its knowledge. Evaluations have been conducted of two novel algorithms Outlier Estimation with Backward Adaptation (OEBA) and Outlier Detection for Categorical Attributes(OECA), which have been integrated into the RDR framework.

1.1 Overview of Thesis

Chapter 2

An objective of this chapter is to review intrusion detection systems (IDS). The chapter firstly introduces commonly known network attacks, which exploit network vulnerabilities to accomplish their mission. The intention of attacks differ from one to another, resulting in different mechanisms for the attack. This section presents various intentions and mechanisms for network attacks. Defensive techniques are reviewed in a subsequent section, including intrusion detection systems (IDS). The history of IDS is briefly presented. A generic architecture for IDS is summarized for a better understanding of IDS's elements. The IDS literature is reviewed and discussed next. It is found that common among techniques used so far are attempts to construct a universal model for a problem domain and using that model to detect anomalous traffic. Techniques generally use sophisticated statistics and mathematics. The most common audit data used in the analysis is TCP connection information. The use of alternative audit data based on traffic volume has emerged recently. Literature related to anomaly detection from traffic volume is reviewed and discussed, as the thesis uses traffic volume as its audit data. Finally, there is some discussion about the limitations of IDS.

Chapter 3

Expert systems (ES) have been used extensively to automate processes in many tasks. An ES basically consists of a knowledge base and an inference engine. A knowledge base is a collection of knowledge learned from a particular domain, which

will be retrieved for later reference or used by an inference engine to derive a conclusion for a situation. The chapter briefly discusses ES from both philosophical and practical views. One difficulty in constructing a knowledge base is the process of knowledge elicitation. A technique called Ripple Down Rules (RDR) has been proposed to facilitate the task. RDR has been well proved as a simple and incremental technique for constructing a knowledge base. It was originally used in classification tasks. Soon after, it was applied to many other areas, including configuration, heuristic search and image processing tasks. This chapter provides a discussion on RDR from its philosophy, through its terminology, elements and structure, to its application. The philosophy of RDR is based on situated cognition, where a knowledge is a justification in a particular context. RDR terminology, including a formal representation, is reviewed to provide a ground for later discussion. RDR was originally implemented with a binary tree. It was later enhanced to n-ary trees and composite rules (or decision list). A variety of tasks that RDR has been applied to are then reviewed. Finally, the chapter concludes with problems and further research into RDR.

Chapter 4

As mentioned earlier, RDR is used to segment a problem space into sub-spaces of homogeneous data, which should be more easily profiled by a learning algorithm. This chapter presents a novel algorithm that is able to learn homogeneous data and detect outlying data that differs from the population. A variety of outlier detection techniques are reviewed prior to an introduction to our novel algorithm, named Outlier Estimation with Backward Adaptation (OEBA). The algorithm OEBA is designed for continuous attributes because most attributes in a network domain are numerical.

False positive and false negative rates are used to evaluate OEBA. Firstly, a simulation is conducted on noise free data sets. It demonstrates that the algorithm does not produce false positive warnings with a suitable threshold. Next, a simulation is conducted on data sets with randomly injected outliers. Outliers

are characterized by two attributes; i.e., the number of outliers in a data set and the difference between outliers and valid data. Results from all these simulations demonstrate that there exists a range of thresholds that produces no false positive and no false negative instances for data sets with a range of outlier contamination.

Chapter 5

One of many tasks in network monitoring is anomaly detection, which is done both manually and automatically. Manual detection of traffic anomalies is typically performed by network administrators of an organization. The chapter explains anomaly detection in the School of Computer Science and Engineering (CSE), University of New South Wales (UNSW). Traffic data are archived by a network tool called RRDTool and illustrated on a web page on request. An administrator looks up graphs of network traffic consumption to detect anomalies for further investigation. To automate the task, RDR is used to segment network traffic into regions of homogeneous traffic and the algorithm OEBA is applied to learn the behavior of a region. A technique for profiling network traffic behavior and how to construct OEBA-profiles and organize them in a RDR tree are explained. A profile retrieval and a matching mechanism, which is the core of anomaly detection, are discussed next. After the model is explained, the chapter presents an implementation. The study implements a system prototype to detect network anomalies from traffic consumption in CSE, UNSW. The system is explained in detail: its architecture, audit data, knowledge base structure. Examples of the graphical user interface of the prototype are presented along with results of its use.

Chapter 6

A problem in any expert system is brittleness. Expert systems are brittle because they do not realise the limits of their knowledge. The CYC project (Guha and Lenat, 1990) is an attempt at a solution to this problem by building a knowledge base of common sense, or general knowledge, as a foundation on which other expert systems could be built. In RDR, an attempt called *prudence* is made to provide

experts with warnings when a system reaches the limits of its knowledge. It has been shown that the technique could successfully warn about most cases where the system had made an incorrect conclusion. However, it was found that false positive was high at 15%. It was our intention to improve prudence performance by applying our learning algorithm OEBA. An evaluation of the algorithm OEBA on the Garvan data set demonstrates an improvement in performance. However, during the investigations, it was found that the majority of false positive warnings were from categorical attributes. Another learning algorithm was developed for categorical attributes and named Outlier Estimation for Categorical Attributes (OECA). When both algorithms OEBA and OECA are applied to the prudence mechanism, the false positive rate can be successfully reduced to 5% or lower.

As both algorithms OEBA and OECA are applied on each attribute separately, correlations between attributes are not addressed. The chapter also presents an extension of the approach to address correlations between attributes. The chapter also presents an investigation of the characteristics of warnings produced.

Chapter 7

The main points from the thesis are summarised and future directions are outlined.

Chapter 2

Network Attack and Intrusion Detection Systems

The fact that network attacks can change significantly from one period of time to another makes Internet security critical. This chapter explains the nature of network attacks, their effects and defensive techniques against them.

This chapter is organized as follows. Network attacks are summarized in Section 2.1. The defensive techniques are reviewed in Section 2.2. Intrusion detection systems are one approach among others to detect network attacks before they cause any further damage to the system. The generic architecture of intrusion detection systems is introduced in Section 2.3. Afterwards, techniques used to implement such systems are discussed in Section 2.4. More recently, new approaches for network intrusion detection analyze network measurement, instead of packet header information. This is discussed in 2.5. Finally, problems and limitations in the implementation of intrusion detection systems are discussed in Section 2.6.

2.1 Network Attacks

The number of attacks on computer networks has grown every day. Network attacks occur in many forms and with different objectives. Hence, there exist many attempts to group or classify those attacks. This section gives a brief review of

these classifications of intrusions.

Smaha 1988, for example, has divided network intrusions according to their characteristics into 6 types, i.e., attempted break-in, masquerade attack, penetration of security control system, information leakage, denial of service and malicious use. Attempted break-in is an attack where someone is attempting to break into a system but is not yet successful. This can be observed by an abnormally high rate of password failures. The masquerade attack is one step further where the attempt is accomplished and the attacker can log into the system through a legitimate account. However as the behaviour can be strangely different from the legitimate user, this makes it possible to be discovered. Unlike the first two types that are caused by outside intruders, the penetration of the security control system is caused by a legitimate user attempting to unauthorizedly access resources, e.g., files or programs, not normally permitted to him. This can however be noticed by monitoring for specific patterns of activity. Similarly, the leakage is caused by a legitimate user with authorized access trying to disclose sensitive resources to unauthorized people. This can be seen from unusual access times or strange behaviour. Denial of service, probably the most well known attack, is the situation where an intruder tries to monopolize a resource, i.e. preventing legitimate users of a service from using that service. The last type, malicious use, may include miscellaneous attacks such as deleting files, resource hogging, which can be detected by atypical behavior profiles, violations of security constraints, or use of special privileges.

Another view of network attacks is presented by (Anderson, 2001), where attacks are classified according to their source location and mechanisms, i.e., attacks on local networks and attacks using Internet protocols and mechanisms.

(Lyle, 1998), instead, considered the effect of network attacks as criteria. The author divided attacks into 3 categories, i.e., attacks on integrity, attacks on confidentiality and attacks on availability.

No matter what categorization is used to group network attacks, they are just different perspectives on network attacks. In this thesis, we firstly discuss system vulnerabilities, as most network attacks involve a combination of vulnerabilities of underlying systems, and running applications. The common intentions of attacks

are summarized. The main mechanisms that attacks have used are discussed in the subsequent section.

2.1.1 Vulnerabilities

The exact vulnerabilities being exploited change from time to time; as bugs get fixed, new software introduces new bugs. However, critical vulnerabilities have been profiled and ranked by the SANS Institute and National Infrastructure Protection Center (NIPC) to provide information for organizations to prioritize their efforts and close the most dangerous holes first. As of November 2005, the SANS Institute released the top 20 vulnerabilities summarized into four categories, i.e., Windows systems, cross-platform applications, Unix systems and networking products, as follows (Institute, 2005).

Windows Systems

Vulnerabilities in Windows systems were reported to be found most in Windows services, Internet Explorer, Windows libraries, Microsoft Office and Outlook Express and from weaknesses in configuration. As most services provide remote interfaces to client components through Remote Procedure Calls (RPCs), remotely exploitable buffer overflow is the number one issue for Windows services. In Internet Explorer, the main problems reported are memory corruption, spoofing and execution of arbitrary scripts. Since Windows libraries are used for many common tasks such as HTML parsing, image format decoding, etc., a critical vulnerability in a library can impact a range of applications that rely on that library. Microsoft Office and Outlook Express are the most widely used email and productivity suites worldwide. Damages to infected machines and organizations from viruses and malicious codes have been reported constantly. Configuration weaknesses, i.e., weak passwords on accounts or network shares, are usually exploited by families of bots and worms.

Cross-Platform Applications

In cross-platform applications, top vulnerabilities were reported in backup soft-

ware, anti-virus software, PHP-based applications, database software, file-sharing applications, DNS software, media player applications, instant messaging applications, and Mozilla and Firefox browsers. There are some other applications that cannot be classified into any of above categories. The most common exploits reported for these are:

- Malicious software like spyware, Trojans or adware on users' systems which may completely compromise a user's system without requiring much user interaction
- Buffer overflow
- Worms
- Bots
- Leakage of sensitive information
- Distributed Denial of Service (DDos)
- Rogue bandwidth utilization

DNS software is found to be prone to many types of transaction attacks, including cache poisoning, domain hijacking and man-in-the-middle redirection.

Unix

Although Unix is claimed to be stable, there are still vulnerabilities that have been exploited. The top vulnerabilities are weaknesses in system configuration. For example, weak passwords for user accounts makes SSH one of the services very popularly targeted with brute-force password guessing attacks. These attacks can be extended to other interactive services like telnet, ftp, etc.

Networking Products

The top vulnerabilities in networking products have been reported for Cisco's Internetwork Operating System (IOS), as the most common operating system for enterprise routers and switches in the world, powering nearly 85% of global Internet

backbone (Institute, 2005). It was shown to have several vulnerabilities that allow denial-of-service conditions or remote execution of malicious codes. There are also Cisco products that run on application specific OS, which have been reported to have some vulnerabilities that can result in the same exploitation. Not only Cisco products have been reported with vulnerabilities, Juniper, CheckPoint and Symantec products have also been reported with vulnerabilities in operating systems, for example, JunOS, the second most common OS for backbone Internet routers. It has been reported that exploits could reboot Juniper routers and compromise the Symantec and CheckPoint Firewall/VPN devices.

Network vulnerabilities are present in every system. It is very difficult, if not impossible, to get rid of all of them, as network technology advances rapidly. The best way is to minimize them as much as possible. To do so, we need to keep vigilant on vulnerabilities reported and have the system patched immediately. Discovered vulnerabilities are usually posted publicly on web sites, e.g., www.securityfocus.com, www.cert.org, www.sans.org, cve.mitre.org, www.ciac.org/ciac/.

2.1.2 Intentions of Attacks

1. Information Disclosure

   Many attacks aim to gain access to confidential or personal information, e.g., passwords, credit card numbers. With this confidential information, an attacker can further access private resources. For example, with illegitimate access to the root password, an attacker can take control over the whole system.

2. Malicious Use

   With an aim to illegitimately use the resources of the system, an attacker seeks access to the system. An attack can be originated either from a inside who wants higher privileges than assigned or an outsider who wants to use the system's resources.

3. System Failure

   Some attacks aim to render resources unavailable to legitimate users. This

might be to destroy commercial competitiveness or might be a military or terrorist attack.

2.1.3 Mechanisms of Attacks

1. Eavesdropping Attack

   Eavesdropping can occur when an attacker gains access to the data path in a network and can monitor and read the traffic. It is considered a passive attack as it basically eavesdrops or registers a user's activities to gain private information. This is also called sniffing or snooping. If the traffic is in plain text, the attacker can read the traffic as soon as they gain access to the path by sniffing the wire or wireless signal.

   (a) Keypress snooping

   Keypress snooping aims at capturing confidential information, i.e., passwords, from an unaware target by installing keylogging software on the target's machine.

   (b) Packet sniffing

   Sniffing is used to gain the information being passed on a network, e.g., passwords, credit card numbers. An attacker must gain access to the network TCP/IP traffic path before he/she can capture data packets that make up the conversation, and assemble the packets into a format that is readable to the attacker. It is often the first step in many successful attacks to find system vulnerabilities.

2. Port Scan Attack

   A Port Scan is one of the most popular reconnaissance techniques attackers use to discover services they can break into. A port scan exploits the basic concept that services, running on machines connected to a Local Area Network (LAN) or Internet, listen at TCP or UDP ports. By sending a message to each port, one at a time, the technique can determine whether the port is used and can therefore be probed further to find a weakness. Normally port scanning does not cause direct damage. Potentially it helps the attacker find

which ports are available to launch various attacks. Examples of Port Scan are Stealth scan, SOCKS port probe, Bounce Scan (Javvin Technologies, 2006).

3. Denial of Service

   Denial of Service (DoS) is a well-known attack designed to render a computer or network incapable of providing normal services. The common targets of DoS attacks are a computer's network bandwidth and connectivity. Bandwidth attacks aim to cause unavailability of bandwidth in the network by flooding the network with a high volume of traffic to consume all available network resources so that legitimate user requests cannot get through. On the other hand, connectivity attacks aim at making a server incapable of processing legitimate user requests. By flooding a server with a high volume of connection requests, the server's operating system resources are all consumed.

   (a) Flooding with SYN packets is the simplest DoS connectivity attack. The attacker sends a large number of SYN packets and never acknowledges any of the replies, which causes the recipient to accumulate more records of SYN packets than the software can handle.

   (b) Smurfing is another technique to bring down a target machine in a network by swamping the target with more packets than it can handle; then the attacker can take over the resource. The technique exploits a vulnerability in the Internet Control Message Protocol or ICMP, which enables users to send an echo packet to a remote host to report the availability of a host. The problem arises with broadcast addresses that are shared by a number of hosts. The attacker constructs a packet with a source address forged to be that of a victim, and sends it to a broadcast address. The hosts on the broadcast address will each respond by sending back a packet to the target.

4. Distributed Denial-of-service Attack

   A more advanced DoS attack is Distributed DoS or DDoS. Rather than just exploiting SYN packets or a smurf amplifier, it uses many computers to launch

a coordinated DoS attack against one or more targets. The attacker is able to multiply the effectiveness of the Denial of Service by harnessing a large number of machines over a period of time and installing attacking software. At a predetermined time or on a given signal, these machines all bombard the target site with messages.

5. Spoofing Attack

   A spoofing attack is a situation in which one person or program successfully masquerades as another by falsifying data and thereby gains an illegitimate advantage. There are several scenarios for spoofing attacks, e.g., man-in-the-middle attack, IP spoofing, phishing, email spoofing.

   (a) A man-in-the-middle attack is an attack in which an attacker is able to read, insert and modify at will, messages between a sender and a receiver without either party knowing that the link between them has been compromised. The attacker must be able to observe and intercept messages going between the two victims (Wikipedia, 2006a).

   (b) IP spoofing. An intruder can create packets with spoofed source IP addresses, exploiting applications that use authentication based on IP addresses. This can lead to unauthorized use and possibly root access on the targeted system (Center, 1995).

   (c) Phishing is an attack where a legitimate web page such as a bank's site is reproduced with the same "look and feel" on another server under the control of the attacker. The intent is to fool the users into thinking that they are connected to a trusted site, for instance to harvest user names and passwords. This attack is often performed with the aid of URL spoofing, which exploits web browser bugs in order to display incorrect URLs in the browsers location bar; or with DNS cache poisoning in order to direct the user away from the legitimate site and to the fake one. Once the user puts in their password, the attack-code reports a password error, then redirects the user back to the legitimate site (Wikipedia, 2006c).

   (d) Email spoofing is an attempt to trick users into believing that they

receive email from one source when it actually was sent from another source. Once believed, users potentially reveal sensitive information (such as passwords) or make a damaging statement (Center, 2002).

6. Viruses, Worms, and Trojan Horses

   Viruses, worms and Trojan Horses are all malicious programs that can cause damage to your computer, but there are differences among the three (Beal, 2006).

   (a) A virus is a software program deliberately designed to interfere with computer operations and able to spread itself to other computers on the network or the Internet. Almost all viruses are attached to an executable file, which means they can only infect the machine when a user runs or opens this malicious code. It is important to note that a virus needs human action to spread itself to other machines. People usually do not know that they are spreading viruses by sharing infected files or sending emails with viruses as attachments to the emails.

   (b) A worm is similar to a virus by its design, but it can spread from one computer to another by itself. Taking advantage of file or information transport features on your system, a worm is able to travel without the help of a person. A worm can also replicate itself on the infected machine, so rather than sending one copy of itself, hundreds or thousands of copies can be sent out. Due to the copying nature of a worm and its ability to travel across networks, the final result in most cases is that the worm consumes too much system memory (or network bandwidth), causing Web servers, network servers and individual computers to stop responding.

   (c) A Trojan Horse is a malicious computer program that masquerades as a useful tool. The victim is tricked into believing that it is a legitimate software or file from a legitimate source. The effect of Trojan Horses varies. For instance, a Trojan Horse may install a keylogger or software that allows remote control of the victim's computer, or it can delete

files or destroy information on your system. A Trojan Horse can also create a backdoor on the infected machine that gives access to malicious users, which possibly allows confidential information to be compromised. Unlike viruses and worms, Trojans do not reproduce by infecting other files, nor do they self-replicate.

2.2 Defense Against Attacks

Tools to accomplish attacks are becoming more widespread and easier to use every day. The term "script kiddies" is used for those unskilled attackers who want to experience the fun of attacking. It is known that the most common attacks are launched by script kiddies, rather than skilled attackers. There are four basic techniques to protect networks from attacks. They are configuration management, the application of firewalls, encryption and intrusion detection systems.

2.2.1 Configuration Management

Configuration management is the most critical aspect of securing a network. The tighter the configuration, the more secure the network. To apply a tight configuration management policy, system administrators have to make sure, for example, that all the machines are running up-to-date copies of the operating system, that all the patches have been applied, that no serious hole is left open in any configuration files or services, and that all known default passwords are removed from installed products. It is quite an effort to maintain a network of a number of machines with tight configuration management. There are several tools to help system administrators to keep things tight. Some enable centralized control, which makes management more convenient. However, the task requires not only skilled, but also diligent system administrators and tight configuration can result in limited services the system provides and limited permission for users.

2.2.2 Firewalls

The most widely used solution to protect a network is a firewall, which stands between a local network and the Internet and filters out traffic that might be harmful. A firewall can operate on three different levels, i.e., at the IP packet level, at the TCP session level or at the application level.

At the IP packet level, a firewall simply examines addresses and port numbers of packets entering or leaving the network and accepts or rejects them based on user-defined rules. It can block IP spoofing attacks and stop some denial-of-service attacks. A more complex firewall, called a circuit gateway, operates at the TCP session level. A Circuit Gateway is concerned with allowing or disallowing a session (such as Telnet or FTP), without analysing the content of the transmitted packets. It applies security mechanisms when a connection is established. Once the connection has been made, packets can flow between the hosts without further checking. Operating at the application level is the application relay. It can act as a proxy for some services, e.g., mail, telnet, and Web. The application relay can be set to apply security mechanisms to specific applications, e.g., filter out macros or active content from incoming files or Web pages. This is very effective, but can reduce performance. Even though having firewalls installed for networks still requires a proper configuration and policy, the idea of having a small number of machines to manage, rather than configuring all machines in the network, is attractive.

2.2.3 Encryption

In cryptography, encryption is the process of obscuring information to make it unreadable without special knowledge. It is employed to protect the confidentiality of information; either individual or organizational. Although encryption can ensure secrecy to a certain degree, it is not completely secure to rely only on encryption. Other techniques are still needed to make communications secure, particularly to verify the integrity and authenticity of a message; for example, a message authentication code (MAC) or digital signature.

Encryption can be enforced at different layers, depending on products. For

example, a product called Secure Shell (SSH) encrypts links between two parties. IPv6 provides encryption and/or authentication at the IP layer. In a virtual private network or VPN, data are encrypted at firewalls of communicating companies in the network.

Although secrecy and confidentiality can be guaranteed by encryption, there are other kinds of attacks that cannot be prevented. For example, the problem of malicious code can arise if encrypted mails or Web pages are allowed through the firewall. They can bring all sorts of unwanted things with them.

2.2.4 Intrusion Detection System

Intrusion detection is an attempt to detect inappropriate, incorrect, or anomalous activities, which can cause damage to an organization. In computer networks, intrusion detection systems have attracted high attention as network attacks have increased in number and evolved in technique every day.

An intrusion detection system, or IDS, attempts to detect an intruder breaking into the system or a legitimate user misusing system resources and raises an alarm to notify the proper authority (This authority is from now on referred to the Site Security Officer or SSO).

A History of IDS

In 1980, Anderson wrote a report, in which a term "intrusion detection" was born, with the purpose of improving the computer security auditing and the surveillance capability of the customer's systems (Anderson, 1980). He reported that audit trails contained vital information that could be valuable in tracking misuse and understanding user behavior and outlined the considerations and general design of a system which provided an initial set of tools to computer system security officers for use in their jobs. The report is considered a foundation for later intrusion detection system design and development. His work was the start of host-based intrusion detection and IDS in general.

In 1983, Denning with SRI International began working on a government project in intrusion detection development. Their goal was to analyze audit trails and create profiles of users based on their activities. One year later, the Intrusion Detection Expert System (IDES)(Denning, 1987), which was the first model for intrusion detection, was developed. It was used for tracking and analyzing audit data containing authentication information of users on ARPANET, the precursor of Internet. A model for a commercial intrusion detection system development was published in 1987 by Denning from her research and development work at SRI. The model is the basis for most of the work in IDS that followed.

Meanwhile, there were other significant advances occurring at many laboratories. At Los Alamos National Laboratory LANL, Wisdom and Sense (W&S)(Vaccaro and Liepins, 1989) was developed as an anomaly detection expert system in 1984. In 1988, MIDAS (Sebring et al., 1988) was developed for US National Computer Security Center Multics. In the same year, the Haystack project (Smaha, 1988), at the University of California Davis' Lawrence Livermore Laboratories, released a version of intrusion detection for the US Air Force, In 1990, an idea of network-based intrusion detection was introduced by Heberlein. The Network Security Monitor (NSM) was the first network-based intrusion detection system. This new awareness led to more interest in the field of intrusion detection and investments in that market increased significantly. Simultaneously, the Automated Security Measurement System (ASIM) was developed at the Air Force's Cryptologic Support Center to monitor network traffic on the US Air Force's network. ASIM was able to address scalability and portability issues of most network products. Additionally, ASIM was the first solution to incorporate both a hardware and software solution for network-based intrusion detection. ASIM is still in use and managed by the Air Force's Computer Emergency Response Team (AFCERT) at locations all over the world (Innella, 2001).

Commercial development of intrusion detection technologies began in the early 1990s. The first commercial vendor of IDS tools was Haystack Labs, which was formed by the developers from the Haystack project. Its product was called Stalker, the last generation of the technology from the Haystack project. Science Appli-

cations International Corporation (SAIC) was also developing a system of host-based intrusion detection, called Computer Misuse Detection System (CMDS). In 1994, the development group on the ASIM project formed a commercial company, the Wheel Group. Their product, NetRanger, was the first commercially viable network-based intrusion detection device. The intrusion detection market began to gain in popularity around 1997. In that year, Internet Security Systems (ISS), which is a security market leader, developed a network intrusion detection system called RealSecure. In 1998, Cisco recognized the importance of network intrusion detection and purchased the Wheel Group, incorporating a security solution for their customers. Similarly, a merger of the development staff from Haystack Labs and the departure of the CMDS team from SAIC formed Centrax Corporation. From there, the commercial IDS world expanded its market-base and a cycle of start-up companies, mergers, and acquisitions ensued.

Generally, intrusion detection systems are classified on what patterns or profiles represent. There are two major categories of intrusion detection systems; i.e., anomaly-based and misuse-based. For example, if a system maintains profiles of normal behaviour, it is classified as anomaly-based. On the other hand, a system that maintains profiles of intrusive activities is called a misuse-based system, which is sometimes referred to as a signature-based system.

Anomaly-Based Intrusion Detection System

In anomaly based intrusion detection, profiles of normal behavior are defined in relation to previously observed characteristics. The system reacts to deviations from these profiles. Anomaly-based intrusion detection systems are well-known for their ability to catch novel intrusions as well as variations of known intrusions. This is because intrusions are not covered by the pre-defined profiles. However, this is also a problem. As the system constructs a normal model of subjects from previously observed behaviour, a deviation, which may not be intrusive, might be treated as an intrusion.

A profile for an anomaly-based system can be self-learned or programmed. A

system with self-learning profiles can adapt itself to changes in network environment; however, this comes with a trade-off: a system with self-learning profiles might learn to accept dangerous behaviour as normal when it changes slowly over time, which might cause false negatives. Because of this, hybrid profiles combining two approaches have been proposed. Our paradigm also uses hybrid profiles in detecting anomalies, as will be discussed later.

Misuse-Based Intrusion Detection System

Unlike anomaly-based intrusion detection systems, misuse-based detectors focus on the evidence of intrusive behaviour. They require that the nature, patterns or signatures of intrusive behaviour must be maintained. Such systems are sometimes called signature-based systems.

These signatures would be used against audit data to find evidence of intrusion. With the knowledge of intrusive manifestations, the processing of audit data is simpler and produces less false positives than anomaly based detection as we know what we are looking for. The main advantage of misuse-based (or signature-based) systems is that they can effectively detect known attacks with a low false positive rate. This is because only an observation with a corresponding intrusive signature is treated as an intrusion. The system needs be programmed with an explicit set of decision rules, where what can be expected to be observed in the event of intrusion is specified. This is a highly qualified and time consuming task and is often performed at regular intervals, in a batch oriented fashion. However, an organization can subscribe to some outside sources, where well-known and recently detected intrusion signatures are provided. It is possible that machine learning could be used to build rules, but this would require specific examples to be identified for the learning and cannot be considered as self-learning.

Since the signature database needs to be manually revised for each new type of attack that is discovered, this limitation has led to an increasing interest in anomaly techniques. With pros and cons for both techniques, a hybrid between anomaly and signature has also been proposed.

2.2.5 Other Research

Alternative to traditional ways to protect networks as discussed above, there are studies in network architecture and protocol modification. Yang et al. (Yang, Wetherall, and Anderson, 2005) address DoS attacks by designing a new architecture called Traffic Validation Architecture(TVA). Hosts and routers are extended with some features to limit the impact of packet floods so that two hosts can communicate despite attacks by other hosts. This work is based on *capabilities*, which allows destinations to control the packets they receive.

Another technique is proposed by Walfish et al. (Walfish et al., 2006). The authors address, in particular, application-level DDos attack by the technique called *speak-up*, which tries to slow down the bad clients. The approach encourages all clients to send more data to the server. They suppose that bad clients are already using most of their upload bandwidth, then encouragement will not change their traffic volume. On the other hand, good clients use only small fraction of their available bandwidth, so they will react to encouragement by sending more traffic. As a result, the paper claims that good clients will be better represented in the mix and thereby capture a larger portion of the server. The approach requires a mechanism to cause a client to send more traffic for a single request than it would if the server were unattacked.

2.3 Generic Intrusion Detection System

Many intrusion detection systems(or IDS) are based on the general model proposed by (Denning, 1987). The model, independent of the platform and type of intrusion, has six main components: subjects, objects, audit records, profiles, anomaly records, and activity rules. The system maintains a set of historical profiles of monitored subjects with respect to objects, matches each generated audit record against the profiles, and reports any anomalies detected.

Even though each IDS differs in many aspects from the choice of data source to the processing approach, there are four common tasks in any IDS; i.e., audit data

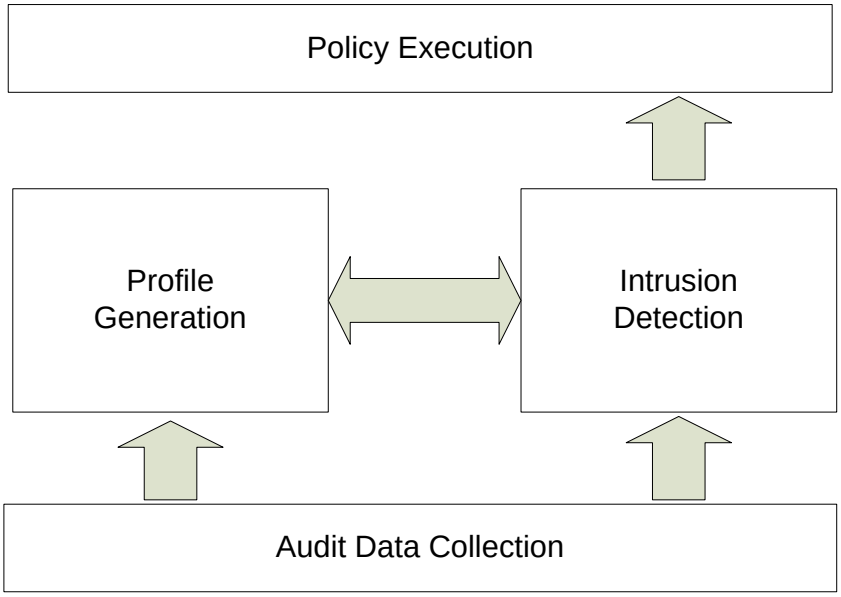


Figure 2.1: Generic tasks of intrusion detection systems.

collection, reference data generation, intrusion detection, and policy execution as shown in Figure 2.1. Each of these tasks is explained below.

2.3.1 Audit Collection and Storage

The very first step in any detection system is to decide what data from the monitored system could be used as audit data. Many different parts of the system can be used as sources of data; for example, keyboard input, command logs, application logs. However, the most common sources of audit data are host-based logs and network traffic data.

Using host-based logs is motivated from the idea that users behave in a consistent manner. In the past, most systems have used host-based logs as audit data; for example, Haystack, IDES, W&S. Haystack (Smaha, 1988), maintains profiles for each user from past behavior and can monitor current behavior of users. IDES (Lunt et al., 1988) calculates statistics from past behavior of users and correlates the current activity with the calculated profiles. W&S (Vaccaro and Liepins, 1989) studies historic user records to produce a set of rules describing the normal behavior of each user.

Current IDSs are likely to monitor network traffic data. Examples of such systems are NSM, EMERALD, SmartSifter. NSM (B. Mukherjee, 1994) was the first IDS solely using network traffic data to analyse the intrusion. On the other hand, EMERALD (Porras and Neumann, 1997) uses both host-based and network traffic data. Another issue is the storage of audit data. Audit data must be stored somewhere, either for later reference or temporarily awaiting processing. Typically, the volume of data is exceedingly large. This is a crucial element in any intrusion detection system.

2.3.2 Reference Data Generation

Reference data or profiles play a central role in intrusion detection systems. They contain the knowledge of the system of interest; for example, the normal behaviour of a network or signatures of known attacks. The type of profile is basically used to categorize intrusion detection systems; an anomaly-based IDS uses profiles of normal behaviour while a signature-based IDS maintains profiles of intrusive behavior.

A profile can also be distinguished by how it is created; i.e., it can be created by self-learning or programmed by a network administrator. With self-learning profiles, the system is able to automatically learn by example the behaviour of a large number of monitored subjects. It is done essentially by observing data for a period of time and building some model of the monitored subjects. There are several techniques used; for example, rule modeling, descriptive statistics, hidden Markov models, artificial neural networks. Ideally, the system can be left to run unattended and profiles are updated when new data is accepted as belonging to the profile.

On the other hand, a programmed detector needs someone, say a network administrator or the SSO, to teach the system about what is considered normal and what is abnormal enough to signal a security violation. Generally, this kind of system relies on predefined descriptive statistics, for example, the number of unsuccessful logins, the number of network connections, which are configured by the

SSO. Along with these monitored statistics are thresholds which are configured by the SSO. When any of these thresholds is exceeded, the system raises an alarm. Techniques that have been used in this approach include state modeling, expert systems, string matching, model based systems, state transition analysis and pattern matching.

A computer network is a dynamic environment where its attributes, such as the number of users, the link capacity, etc., can change over time and new attacks emerge every day. Profiles should be able to capture the most recent situation of the monitored network. The pros and cons of each kind of profile mainly involve profile update. A system with programmed profiles requires human interaction to update its profiles. It lacks a learning capability; however, a self-learning system might learn to accept dangerous behaviour as normal when it changes slowly over time.

2.3.3 Intrusion Detection

An intrusion detection system tries to detect potentially intrusive characteristics from audit data against reference data and raise an alarm to notify the SSO. A detection engine in general is an algorithm to produce an output for an audit record based on the system's reference data. The output here is a conclusion about whether an audit record is an anomaly. A detection engine is tightly coupled with the representation of reference data or knowledge of the system. For example, a distribution is used to represent a parameter. To detect an anomaly, the detection engine might calculate the probability of an observed value that follows that distribution. If the probability is within a confidence interval, that value is treated as normal. Another example is that system behavior might be captured in rules. The detection engine then uses an inference technique to derive a conclusion about an audit record.

There are many techniques that have been proposed and implemented for detecting intrusions. The earliest used statistical measures to specify valid thresholds for parameters. Many AI approaches, e.g., expert systems and neural networks have

also been used.

Another element to take into account when considering techniques for intrusion detection is time series or temporal relationships. Some techniques can handle the order and interarrival times of the observations x_1, ... , x_n, as well as their values, but the others cannot. A time series has the advantage of measuring trends in behaviour over time and detecting gradual but significant shifts in behaviour.

Statistical Approach

Using statistics to identify how anomalous an event is, has been widely used in intrusion detection for many years. The system basically collects data on certain network parameters and applies some statistical techniques on the collected data to generate profiles for parameters, for example, a system might learn the distribution of a monitored parameter. The system then considers the variance of monitored parameters from their profiles; i.e., the higher the variance, the more likely the network is experiencing an attack. The system may apply simple rules which can be defined by the SSO to apply to the variance. Threshold is the simplest form of a rule that is predefined for the system to raise an alarm when the monitored parameters exceed it. Examples of IDS using descriptive statistics are Haystack (Smaha, 1988), IDES (Lunt et al., 1988), EMERALD (Porras and Neumann, 1997), etc.

Expert Systems

Cannady stated that the majority of techniques employed to detect network intrusions use expert systems (Cannady, 1998). An expert system consists of a set of rules encoding the knowledge of human expertise. These rules are used, by an inference engine, to make conclusions about security-related data. A rule is in the form of *IF (antecedent) THEN (consequent)* statement, which is close to the natural thinking of humans. The ease of capturing knowledge and formulating rules makes this approach attractive. This form also provides modularity, from which the knowledge engineer can easily review, verify, and correct each rule without, in

theory, any effect on other rules. But as we will see, in practice, rule addition is more difficult than this.

Neural Networks

With the capability to provide a probability estimate that the data matches the characteristics which it has been trained to recognize, neural networks have been proposed for intrusion detection arena. The neural network gains experience initially by training the system to correctly identify preselected examples of the problems. Its response is reviewed and the configuration is refined until it reaches a satisfactory level. In addition to the initial training period, the neural network gains experience over time as it conducts analyses on data related to the problem. Neural networks have been used in both anomaly-based and misuse-based system. Examples of anomaly-based intrusion detectors using neural network are Hyperview (Debar, Becker, and Siboni, 1992) and NNID (Ryan, Lin, and Miikkulainen, 1998).

State Modeling

The basis of this approach is to encode data as an automaton-like diagram with a number of states and transitions. For example, intrusions are the input to the state generation mechanism in misuse-based detection, whereas, a security benign operation is encoded as a set of states in anomaly-based detection. As data is analyzed, the system makes transitions from one state to another. Examples of systems using state modeling are USTAT (Ilgun, 1993), IDIOT (Kumar and Spafford, 1994).

Predictive Pattern Generation

Based on the assumption that sequences of events from a particular subject are not totally random, but follow a perceptible pattern, this technique was proposed to predict future events, given a sequence of events that have already occurred (Teng, Chen, and Lu, 1990). The approach incorporates time-based rules and probability. The following rule:

$$E_1 \rightarrow E_2 \rightarrow E_3 \Rightarrow (E_4 = 95\%, E_5 = 5\%)$$

means that given a sequence of observed events E1, E2, and E3, the probability of seeing E4 is 95% and that of E5 is 5%. This probability is based on previously observed data. The rules are modified dynamically during the learning phase and low quality rules are continuously eliminated.

2.3.4 Policy Execution

The last task, policy execution, is when an action is taken when an intrusion is detected, i.e., to notify the SSO, or update related reference profiles. Action is generally predefined by the SSO depending on the level of intrusions detected.

2.4 Intrusion Detection Systems: Examples

In this section, some intrusion detection systems are briefly reviewed. We will see that some systems use one technique to detect anomalies, while others might use a combination of techniques. A discussion of these systems is provided after they are reviewed.

2.4.1 Haystack

Haystack (Smaha, 1988) was developed for the detection of intrusions based on users' log profiles. It was designed to detect six different types of intrusion as discussed earlier, i.e., attempted break-in, masquerade attack, penetration of security control system, information leakage, denial of service and malicious use. The system employs two methods of detection; i.e., both anomaly-based and misuse-based detection. With misuse-based detection, Haystack looks for evidence of predefined "bad" behaviour and atypical or suspicious behaviour. A suspicion quotient is used as a measure of the abnormality of the session with respect to a particular weighting of features. Haystack computes a weighted multinomial suspicion quotient from a list of features whose values are outside the expected security ranges and

the estimated significance of each feature violation for detecting a target intrusion (B. Mukherjee, 1994).

Using statistics from past behaviour, anomaly detection is performed on two model types; i.e., a per user model and a generic user group model. About two dozen features of a user's session are monitored, including time of work, number of files created, number of pages printed, etc. Smaha states that if the anomaly-based detection sub-system attempts to learn suspicious behaviour that is actually normal, the SSO can choose to process past user sessions to look for trends that could indicate this. However, the author has not suggested how to present the information to the SSO.

2.4.2 MIDAS

MIDAS (Sebring et al., 1988) was developed by the National Computer Security Centre, in cooperation with the Computer Science Laboratory, SRI International. From observation of how SSOs analysed audit logs manually to find evidence of intrusive behaviour, the authors identified that successful (manual) intrusion detection involves knowledge and symbolic reasoning with a measure of uncertainty. That was why they decided to use a rule-based expert system to the task.

MIDAS uses the Production Based Expert System or P-BEST, which is a forward chaining expert system shell, in which the introduction of a new fact in its fact base triggers the re-evaluation of the rule base. Rules are in three distinct categories, i.e. immediate attack, user anomaly, and system state. Aiming at known attacks, the immediate attack heuristics are static, predefined by the SSO, and operate without any background knowledge or statistics of the system. On the other hand, user anomaly and system state use statistical profiles of past behaviour to detect deviations from them. In the class of user anomaly heuristics, there are two types of profiles; i.e., a session profile and a user profile. The final class of rules, system state, maintains knowledge about the system statistics as a whole, without concerning individual users.

The structure of the rule base is multi-tiered; a conclusion at a lower layer can

cause the firing of a rule the next higher layer (Axelsson, 2000; B. Mukherjee, 1994). In general, the lower layer handles deduction about some types of events, e.g., the number of bad logins, and asserts some facts that some threshold of suspicion is reached, e.g., bad logins is 5. The process then passes to the next higher layer, which decides whether to actually raise an alarm.

The authors stated that although MIDAS could detect simulated intrusion attempts, it gave too many false alarms. They intended to employ other algorithms for anomaly detection and some means to validate the rule base for completeness and consistency.

2.4.3 IDES

The Intrusion Detection Expert System or IDES (Lunt et al., 1988) is one of the classic intrusion detection systems. The IDES project had been underway for a number of years, continuing into the Next-generation Intrusion Detection Expert System or NIDES, after the IDES project was officially finished. A number of versions were implemented under the project; some with minor change, some with fundamental changes. This review will focus on the original IDES.

The basic motivation is that users behave in consistent manner from time to time which can be summarized by calculating various statistics for the user's behaviour. Then current activity can be correlated with the self-learned profile, and deviations can be flagged as anomalous behaviour. IDES monitors three types of subjects; i.e., users, remote hosts, and target systems. Thirty-six parameters are monitored; 25 for users, 6 for remote hosts, and 5 for target systems. IDES measures these parameters during each user session (from login-time to exit-time). Based on these parameters, profiles on subjects are self-generated, and are typically updated to reflect new user behaviour once a day. IDES then uses an expert system to verify each new audit record against the known profiles. It also addresses the problem of different but authorized behaviour of subjects by defining two sets of profiles for the monitored subjects depending on whether the activity takes place on a weekday or a weekend. IDES is also able to extrapolate from current statistics

and compares this extrapolation with the profile for the subject. This is to prevent the system from reporting abnormalities on some continuous parameters when it is roughly half-way through the session.

Complemented with anomaly detection, IDES employs an expert system technique for signature-based detection. This sub-system detects network intrusions based on rules regarding known attack scenarios, known system vulnerabilities, site-specific security information and expected system behaviour. It is implemented using P-BEST.

2.4.4 NIDES

The Next-generation Intrusion Detection Expert System or NIDES (Anderson, Frivold, and Valdes, 1995) is a direct continuation of IDES. NIDES follows the same general principles of IDES, i.e., it has a strong anomaly detection foundation, complemented with a signature based expert system environment. There are four major versions of NIDES development, each with refinements based on input data. The four versions are presented in the following paragraphs.

NIDES-Alpha In this version, the same functionality of IDES is preserved, except that the architecture is changed; i.e., more modular and built on a client-server architecture. The anomaly detection functionality is also enhanced to deal not only with simpler parameterised distributions, but also with multi modal distributions.

NIDES-Alpha Patch There were three changes made in an attempt to improve NIDES-Alpha performance. Firstly, the information retrieval process was changed: rather than traversing the entire list at the time of audit record processing, this process was moved to the profile generation stage. Secondly, a feature was added to allow the NIDES-Alpha to update profiles based on the audit record timestamps instead of the real time clock. Finally, a user configurable subject profile cache was added to speed up processing in the anomaly detection module.

NIDES-Beta In this version, several new features were added. These features include an optimised profile storage structure, real-time configuration on NIDES analysis, expanded status reporting, data management facility, and expanded rule base (from 21 to 39 rules).

NIDES-Beta Update This version is considered the final release of NIDES. The main improvement was in handling file access statistics which enabled NIDES to process difficult cases in a matter of hours, instead of aborting processing. Other new features included the introduction of Perl scripts to allow NIDES to work in a cross-platform environment, an enhanced capability to detect the use of sniffers, and an expanded audit record fact template to enable the rule based detection part to consider all the available fields in audit records.

Experiments demonstrate that NIDES is capable of detecting anomalies of interest and both false positive and false negative rates can be kept at reasonable levels.

2.4.5 Wisdom and Sense

The Wisdom&Sense or W&S (Vaccaro and Liepins, 1989) is an example of a self-learning anomaly-based system with a rule modeling approach.

W&S was designed with a number of requirements, including[1]

- to reduce audit data to more usable forms.
- to build its own rule base without human guidance. [W&S studies historic audit data and uses decision tree learner to produce a forest of rules describing "normal" behaviour.]
- to store and use very large, instantiated rule bases efficiently.
- to tolerate conflicting rules.

[1]Verbatim from (Axelsson, 2000; Vaccaro and Liepins, 1989)

- to deal with uncertain and erroneous knowledge.
- to continue to learn from experience and adapt to transient conditions.
- to accept human modifications to its rule base, but not be overly dependent on scarce human expertise.
- to make real time graded decisions regarding anomalous behaviour.
- to provide human-readable feedback on anomalies to aid in anomaly resolution.
- to create minimal interference with the real functions of its host system.
- to be portable to different applications, operating systems, and hardware.

An audit record for W&S is typically one event for each process execution. W&S studies historic user records and produces a set of rules describing the normal behaviour of a user, irrespective of the temporal relationships among those records. The author suggested that around 500-1000 records per use is a good target to aim for. These rules are fed into an expert system that evaluates a recent audit data for violations of the rules and alerts the SSO when the rules indicate an anomaly.

W&S reads the rule base, dictionary and new data records, either from a file in batch mode or as they become available. It then evaluates the thread class of which they are part and reports an anomaly whenever the thread score or individual audit record score exceeds an operator defined threshold. The audit record score is derived from summing all contributions from the different failed rules of that record. The thread score is a sum of all scores from that thread's audit records.

Experiments with staged intrusion attempts show that the system is time efficient in detecting anomalies. However, the authors suggested further research into the nature of the computer security threat.

2.4.6 EMERALD

Event Monitoring Enabling Responses to Anomalous Live Disturbances or EMERALD (Porras and Neumann, 1997) is a framework for a scalable, distributed, in-

teroperable intrusion detection system, aiming at detecting penetration external to the organisation. It is designed to operate on three different levels; i.e., service analysis level, domain-wide level, and enterprise-wide level.

The architecture of EMERALD builds around local EMERALD monitors, which are distributed and independently tunable at different levels. Each monitor communicates with others which are distributed throughout the network. These monitors combine signature-based analysis with statistical profiling to provide localized real-time protection of the most widely used network services on the Internet. The monitor consists of four main modules; namely, resource object, profiler engine, signature engine and universal resolver.

The module resource object handles all target specific issues. All configuration parameters are stored here. It also maintains the subscription list for communication with its peers.

The profiler engine performs some anomaly-based detection on audit data. Inspired by the IDES and NIDES statistics component, the profiler engine separates profile management from the mathematical algorithms used to assess the anomaly of events. Profiles are provided as classes defined in the resource object.

The signature engine provides signature-based detection. It is operated with a small set of rules and on a reduced audit data stream, which contains much less noise.

The universal resolver is the centre of processing. It handles correlation between the result of local modules, decides whether an intrusion occurs, decides whether a response should be invoked. It also handles communications with the peer monitors. The universal resolver applies an expert system to infer reports from the profiler engine, the signature engine, and other peer monitors to decide what response to invoke.

2.4.7 NSM

Network Security Monitor or NSM (Heberlein et al., 1990) is the first system to use network traffic directly as the source of audit data. Using standard protocols, such as TCP/IP, telnet, ftp, etc., NSM can monitor a network of heterogeneous machines without having to convert a variety of audit data into some canonical format. NSM listens passively to all network traffic that passes through a LAN network and detects intrusive behaviour from this input.

NSM uses host vectors and connection vectors as input to a simple expert system that analyzes the data for intrusions. The expert system takes other input, such as profiles of expected traffic behaviour, knowledge about capabilities of each network service, the level of authentication required for each service, the level of security for each machine, and the signature of past attacks From these input, NSM makes a decision, based on anomaly reasoning, about the likelihood that a particular connection represents intrusive behaviour, or if a host has been compromised.

The suspicion level of a particular connection is a function of four factors, i.e. the abnormality of the connection, the security level of the service being used for the connection, the direction of the connection sensitivity level, and the matched signatures of attacks in the data stream for that connection. Results are shown to the SSO in the form of a sorted list, where each row in the list consist of a connection vector and the computed suspicion level.

NSM is another hybrid system combining signature-based detection with anomaly-based reasoning. The prototype system was deployed at UC Davis Lawrence Livermore National Laboratory, Department of Energy and US Air Force sites. It has been reported to be satisfactory in correctly identifying intrusive behaviour.

2.4.8 Hyperview

Hyperview (Debar, Becker, and Siboni, 1992) is a hybrid system. It consists of two main components; i.e., an expert system that monitors audit trails for signs of known intrusions, and a neural network that adaptively learns the behaviour of a

user and raises an alarm when the audit trail deviates from the learned behaviour. The audit trail is host-based and from a number of different sources with different details, for example, keystrokes made by the user, commands issued by the user.

In the anomaly-based detection component, a neural network is chosen to learn a multivariate time series from the audit trail. The neural network is connected to an expert system that monitors the operation and training of the network to prevent the network from learning anomalous behaviour, and evaluates its output. The output of this component is then fed to the other component, the signature-based detection.

A further expert system is employed to match audit trails and output from the first expert system against known intrusive traces. It decides whether to raise an alarm or not.

2.4.9 USTAT

USTAT (Ilgun, 1993) is a mature prototype implementation of the state transition analysis approach to intrusion detection. The technique assumes that a computer is initially in some secure states, and it ends up in a compromised target state via a number of penetrations, which are viewed as state transitions. The SSO is required to specify those state transitions against known intrusions. USTAT uses this specifications to evaluate an audit trail.

USTAT consists of four major modules, namely audit collection, knowledge base, inference engine, and decision engine. The audit collection module is to collect audit data and to store that data for future reference. The knowledge base module consists of two components, i.e., the fact base and the rule base. The fact base contains information about the objects in the system. The rule base, on the other hand, contains the state transition diagrams that describe intrusion scenario. The inference engine evaluates a new audit record using information from both rule base and fact base. It also updates the fact base with state information. The decision engine reports a detected intrusion to the SSO, and provides a suggestion of possible actions to preempt a state transition that can lead to a compromised

state.

2.4.10 IDIOT

The IDIOT (Kumar and Spafford, 1994) is a misuse-based intrusion detection system that was developed at COAST, University of Purdue. The system divides the intrusion detection effort into three distinct abstraction layers, i.e., an information layer, signature layer, and pattern matching engine. In the information layer, any machine/platform dependencies in the audit data are isolated. In the signature layer, the signatures of intrusive behaviour are described in a system independent fashion. Finally, preprocessed audit data is matched against predefined signatures by the matching engine.

The system is based on a number of matching engines independently analyzing an audit record. The authors employ coloured Petri nets to represent intrusion signatures. When the system starts, it reads an audit record and passes it through all pattern matching engines, each of which matches an audit record against its predefined intrusion pattern to decide whether to update its state according to the audit record.

2.4.11 Snort

Snort is the most popular open-source network intrusion detection system (Beale et al., 2003; Cartwright, 2007). It was originally written as a packet sniffer by Martin Roesch in November 1998. Signature-based analysis was added later in January 1999. It can detect a variety of attacks and probes, such as buffer overflows, stealth port scans, web application attacks, SMB probes. Snort is now a free software and open source package, owned and developed by Sourcefire, of which Roesch is a founder and chief technical officer. The package provides many network monitoring functions, e.g., it can perform protocol analysis, content searching/matching, and intrusion prevention, i.e., dropping packets deemed to be attacks. Its advantage is that users are free to add domain specific functions into a rule base. However, configuration is its disadvantage because it must be done by editing the text-based

config file (Beale et al., 2003; Cartwright, 2007; Wikipedia, 2007c; Lizard, 2002).

Snort consists of four main components, i.e., a packet sniffer, a preprocessor, a detection engine, and an alert/logging unit. Packets are acquired by a packet sniffer and passed to a preprocessor, which checks raw packets against certain plug-ins. These plug-ins check for a certain type of behavior from a packet. Once a packet is determined to have a particular type of behavior, it is sent to a detection engine. At a detection engine, data from preprocessor is checked through a set of rules. If the rules match the data in the packet, then they are sent to alert processor. Rules are built on specific attack characteristics, e.g., server CPU utilization, specific types of network traffic, and other numeric characteristics easily measurable and likely to be affected by an intrusion. Users are free to define any domain-specific variables to use within rules. This provides flexibility with simplicity to users and makes the system widely used (Beale et al., 2003; Cartwright, 2007).

2.5 Traffic Volume Anomaly Detection

It is natural that a system is developed appropriate for the nature of the audit data available. Earlier, when only stand-alone machines were available, log files contained only actions made on the machine, e.g., key stokes, commands being executed. When networks were introduced, most of the focus was on issues such as connectivity, availability, data integrity, etc. At that time, network traffic logs were not available; most available tools operated on host-based data, for example, Haystack (Smaha, 1988), IDES (Lunt et al., 1988), W&S (Vaccaro and Liepins, 1989).

A computer network gained more widespread use, new tools to support network management were released. These tools basically provided details of network activities to a network administrator. For example, a packet sniffer is a tool that can intercept traffic passing over a network. A network administrator can look into the contents of packets in the network. Hence, a trend for intrusion detection systems has been to move network-based audit data. For example, NSM (B. Mukherjee, 1994), EMERALD (Porras and Neumann, 1997), and SmartSifter

(Yamanishi, Takeuchi, and Williams, 2000; Yamanishi and Takeuchi, 2001) used network-based data to analyze intrusive behaviour. The most widely used network data are TCP connections, which have become a standard for most intrusion detection systems. They have also been used as test data set for the well-known IDS competition, the KDD-Cup, which is organized by the Association for Computing Machinery or ACM. However, analysis of TCP connections is considered to be high in computational complexity.

More recently, a functionality that most network management tools must provide is a graphical visualization of network utilization. The graph generally shows on-going network status regarding the volume of a parameter, for example, WWW usage, FTP usage, etc. Graphical visualization is more expressive and easier to understand than packet header information. This is why most network administrators nowadays perform their network monitoring task by inspecting graphs of network utilization. There are studies, for example, in (Barford and Plonka, 2001; Lakhina, Crovella, and Diot, 2004a), demonstrating that investigating the volume of traffic over a network can effectively reveal traces of intrusive behavior.

2.5.1 Traffic Volume: An Alternative Audit Data

Traffic volume data is a sequence or time series of bandwidth usage in a network. Traffic volume data can be collected by many network tools such as FlowScan (Caida, 2006), RRDtool (Oetiker and GNU, 2006). These tools are equipped with many functionalities from archiving a variety of network data to graphical visualization. Generally, they archive network usage by protocol, i.e., WWW, FTP. Users can choose to view the usage overall or a particular protocol. The archived data is visualized in a graphical form.

Figure 2.2 is an example of a graph of collected traffic volume generated by RRDtool. The graph is the usage volume (Y-axis) against time (X-axis). Depending on tools, the timespan of a graph can be set arbitrarily.

From a graphical visualization of traffic volume, a network administrator can study network characteristics, for example, during what time of the day the net-

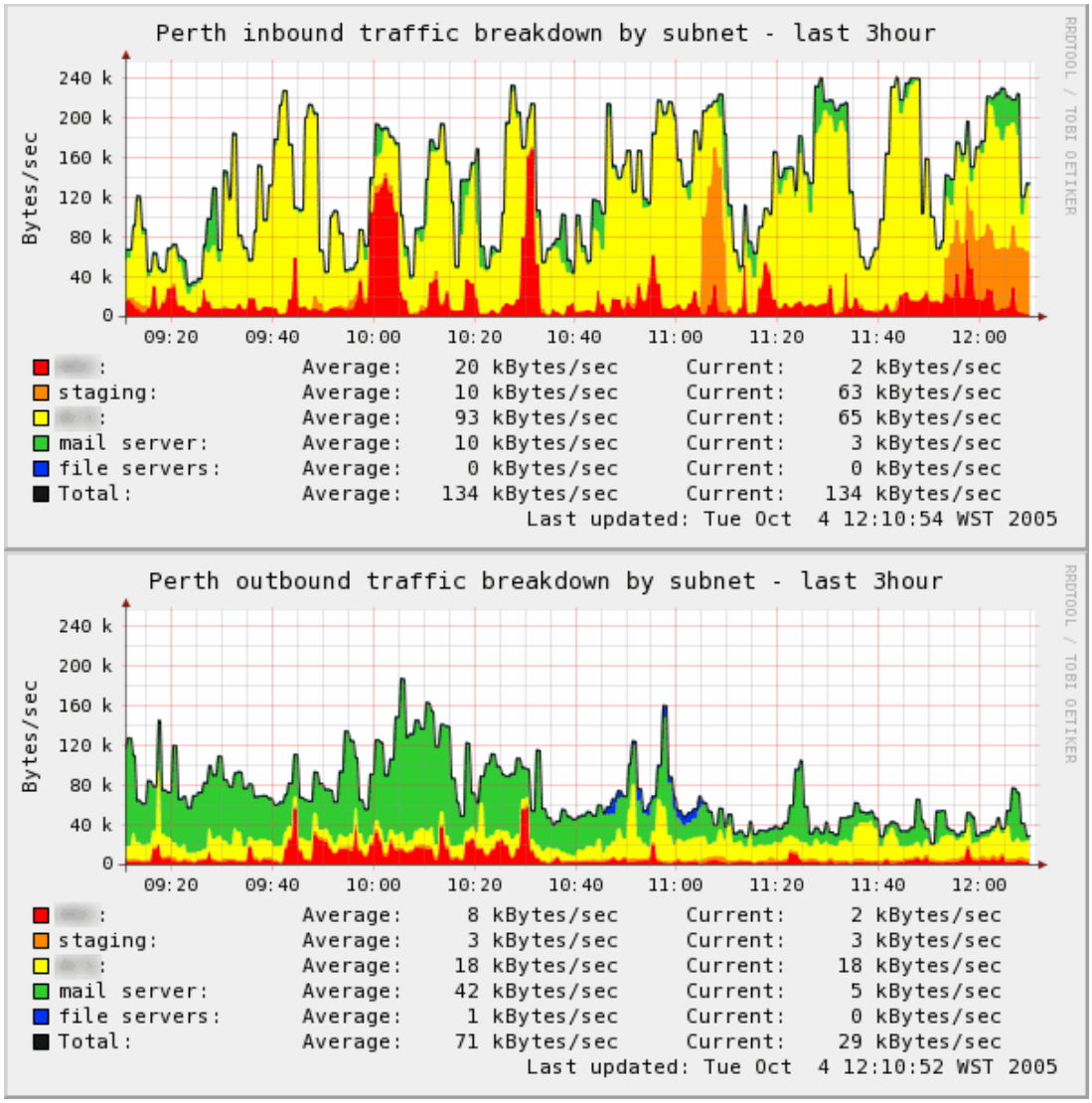


Figure 2.2: An example of graphical visualization of traffic volume. These graphs show network traffic inbound and outbound on a WAN link (Oetiker and GNU, 2006).

work is highly congested, what is the maximum usage, or when the network is mostly idle. On top of the knowledge he or she can learn from the graph, a network administrator can inspect the performance and the status of the network, for example, when an outage occurs or when it is under an attack.

When a network is attacked, traffic volume will significantly change from its normal state; either increasing or dropping for a period of time, depending on the attack type. This helps network administrators detect anomalous traffic. Then they can track down the source of the anomaly. It has been demonstrated that visual analysis of traffic flow can lead to anomaly identification and the characteristics of anomalies that can be detected by the graphical visualization have been investigated (Barford and Plonka, 2001; Lakhina, Crovella, and Diot, 2004a).

2.5.2 Intrusion Detection on Traffic Volume

Recently, there have been many attempts to detect anomalous activities in the network automatically by inspecting traffic volume. Traffic volume is a kind of time series data. In statistics, time series have long been investigated. Many approaches have been proposed to model time series, to forecast new observation, to detect outliers, etc. Those techniques range from simple statistical measurement such as mean and variance, to more complex techniques such as Holt-Winters, Wavelet, etc. This section gives a brief review of studies of intrusion detection based on traffic volume.

Simple Statistical Measurement

(Mandjes, Saniee, and Stolyar, 2005) investigated anomaly detection in IP networks. The focus of their study was to derive a formula that can simplify voice over IP traffic. The authors used simple statistical measurements, i.e., mean and variance of byte count measurements, to construct general formulae for the variance of cumulative traffic over a fixed time interval. The formula is known as Riordan formula (Mandjes, Saniee, and Stolyar, 2005), which provides an estimate for the variance of the VoIP load that passes through a monitored switch or router

interface. They suggested that standard measurements from a router, for example SNMP MIBS, were sufficient for their technique.

A further study on anomaly detection showed that cumulative traffic inspected every 5 minutes and simple statistics such as mean and variance could detect over/underload anomalies efficiently.

Holt-Winters Forecasting Algorithm

The Holt-Winters algorithm is a well-known forecasting algorithm. It is basically a quantitative forecasting method that uses mathematical recursive functions to predict trend behavior. It uses a time series model to make predictions assuming that the future will follow the same pattern as the past.

Brutlag had the algorithm integrated into the open source software RRDtool (Oetiker and GNU, 2006) and Cricket (Allen, 2003; Allen, 1999), to enable real-time monitoring at the IAP/ISP level (Brutlag, 2000). He also suggested to having a window length of at most an hour, i.e., for five minute intervals, the window length should be between 9 and 12 intervals.

From a derived network traffic model, an anomaly was detected if an observed value of time series fell outside a confidence band. Confidence bands in his work were also variable, i.e., updated via an exponential smoothing algorithm (Brutlag, 2000). He suggested that a moving window of a fixed number of observations should be used to reduce the number of false positives.

His experiments on WebTV traffic data showed promising results in anomaly detection. Unfortunately, there was no figure for false positives or false negatives published in his work to enable comparison.

Wavelet Filter

Focusing only on time-frequency characteristics of traffic, Barford et al. applied a well-known signal analysis technique, namely, Wavelets (Goupillaud, Grossmann, and Morlet, 1984) to investigate four classes of network traffic anomalies, i.e.,

outages, flash crowds, attacks and measurement failures (Barford et al., 2002).

The term "wavelet" is defined by Morlet (Goupillaud, Grossmann, and Morlet, 1984). The theory refers to the representation of a signal in terms of a finite length or fast decaying oscillating waveform (known as the mother wavelet). This waveform is scaled and translated to match the input signal.

In their work, wavelet filters are used to extract component signals from the IP-flow and SNMP measurements collected over a six-month period at the border router of a university. Their experiments demonstrated that both ambient and anomalous traffic were noticeable in those extracted component signals. The authors proposed to detect sharp spikes by calculating a deviation score for local variance of the signal. They have found that most anomalies had deviation scores of 2.0 or higher.

Barford et al. compared their approach with the classic Holt-Winters forecasting algorithm. The technique is more capable of detecting anomalies than the Holt-Winters algorithm (Barford et al., 2002).

Principal Component Analysis

Principal Component Analysis or PCA is technique for simplifying a dataset, by reducing multidimensional datasets to lower dimensions for analysis. Technically speaking, it is a coordinate transformation method that maps a given set of data points onto new axes. These axes are called the principle axes or principal components (Lakhina, Crovella, and Diot, 2004b; Wikipedia, 2006b).

Taking advantage of PCA, Lakhina et al. proposed to diagnose network anomalies by separating the high-dimensional space of network traffic measurements into disjoint subspaces of normal and anomalous network conditions and comparing these subspaces with predefined models (Lakhina, Crovella, and Diot, 2004b).

The authors conducted experiments on measurements of traffic links in backbone networks, namely, Sprint-1, Sprint-2 and Abilene. They evaluated the efficiency of their technique by validating against OD flow data. An OD flow is defined as *the traffic that enters the backbone at the origin node or Point of Presence (PoP)*

and exits at the destination PoP (Lakhina, Crovella, and Diot, 2004b). That is an OD flow can propagate through more than one link from its origin to the destination. For example, an OD flow "a-d" might propagate from link a-b, through link b-c, to link c-d. Therefore, the traffic observed on each backbone link is the superposition of OD flows.

Their system was configured to work with data binned on 10 minute intervals. Their investigations have shown that the technique can detect volume anomalies and estimate the amount of traffic involved in the anomalous Origin-Destination flow.

Sketch

Unlike other investigations that detect anomalies from network traffic measurements, Krishnamurthy et al. (Krishnamurthy et al., 2003) propose to detect anomalous traffic from derived summaries of traffic data. A summary of network traffic data is derived from a sketch-based technique. Sketch (Jung, Krishnamurthy, and Rabinovich, 2002; Gilbert et al., 2001; Datar and muthukrishnan, 2001) is a probabilistic summary technique for analyzing large streaming data sets. The authors name this new data structure *a k-ary sketch* (Krishnamurthy et al., 2003).

Firstly, the authors summarize network traffic into k-ary sketches and then implement a variety of time series forecasting models, for example, ARIMA, Holt-Winters, etc., on top of those sketches. Anomalies are marked by detecting flows with large forecast errors.

Spectral Analysis

Spectral analysis has also been applied to this area as an analysis tool. Cheng et al. (Cheng, Kung, and Tan, 2002) propose to use spectral analysis to complement existing DoS defense mechanisms. Spectral analysis is used to identify normal TCP traffic so that, when used after other DoS defense methods which identify traffic aggregate as candidates for attack traffic, it can rule out those candidates which are considered to be normal TCP traffic. As they suggested that normal

TCP flow should exhibit strong periodicity around its round-trip time, the number of packet arrivals of a flow in fixed-length intervals is used as input signal. The power spectral density is estimated to reveal periodicity of signal. A threshold of 5e-3 was used to determine whether a flow is normal TCP traffic. It has been shown that the technique can correctly identify normal TCP flow at the rate of 81.8%, while false positive rate and false negative rate are at 15.7% and 18.2%, respectively. When the technique is used on top of other DoS defense mechanisms such as Aggregate Congestion Control (ACC) and Pushback, the authors report that the volume threshold to flag flows as suspicious is lower and the false positives are reduced. However, they do not state any number explicitly. As usual, the technique relies on a threshold. This means the approach needs some training sets to estimate an appropriate threshold.

Another use of spectral analysis is proposed by Hussain et al. (Hussain, Heidemann, and Papadopoulos, 2003; Hussain, Heidemann, and Papadopoulos, 2006). The authors propose a method to identify repeated attacks that are originated from the same set of attackers using attack fingerprints of seen scenarios, i.e., a combination of attacking hosts and attack tool. The work only focuses on flooding attacks with an assumption that when attacked, most other traffic is squeezed out. Attack fingerprints are generated based on power spectral density that are calculated from attack streams, which are isolated from other network traffic using known attack signatures. An attack fingerprint is tested against many emulated attack scenarios, including the same one the fingerprint is generated from. It has been reported that attacks have the most accurate match against themselves. The paper also shows experiments to identify and isolate factors that affect the attack fingerprint.

2.5.3 Discussion on Traffic Volume Anomaly Detection

In the past 10 years, network measurement such as traffic volume has increasingly been used in network intrusion detection. An increasing number of studies have been published, especially at the Internet Measurement Conference or IMC.

From the literature, most systems are anomaly-based, i.e., background knowledge of normal circumstances must be profiled as reference models. Great efforts have been made in the area of accurate and efficient models of network traffic. These include techniques in statistics and mathematics as seen in the previous section.

One thing in common among the techniques proposed is that they try to derive a single general model for the network traffic. To construct a universal model for time series usually requires some complex formulae for the model and some training sessions for a model to be learnt. In general, the more complex the model, the more complex patterns it can cover but the more training data that is required. This means the system cannot easily adapt to changing behavior, new schedules, or occasional irregular events (such as a conference being held), which may affect traffic patterns.

In our study, network measurement from the school of Computer Science and Engineering, the University of New South Wales is used as audit data. Instead of learning a generic model for network behavior, we partition a problem space into smaller sub-spaces of homogeneous traffic patterns. With this solution, a new pattern can be added when required without affecting other patterns. The methodology will be explained in detail later in this thesis.

2.6 Limitations and Questions of Intrusion Detection Systems

The main aim of an intrusion detection system is to find intrusion attempts so that the proper action can be taken. To accomplish this aim, there are some issues that have emerged since such systems were first introduced.

System Performance

The first concern is the problem of false alarms and false negatives. A false alarm or false positive is a warning that is incorrectly raised in a benign situation. This

is one of the foremost issues in intrusion detection systems, as they are likely to produce a considerable number of false positives. A high degree of false alarms causes human experts to investigate a problem for nothing, resulting in an extra cost to organizations. Also the problem of missing actual intrusions might occur because of too many false positives. Failing to sound an alarm when someone gains or attempts to gain access, on the other hand, can cause more serious problems.

Detection Paradigm

As mentioned earlier, there are two main paradigms for such systems, i.e., anomaly detection or misuse detection. Each has its own virtue; however, some trade offs are present. In anomaly detection, the system is more likely detect new intrusions, but probably with a high false positive rate. Misuse detection, on the other hand, produces significantly less false positives as it matches observations against intrusion signatures. However, it lacks the capability to detect new intrusions. An alternative is the hybrid paradigm.

Profiling Strategy

Both issues of system performance relate to how well profiles represent the domain. A too specific profile which is less tolerant of changes or trends is more likely to incur high false positives, whereas a too general profile which has wider coverage than expected can easily fail to detect anomalous attempts. Modeling techniques are always the core of the problem.

Another concern for the profiling strategy is rare events or novel phenomena. Statistical or learning methods do not easily accommodate individual special cases.

Audit data

There are two major sources of audit data in anomaly detection systems, i.e., network data and host based data. Early systems investigated networks using host-based data. The trend for audit data in recent systems is for network data.

Another concern is the features to be investigated. As data consist of numerous attributes, to have an intrusion detection system monitor the right features is a

critical decision.

Resource Consumption

The more advanced the communications technology, the higher the volume of data the network can transfer. Typical IDSs store their data not only while they are processing, but also for future reference. As network traffic increases, the IDS system may not cope with the volume of data or may have to sample less frequently.

Real time Detection

Early systems detected anomalous behaviour in batch mode; this means they inspected the network from data archives off-line. More recent systems put more concern on real-time detection. This does not mean batch mode detection is not necessary. There are many cases, e.g., attacks manifest in a series of anomalous events, where anomalies can be accurately depicted after they occur and in some circumstances this is more desirable than being given an immediate warning when something is amiss, where the warning may be a false positive.

Chapter 3

Knowledge Based Systems

An expert system gains its power from the knowledge it contains. It is critical how it captures the expert's understanding of the domain. Therefore, efforts are made to assure the effectiveness of the process. The task is known for its tediousness where knowledge engineers must interact with domain experts to acquire, organize and study the knowledge within a domain. This task is called knowledge acquisition and remains a challenge in developing an expert system (Puppe, 1993).

An expert system is defined by Durkin as "A computer program designed to model the problem-solving ability of a human expert" (Durkin, 1994). There are two major components in an expert system: a knowledge base and an inference engine, as shown in Figure 3.1. The knowledge base is a database containing highly specialized knowledge of human experts about the problem domain. It includes facts, rules, concepts, and relationships. The inference engine is the knowledge processor which is implemented to work with available information on a given domain, coupled with the knowledge stored in the knowledge base, to draw conclusions or recommendations (Durkin, 1994). We use the term knowledge-based system interchangeably with expert system.

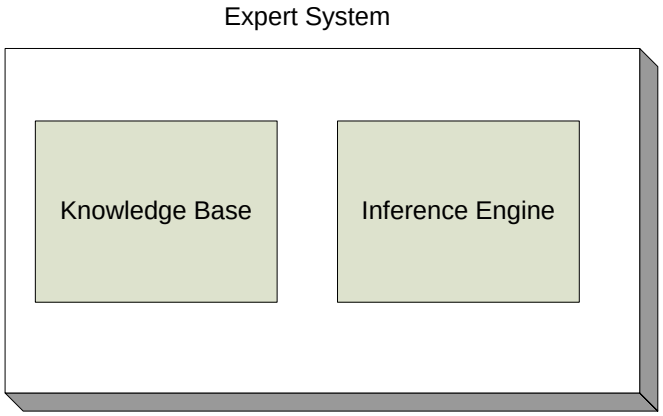


Figure 3.1: The architecture of an expert system.

3.1 Knowledge Base

The process of building a knowledge base has been long studied. The process typically involves two parties: a knowledge engineer and a domain expert. The knowledge engineer is

> "someone who investigates a particular domain, determines what concepts are important in that domain, and creates a formal representation of the objects and relations in the domain. (Russell and Norvig, 1995)"

The knowledge engineer usually interviews the real experts of the domain to learn about the domain and elicit the required knowledge to solve problems in the domain. This process is usually called **knowledge acquisition**.

However, there has been some disagreement over the terminology used in the field. Two terms, i.e., **knowledge engineering** and **knowledge acquisition** are used. Knowledge acquisition (KA) is viewed as having a wider scope in which the knowledge engineering is defined as one of KA's subtasks. For example, (Suryanto, 2005) follows the definition of Knowledge acquisition, by Buchanan et al.:

> "the transfer and transformation of potential problem-solving expertise from some knowledge source to a program" (Buchanan et al., 1983)

and the definition of Knowledge engineering, by Feigenbaum:

> "the process of reducing a large body of knowledge to a precise set of facts and rules"(Feigenbaum, 1984).

That is, knowledge engineering is part of knowledge acquisition. In his approach, knowledge acquisition includes two main sub-tasks, i.e., knowledge elicitation and knowledge engineering. Knowledge engineering covers four main activities, namely, analysis, development, maintenance and reuse.

Elsewhere, knowledge engineering is viewed as the whole process of building an expert system, while knowledge acquisition is defined as a task of knowledge engineering (Durkin, 1994; Puppe, 1993; Russell and Norvig, 1995). For example, Durkin defines the knowledge engineering as

> "the process of building an expert system" (Durkin, 1994)

and the whole process includes problem assessment, knowledge acquisition, system design, testing and evaluation, documentation, and maintenance. Furthermore, the two terms are also used interchangeably (Gaines and Shaw, 1993).

Even though differences in definition of terminology have not been clarified, we believe that they minimally affect insight into the process of building an expert system. In fact, the various views of implementing an expert system are all similar, i.e., the process starts from collecting information regarding the problem, through transforming relevant information into knowledge using some knowledge representation approaches, to evaluating and maintaining the knowledge inside the knowledge base.

In this thesis, we put more focus on the process of acquiring knowledge from human experts, which includes initial knowledge elicitation, but also includes knowledge maintenance.

3.2 Knowledge Elicitation

There are many sources to gather knowledge of the problem of interest. There may be books, reports, database records, or domain experts. Acquiring knowledge from a human expert is specifically termed **knowledge elicitation**. During the process,

not only knowledge is aimed to be captured, the problem solving strategies are also required.

Generally, the knowledge elicitation task is in a cycle of collecting knowledge and analysis to gain an understanding of the problem and insight into its solution. We can summarize the life cycle of knowledge elicitation as collection, interpretation, analysis and design.

Acquiring knowledge from human experts requires effective interpersonal communication skills and the ability to obtain the cooperation of the expert. The process usually starts with a wide scope, in the early stage, to obtain a basic understanding of the problem. During later sessions, the scope becomes narrower and more specific knowledge is collected. Durkin states that:

> "This iterative style of collecting information is like a funnel effect – moving from the general to the specific" (Durkin, 1994).

The next step, after information has been collected, is to review such information and identify key pieces of knowledge. As mentioned earlier the information collected is quite general in the early stage; the interest is to define the overall problem specification. In later stages, more knowledge will be uncovered.

Highly coupled with the interpretation is the analysis, where theories of the organization of the knowledge and the problem-solving strategies should be formed. This task also includes choosing a knowledge representation structure.

Following analysis, some understanding of the problem should be established, which should aid further investigation. Hence, the last task in the cycle is to form a new starting point for collecting additional knowledge. And the cycle starts once again.

It seems like there is no end point in the cycle of knowledge acquisition as stated by Durkin:

> Developing an expert system is somewhat like teaching a child some new subject. As the child obtains more knowledge about the subject, he or she understands it better and can use this understanding to solve

> problems. In a similar fashion, an expert system can continue to improve its performance by gaining more knowledge. (Durkin, 1994)

In theory the cycle can be ended when the expert system meets the requirement of the organization wishing to employ the expert system; however in the work here, knowledge acquisition and maintenance is on-going.

3.2.1 Knowledge Elicitation Methodologies

Many Knowledge Elicitation (KE) methods have been used to obtain the information required to solve problems. These methods can be classified in many ways. For example, KE methods can be classified into two main categories, i.e., human-mediated or machine-mediated approach (Suryanto, 2005). In human-mediated approaches, knowledge is elicited from a human expert through interaction with a knowledge engineer. The process may simply be an interview or possibly using some knowledge acquisition tools, such as card sorting, laddering or repertory grids (Cordingley, 1989; Suryanto, 2005). In machine-mediated approaches, knowledge can be elicited directly from a human expert using knowledge acquisition tools without any interaction or assistance from the knowledge engineer. Some of the same techniques such as repertory grids may be used directly by experts.

Mcgraw and Harbison-Briggs classify KA approaches into three categories: interviewing experts, learning by being told, or learning by observation (Mcgraw and Harbison-Briggs, 1989). The first category, interviewing experts requires no equipment, it is just a conversation between knowledge engineers and experts. In the second approach, learning by being told, the discussion or conversation is conducted through a user interface and the experts have to represent and refine their own knowledge from what they understand, while the knowledge engineer handles the design and helps the expert to understand what is required. In the last category, learning by observation, experts are asked to solve some sample problems or case studies. An induction algorithm is usually used to gain knowledge from what the expert has done and generates rules.

Among many techniques, there are three well known approaches, i.e., card sorting, repertory grid, and laddering, for which we will give a brief overview as they are related to the Ripple Down Rules methodology, used in this thesis. Card sorting is a comprehensive technique of knowledge elicitation and is considered to be an effective ways for elicitating the domain expert's idea about the knowledge structure (Wang et al., 2006). It is used to explore how a human expert groups concepts or terms related to the problem domain. Each concept is written on a card. The expert is then asked to sort the cards into piles such that cards in each pile have something in common. By naming each pile, the expert gives information on attributes and values which denote the properties of concepts.

Repertory grids are a well known technique originally based on the work of (Kelly, 1955) in psychology to determine an individual's view of the world without explicitly questioning an individual about structure per se (Corbridge et al., 1994). The process of taking three elements and asking for two of them to be paired in contrast with the third is the fundamental method of the technique. The repertory grid technique was brought into the area of knowledge acquisition by (Shaw and Gaines, 1988).

The notion of "laddering" was introduced into repertory grids by (Bradshaw, Boose, and Covington, 1987). Laddering techniques involve the creation, reviewing and modification of hierarchical knowledge, often in the form of ladders, i.e., tree diagrams. The expert is asked about a concept more generally and more specifically. For example, given a concept of "this cup in my hand", the expert might generalize it to "a drinking vessel", to "a small container", etc (Mycoted, 2006). A ladder is then constructed for the expert's view to the concept. The expert and knowledge engineer both refer to the ladder presented on paper or a computer screen, and add, delete, rename or re-classify nodes as appropriate. Laddering is considered a "contrived" (Corbridge et al., 1994) technique being used for three major purposes, i.e., to elicit sub-classes, explanation, goals and values (Rugg et al., 2002; Wang et al., 2006). Corbridge et al. (1994) state

> Laddering was developed as a method of clarifying the relations between the constructs which had been elicited by the grid (Hinkle, 1965) and, where possible, organising them into hierarchical relations. In

> the knowledge elicitation context laddering refers to a structured questioning strategy, using a limited number of probes, which is designed to elucidate the relation between concepts in the domain. This "domain orientated" laddering, as Major and Reichgelt (1990) point out, is designed to elicit "structural knowledge" (Clancey, 1983) and is essentially synonymous with "structural questioning" described by Wood and Ford (1991). Laddering may also be used as a method of task analysis to decompose a task into its component subtasks examining the procedural expertise involved. This "task orientated" laddering (Major & Reichgelt, 1990) is similar to "scripting questioning" as described by Wood and Ford (1991). (Corbridge et al., 1994)

3.2.2 Difficulties in Knowledge Elicitation

It is one of responsibilities of expert system developers to decompile expert knowledge into a form that can be studied and entered into the expert system. As stated by Durkin:

> You must be able to pick apart this intuitive knowledge, often called shallow knowledge, and uncover some of the deeper information. Shallow knowledge is formed from experience rather than from first principles, and is usually in the form of heuristics or rules of thumb.(Durkin, 1994)

Most developers have found knowledge elicitation is the most difficult part of the development of expert system. As stated by Duda and Shortliffe in 1983:

> The identification and encoding of knowledge is one of the most complex and arduous tasks encountered in the construction of an expert system... Thus the process of building a knowledge base has usually required a time-consuming collaboration between a domain expert and an AI researcher. While an experienced team can put together a small prototype in one or two man-months, the effort required to produce a system that is ready for serious evaluation (well before contemplation of actual use) is more often measured in man-years. (Duda and Shortliffe, 1983)

Hayes-Roth et al. used the term bottleneck to describe the difficulty in knowledge acquisition:

> Knowledge acquisition is a bottleneck in the construction of expert systems. The knowledge engineer's job is to act as a go-between to help build an expert system. Since the knowledge engineer has far less knowledge of the domain than the expert, however, communication problems impede the process of transferring expertise into the program. (Hayes-Roth, Waterman, and Lenat, 1983)

Difficulties in eliciting knowledge from the expert have long been recognized. Major difficulties are that: the expert may be unaware of knowledge used, the expert may be unable to verbalize the knowledge, and knowledge that the expert provides may be irrelevant, incomplete, incorrect or inconsistent.

Unaware of the Knowledge Used

This problem is recognized in cognitive psychology. In 1981, Dixon reported that humans are not mentally conscious of many of their activities, but perform them through repetition (Durkin, 1994). Durkin also suggests that humans appear to be unaware of their own mental processes when solving a problem (Durkin, 1994).

Unable to Verbalize Knowledge

Again, from a psychological point of view, the problem may be that humans may not be able to communicate their knowledge, not because they cannot express it, but because they are unaware of what knowledge they are using in the activity (Collins, 1985). Durkin notes problems even when experts appear to communicate their knowledge:

> Bainbridge (1986) reports that there is no necessary correlation between verbal reports and mental behavior. Humans will actually do things differently from the way they explain their performance. Chomsky (1957), from work on the theory of natural language grammar, drew attention to the distinction between "competence" and "performance." He concluded that individuals frequently utter sentences that would violate their own rules of grammar owing to performance factors.(Durkin, 1994)

There are also sub cognitive skills, such as riding a bike that are learned by trial and error after observing others perform the task. There are also many tasks where there is an apprenticeship period when the person is told what to do but they do not reach a high level of performance until they have a lot of experience. Flying a plane is a clear example.

Irrelevant, Incomplete, Incorrect, or Inconsistent Knowledge

Another problem is that the knowledge provided may be irrelevant, incomplete, incorrect, or inconsistent. It is highly likely for expert system developers to obtain irrelevant knowledge in the initial stage of knowledge elicitation, where they do not have much understanding of the problem domain and usually assume everything is relevant, and where the expert does not really appreciate what the knowledge engineer is looking for.

Knowledge obtained from experts can often be incomplete. Because of their unawareness of the knowledge they use, experts may unintentionally omit knowledge. This can lead to an incorrect performance of the expert system.

The problem is even worse if the knowledge provided is incorrect. Waterman states:

> "Don't believe everything experts say!" (Waterman, 1986).

This problem may occur if experts lack knowledge on some issues, or make mistakes during the elicitation process. But the most frustrating problem is when experts provide inconsistent knowledge. Inconsistent knowledge is often found when experts are trying to explain their problem-solving strategy. It is also found when experts describe the importance or the priority of some issues.

Selective Bias

Another problem is selective bias where humans tend to bias their approach to solving a problem toward their past experiences in solving similar problems, while they might overlook or ignore information that would lead to a different approach. As stated by Tversky and Kahneman:

> There are situations in which people assess the frequency of a class or the probability of an event by the ease with which instances or occurrences can be brought to mind. For example, one may assess the risk of heart attack among middle-aged people by recalling such occurrences among one's acquaintances. Similarly, one may evaluate the probability that a given business venture will fail by imagining various difficulties it could encounter. This judgmental heuristic is called availability. Availability is a useful clue for assessing frequency or probability, because instances of large classes are usually reached better and faster than instances of less frequent classes. However, availability is affected by factors other than frequency and probability. Consequently, the reliance on availability leads predictable biases,... (Tversky and Kahneman, 1974)

3.3 Ripple Down Rules

The technique called Ripple Down Rules or RDR was proposed by Compton and Jansen based on their experience developing the expert system GARVAN-ES1 (Compton et al., 1989). It is based on the idea that *"the knowledge the expert provides varies with the context and gets its validity from its ability to explain data and justify the expert's judgment in the context"* (Compton and Jansen, 1990). While not disagreeing with the various problem of dealing with experts, Compton and Jansen identify a meta-problem which subsumes the problems outlined above.

3.3.1 Ripple Down Rule and Cognitive Psychology

Traditional efforts to acquire knowledge from experts are based on the idea that knowledge is in the expert's head in some sort of model and hence focus on how to extract these models from the mind in some formalized representation. What if this is not true? Winograd and Flores argued that knowledge is just an individual interpretation within a shared background, and it is neither subjective, nor objective (Winograd and Flores, 1987). That is, there is no model or any representation in the brain. What the experts say is the interpretation of the situation.

Compton and Jansen argue that the knowledge from experts is to some extent

"made up" to justify expert's conclusions are right, rather than to explain the mental process of reaching the conclusions (Compton and Jansen, 1990). And the justification the expert creates will vary with the context and the concerns of the questioner that the expert believes they should address.

Clancey goes further stating that knowledge is not something in the mind to be extracted, but something created in communication and interaction with the environment. That is, other participants or listeners in the conversation directly affect how experts explain their knowledge, namely what is to be represented and what constitutes a representation (Clancey, 1993).

> An essential idea is that this process is perceptual and inherently dialectic. That is, the organization of mental processes producing coherent sequences of activity and the organization of representational forms (e.g., statements in a conversation, added lines to a sketch) arise together. A painter does not have a completed picture inside his head that he is merely executing on paper. A speaker in a conversation is not merely instantiating discourse plans and patterns. Even letters or words are not stored inside as descriptions of how they appear or how the hand or mouth moves. Mental organizations do not merely drive activity like stored programs, but are created in the course of the activity, always as new, living structures. (Clancey, 1991)

Hence, the expert's knowledge or justification is acquired or created for a particular context. RDR represents knowledge based on the current case being justified, that is, knowledge is represented by a rule in form of conjunctions of features which are selected from the current case. Not only knowledge is captured in RDR, the context is also maintained by *cornerstone cases*, which will be discussed later. Compton and Jansen state that it is not possible that a single rule can be created in a context free environment as the knowledge we communicate is a justification in some context (Compton et al., 1989).

3.3.2 Ripple Down Rule Essentials

This section starts with the discussion on how knowledge is represented in RDR in Section 3.3.2. Later on, the maintenance of context in which knowledge is acquired

is visited in Section 3.3.2. Another advantage of RDR is that the expert does not need to understand the organization of knowledge inside the knowledge base. This is discussed in Section 3.3.2.

Knowledge and Rule

In RDR, new knowledge is captured or created to deal with misclassification. For example, case X of class B is misclassified as class A. The expert is then asked to justify why the case is not classified as class A, but class B and identifies some features that distinguish the two cases. The justification will be encoded and stored in knowledge base. The original form of Ripple Down Rule KB consists of rules in the form of tree structure, where each node in the tree represents a rule. Each rule represents a concept or knowledge acquired in the form of conjunctions of features. The representation of a rule is discussed in Section 3.3.3.

Ideally, a single rule in knowledge base should be able to totally represent a concept in the problem domain. However, it is difficult, if not impossible, for concepts in real world to be represented with a single rule as the expert can rarely express such a rule (Suryanto, 2005).

Rules in the RDR framework can be classified into two main categories, i.e., new-concept rules or exception rules. Firstly, a new-concept rule is created for a context that has never been seen before. That is, there is no existing rule which can accommodate the current case, except for the default rule which is defined for any situation. On the other hand, an exception rule is created to refine an existing rule which the current case fires but the conclusion for the case is wrong. That is existing knowledge can be tuned or refined with future discoveries. A exception rule is sometimes referred to as a refinement rule. It has been stated that the exception structure can represent the domain knowledge compactly (Catlett, 1992; Compton, Preston, and Kang, 1995; Kang, Compton, and Preston, 1998; Kivinen, Mannila, and Ukkonen, 1993; Gaines and Compton, 1992; Scheffer, 1996; Siromoney and Siromoney, 1993; Suryanto, 2005; Suryanto, Richards, and Compton, 1999). Hence, any complex concept can be represented with a sub-tree where each rule belonging

to the tree may be refined by another sub-tree of new rules.

Cornerstone Case

A data base of *cornerstone cases* is maintained to provide the expert with insight into the context in which existing rules have been created. A cornerstone case is a case that was misclassified, hence a change in the system's knowledge was required. From the example above, a new rule is created to correctly classify case X to class B. X, as the context of this new rule, is stored in the data base as a cornerstone case.

Cornerstone cases ensure validity of the knowledge. When a new rule is added, the interpretations for all cornerstone cases that might fire the rule must be checked to see that the new rule does not corrupt the knowledge (Compton and Jansen, 1990). That is, the new rule should not change the classification for a cornerstone case. If it does, the expert is asked to select features to distinguish the cases, or to confirm that the conclusion for the cornerstone case should be changed. Not only the validity of knowledge can be maintained, checking the interpretations for all cornerstone cases helps ensure adding new knowledge is a incremental improvement (Compton and Jansen, 1990).

Cornerstone cases also assist the expert when adding a new rule. The expert can compare the differences between a current case and a cornerstone case for the last fired rule, and quickly select salient features for the new rule (Richards and Compton, 1998; Compton et al., 1993; Preston, Edwards, and Compton, 1994).

Organization

Knowledge inside an RDR KBS is automatically structured without any intervention from domain experts. This means experts do not need to know how knowledge is organized inside the KBS, how new knowledge is verified when it is added, or how consistency is maintained. Ripple Down Rules only require experts to provide their domain knowledge expertise. They are constrained in the feature selection task such that they can only select features that occur in the case they are consid-

ering.

Ripple Down Rule is an incremental KA approach. The system can start from an empty knowledge base and new knowledge can always be added to the knowledge base during the maintenance stage. Not only more rules can be added, the RDR KA can start with a minimal ontology and which can be added to whilst in use. For example, whenever needed, the expert can add new attributes to the system. This allows the ontology to grow incrementally. More importantly, users are able to use the system while the expert is maintaining it.

3.3.3 RDR Terminology

This section reviews those basic terms used in RDR paradigm: case, feature, attribute and rule.

Instance and Case

In machine learning, an instance is defined as

> A single object of the world from which a model will be learned, or on which a model will be used (e.g., for prediction). In most machine learning work, instances are described by feature vectors; some work uses more complex representations (e.g., containing relations between instances or between parts of instances) (Kohavi and Provost, 1998).

In RDR, the term "case" has been used instead of "instance" and has been defined as a sequence of $AttributeValue$(s) as shown below.

$$\begin{aligned} Case &\Rightarrow Classification: AttributeValueList \\ AttributeValueList &\Rightarrow \varepsilon \mid AttributeValue\ AttributeValueList \end{aligned}$$

$AttributeValue$ represents value of a particular attribute. Some attributes may be unknown or irrelevant (Suryanto, 2005).

However, the definition of a case is slightly modified in MCRDR as follows.

$$\begin{aligned} Case &\Rightarrow Classifications: AttributeValueList \\ Classifications &\Rightarrow \varepsilon \mid Classification\ Classifications \end{aligned}$$

Feature and Attribute

In machine learning, the terms *feature, attribute* and *variable* are often used interchangeably.

> Attribute (field, variable, feature) is a quantity describing an instance. An attribute has a domain defined by the attribute type, which denotes the values that can be taken by an attribute (Kohavi and Provost, 1998).

In (Kohavi and Provost, 1998), Kohavi et al. classify an attribute into two common domain types, i.e., categorical or continuous. A categorical attribute is an attribute which has discrete values. There are two sub-types, i.e., nominal and ordinal. Nominal type denotes that there is no ordering between the values, such as last names and colors. The type ordinal denotes that there is an ordering, such as in an attribute taking on the values low, medium, or high. A continuous (quantitative) attribute is an attribute whose value is a subset of real numbers, where there is a measurable difference between the possible values. Integers are usually treated as continuous in practical problems.

In RDR, similar to other rule frameworks, we differentiate feature and attribute as follows. While attribute represents a characteristic of an instance, feature captures a higher abstraction of an attribute with some constraints. In other words, a feature is a logical expression on an attribute; this means each feature returns a boolean value while inferencing. For example, A is a continuous attribute of a domain. Constraints defined on attribute A, such as $A < 20$ or $A > 50$, are features. That is, a feature in our framework denotes a specification of an attribute and its value.

Rule

In RDR, a rule is in a typical form of

if (Antecedent) then (Consequent).

,which can be formalized as follows.

$$\begin{aligned} Rule &\Rightarrow Consequent \leftarrow Antecedent \\ Consequent &\Rightarrow \varepsilon \mid AttributeValue \\ Antecedent &\Rightarrow \varepsilon \mid Feature \wedge Antecedent \end{aligned}$$

$AttributeValue$ represents a value of a particular attribute. $Antecedent$ is conjunctions of $Feature$(s). $Consequent$ is the classification or a predefined value of the class attribute.

For example, in a domain, there are three attributes, i.e., A, B and C. A is continuous. B and C are categorical; B = low, medium, high, C = classX, classY, classZ. C is also class attribute. Rule R1 is defined as

$$C = classX \leftarrow (A < 20) \wedge (B == high)$$

$A < 20$ and $B == high$ are $features$ of attribute A and B, respectively. The conjunction of those two features represents $Antecedent$. $C = class - X$ is $Consequent$.

3.3.4 RDR Structure

Ripple Down Rules were originally introduced to deal with the single classification problem (Compton and Jansen, 1988). For example, in a plant identification system, every plant belongs to a single species. Since variations of the framework have been proposed to solve other problems, the technique was later called Sin-

gle Classification Ripple Down Rule (SCRDR) because its inference engine aims at producing one classification as the output. The first variation investigated by Kang is Multiple Classification Ripple Down Rule (MCRDR), which aims at multiple classification problems (Kang, Compton, and Preston, 1995). For example, in a medical diagnosis system, a patient may have more than one disease.

As mentioned earlier that the original form of RDR is tree structure. So far, three main RDR structures have been investigated. Firstly, a binary tree is utilized to support a single classification task in SCRDR. Secondly, an alternative structure to the binary tree is decision list, which is also implemented for a single classification. The last structure is an n-ary tree, which is used in the MCRDR paradigm. There are also structures containing multiple trees. For example,NRDR has multiple SCRDR trees.

Binary Tree

SCRDR normally refers not just to single classification RDR but to a binary tree implementation. Each node representing a rule has two distinct types of edges, labeled with either *except* or *if-not*. An illustration of SCRDR-binary tree is shown in Figure 3.2.

The SCRDR tree can be formally represented as follows.

$$\begin{aligned} SCRDR &\Rightarrow < Rule, Except, IfNot > \\ Except &\Rightarrow \varepsilon \mid SCRDR \\ IfNot &\Rightarrow \varepsilon \mid SCRDR \end{aligned}$$

SCRDR is recursively built on the triple $< Rule, Except, IfNot >$. $Rule$ is the rule defined above. $Except$ is the exception rule (or tree) and is defined as the triple $< Rule, Except, IfNot >$. $IfNot$ is the if-not rule (or tree), which is also the triple $< Rule, Except, IfNot >$.

The inference mechanism works in a top-down manner, starting from the top-most rule. For example a data case is passed to the tree starting from the root. If

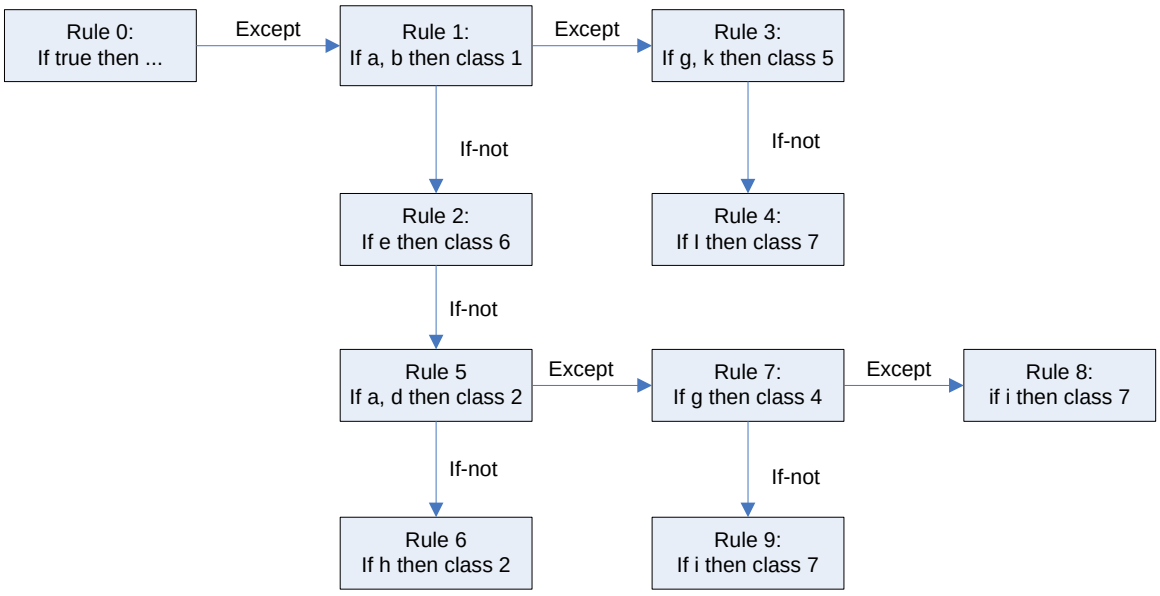


Figure 3.2: An example of SCRDR knowledge base in binary tree structure.

the data case satisfies the current node, the conclusion is temporarily recorded in a working memory and it is then passed to the next node in the *except* branch. If the case does not satisfy the condition of the rule, it is passed to the next node in the *if-not* branch. The process continues until there is no next node to evaluate. The last conclusion recorded in the working memory is the classification of the case.

For example, case {a, b, g}, is passed to the SCRDR KBS in Figure 3.2. Firstly, Rule 0 is fired; no conclusion is recorded in the working memory. The inference engine passes the case down the *except* branch and fires Rule 1. Here, "class 1" is recorded as a tentative classification. Again, the case is passed along the *except* branch to Rule 3, which is not satisfied. Hence the case is passed along the *if-not* branch to Rule 4. Rule is not fired, either. At this point, there is no further branch; the inference terminates and the classification is "class 1".

When the classification of the case is incorrect, the expert is required to identify features in the case which suggest a new conclusion. This new rule is automatically added to the rule giving the last conclusion as its exception. This means the case will be processed in exactly the same way, except that it will also be passed further to the new rule. There is also a mechanism to guarantee that the expert selects sufficient features in the case so that the new rule will not fire on a case for which the

Rule 0:	if (true)	then (null)	except (a,b), (e), (a,d), (h)
Rule 1:	if (a,b)	then (class1)	except (g,k), (i)
Rule 2:	if (e)	then (class6)	except
Rule 3:	if (g,k)	then (class5)	except
Rule 4:	if (i)	then (class7)	except
Rule 5:	if (a,d)	then (class2)	except (g), (i)
Rule 6:	if (h)	then (class2)	except
Rule 7:	if (g)	then (class4)	except (i)
Rule 8:	if (i)	then (class8)	except
Rule 9:	if (i)	then (class7)	except

Figure 3.3: An example of SCRDR in composite rules structure.

previous rule was correct. That is, the expert has to select features that distinguish the cases or confirm that the new conclusion should apply to the stored case.

Composite Rules

Composite Rules (CR)have recently been used to implement RDR KBs for single classification tasks. Composite Rules were first introduced in the IEMS system (Crawford, Kay, and McCreath, 2002a; Crawford, Kay, and McCreath, 2002b). Composite Rules use a clausal form to represent each rule, and are organized in a decision list where the first rule that explains a query is used (Cao and Compton, 2005). Each clause in the decision list is in an extended form of Rule in Section 3.3.3 as follows. *if (Antecedent) then (Consequent) [except (Exception$_1$, Exception$_2$, ..., Exception$_n$)]*

Each clause, when first created, is in the *Rule* form, consisting of *Antecedent* and *Consequent*. When more knowledge has been acquired, i.e., new rules are added, previous rules can be further refined with *Exception*(s). This will be described in more detail below. Figure 3.3 illustrates an example of Composite Rules.

The inference mechanism of Composite Rules works in a top-down manner, but with a slight difference from the binary tree structure, i.e., it searches for the first clause satisfying a data case. For example, a data case is sequentially passed down the decision list, starting from the topmost rule. The inference stops immediately when the case is satisfied. The conclusion of the clause is the classification of the case. To fire a rule, not only is *Antecedent* evaluated, a data case must not entail

any *Exception*.

For example, a data case a, b, g is passed to Composite Rules RDR in figure 3.3. Firstly, the default rule "true" is evaluated, but failed because of the case entail the exception (a,b). The case is then passed down the list to Rule 1. Rule 1 is fired as none of its exceptions hold for the case. The inference stops here and the case is concluded as "class1".

However, the clause can be viewed as in the typical *Rule* form, i.e., the condition part is the conjunction of *Antecedent* and the negation of *Exception*, as defined in formal representation below.

$$
\begin{aligned}
Clause &\Rightarrow Consequent \leftarrow Antecedent \wedge ExceptionList \\
Consequent &\Rightarrow \varepsilon \mid AttributeValue \\
Antecedent &\Rightarrow \varepsilon \mid Feature \wedge Antecedent \\
ExceptionList &\Rightarrow \varepsilon \mid Exception \wedge ExceptionList \\
Exception &\Rightarrow \varepsilon \mid \neg Antecedent
\end{aligned}
$$

The definition of *Consequent* and *Antecedent* are still as same as defined in Rule, in Section 3.3.3. *ExceptionList* is all the circumstances that prevent the clause from being entailed. *ExceptionList* is in a form of conjunctions of exceptions, which are defined as the negation of *Antecedent*.

Not only is the inference different, the KA mechanism is also different. When the classification of a data case is incorrect, instead of asking the expert immediately for a new conclusion, the inference is resumed at the next rule following the misclassifying rule. The process continues until either the case is correctly classified or the decision list is exhausted. Only when the decision list is exhausted, the expert is required to identify features in the case which suggest a new conclusion. A new rule is added to the bottom of the decision list and the negation of its condition (or *Exception* by the definition) is added to the *ExceptionList* of any previous rules that earlier gave a wrong conclusion. This means the case will be passed through to the new rule at the bottom of the decision list.

Cao and Compton have used simulation to evaluate the performance of these two structures. They have found that Composite Rules have a better convergence rate than SCRDR when the expert has a high level of expertise. However, if the expert is not doing well, i.e., when he tends to overgeneralize, the binary tree structure outperforms Composite Rules (Cao and Compton, 2005).

N-ary Tree

To handle multiple independent classifications of a case, MCRDR is built as an n-ary tree (Kang, Compton, and Preston, 1995; Richards and Compton, 1998), where each node representing a rule can have an arbitrary number of children and has a parent, except the root node which has no parent. False branches no longer exist in MCRDR trees. The MCRDR formal representation can be defined as

$$\begin{aligned} MCRDR &\Rightarrow < Rule, Children > \\ Children &\Rightarrow \varepsilon \mid < Child, Children > \\ Child &\Rightarrow \varepsilon \mid MCRDR \end{aligned}$$

Like SCRDR, MCRDR is recursively defined on $< Rule, Children >$. $Children$ is the list of all the children of the node. In RDR, they are all exception or refinement rules.

Figure 3.4 illustrates an example of MCRDR.

Each rule may be seen as a pathway against which data is evaluated (Richards and Compton, 1998). The inference evaluates all the rules in one level and proceeds to the next level of refinement for each rule that was satisfied. For example, a data case is evaluated against the root node, which is a default rule. The case is then passed to all the rules in level 1, which are refinement rules to the default rule. The inference stops when there are no more children to evaluate or when none of the rules can be satisfied by the case. Consequently, it ends up with multiple paths. In each path, the conclusion of the last refinement rule overrides other conclusions made along the path. That is each path represents a refinement sequence. Finally,

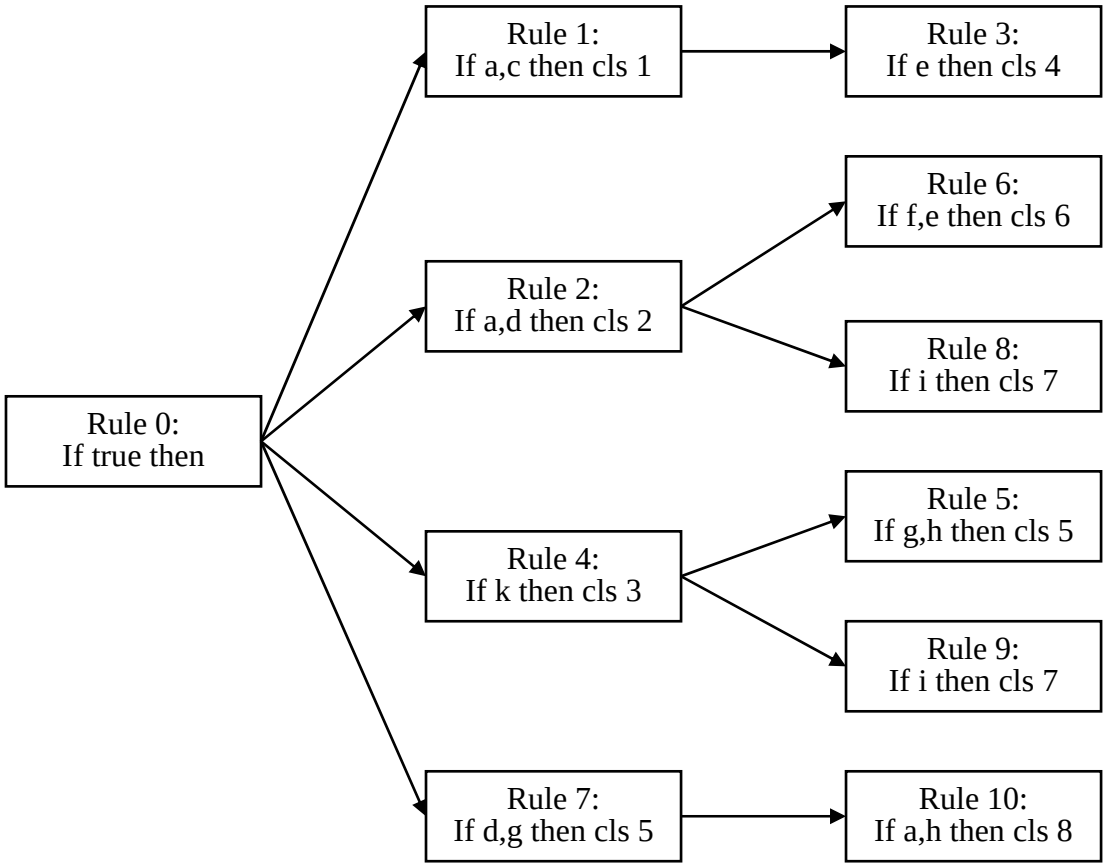


Figure 3.4: An example of MCRDR knowledge base.

the inference comes up with multiple classifications of a data case.

For example, case {a, b, d, d, e} is passed to the MCRDR KBS in Figure 3.4. Firstly, Rule 0 is fired. All its refinement rules are next evaluated. From the four rules, i.e., Rule 1, 2, 4 and 7, only Rule 1 and 2 are fired. The inference proceeds to the next level. From Rule 1, the next level is Rule 3. It is satisfied by the case and there is no further refinement rules. One conclusion being made for the case is "class4" by Rule 3. From Rule 2, its refinement rules are Rule 6 and 8. None of them fires. Hence, along this pathway, the case is classified "class2" by Rule 2.

Knowledge acquisition for MCRDR is different and more complicated than in SCRDR. KA is required when a case is classified incorrectly or a classification is missed. There are three basic steps in this process. Firstly, the expert provides the correct classifications. Then the system must find the location of new rules. Finally, new rules are created and added to the knowledge base.

Acquiring the correct classifications is simple, the expert only states them. By doing this, we end up with 1) correct classifications, 2)new classifications to be added and 3) incorrect classifications to be deleted. It is natural to simply leave correct classifications as they were. For new classifications, the system must decide where to add new rules. To find a location for a new rule, the system firstly decides if a new classification is the refinement of a wrong one or is independent from others. For example, if it is a refinement classification, the system attaches new rule to the rule which produces the wrong conclusion. Otherwise, the system adds a new rule at the top level of the tree. There can also be situations where a conclusion should not be given, rather than changed to another conclusion. In this case, the refinement rule is called a stopping rule and gives a null conclusion.

3.3.5 Construction RDR

Originally, RDR was investigated for classification tasks with two main variants, i.e., SCRDR and MCRDR. With attempts to address problems of other tasks, a variety of RDR have been proposed, based on the two predecessors.

For example, Nested RDR is an extension of original SCRDR which allows

hierarchical structuring of RDRs. It allows the re-use of definitions or ontology and the abstraction of concepts (Beydoun and Hoffmann, 1997; Beydoun and Hoffmann, 1998; Bekmann and Hoffmann, 2005).

Another variant of RDR is Repeat Inference Multiple Classification Ripple Down Rule or RIMCRDR, which allows the inference to be iterated until a task is completed (Compton et al., 1998; Compton and Richards, 1999; Compton and Richards, 2000; Suryanto, 2005). It is proposed to deal with configuration, assignment and planning problems RDR. While knowledge base and knowledge acquisition mechanisms are preserved as in typical MCRDR, the inference mechanism is conducted repeatedly until a task is completed. That is, for each inference pass, conclusions given will be added to working memory as new values and the inference is repeated until either a task is complete or no more new values can be found (Compton et al., 1998). At each stage, only non-conflicting conclusions are added to memory. When no more conclusions can be added the expert manually chooses between any remaining conflicting conclusions.

Both variants aim at adding intermediate concepts, supporting an expert in creating more complex reasoning and hierarchical concepts.

A generalization of RDR (Compton, Cao, and Kerr, 2004) has been proposed which seems to cover both RIMCRDR and NRDR. NRDR has to have specific heuristics to stop cycles while RIMCRDR uses a heuristic of only adding a conclusion where a single conclusion has been reached for that conclusion type. Generalized RDR fires rules in order of addition, except for refinement rules and conclusions cannot be removed during inference. That is where two conclusions might be given for a type, only the first is given. If this is wrong then the KB is refined. That is inference does not attempt to correct knowledge; this is only done by knowledge acquisition. This is simply the application of the decision list principle underlying SCRDR and composite rules, but applied to an environment where there are multiple interdependent conclusions.

3.3.6 RDR Applications

This section gives a review of RDR application in a number of tasks, such as configuration tasks, resource allocation tasks, heuristic search, help desk, text analysis, image processing, pharmacy, web monitoring, etc.

Classification Task

The main utilization of the RDR approach has been for classification tasks. Initial experiments were based on rebuilding GARVAN-ES1, an early medical expert system (Compton et al., 1989). The system was rebuilt as a SCRDR system and it demonstrated rapid rule addition of the order of 20 rules per hour with low error rates (Compton and Jansen, 1990; Preston, Edwards, and Compton, 1994).

There have been a number of pathology systems developed using a commercial RDR tool (Compton et al., 2006). The PEIRS system was an early example of a pathology interpretation system implemented on the single classification RDR approach. This system was largely developed whilst in routine use. After the initial domain model was constructed, the knowledge base was developed entirely by the expert. The only knowledge engineering skills required for the expert was to understand how to identify features for the rules, that is, identifying features in the case which distinguished it from the cornerstone case of the rule giving the wrong conclusion. This was how the wrong conclusion was corrected and the knowledge base grew. The knowledge base had evolved to about 2400 rules at the rate of about 3 minutes per rule and the system ended up being in routine use for four years and was claimed to cover about 25% of chemical pathology (Edwards et al., 1993). Commercial pathology systems are now available based on MCRDR where experts have constructed 10's of 1000's of rules (Compton et al., 2006).

Configuration Task

The RDR approach has also been applied to a configuration task. Puppe defined the task as selection, parameterizing and aggregation of basic objects to assemble

a solution which fulfills the requirements (Puppe, 1993).

A significant configuration task that has been attempted with the RDR is to develop an Ion Chromatography expert system (Mulholland et al., 1993; Compton et al., 1993). In the domain, there are eight components which may be varied giving many different combinations of equipment and reagents to try and optimise the system for different types of samples to be tested. Initially, eight separate RDR classification trees were built using Induct (Gaines, 1991) to decide on each of the eight components. A data case was applied to each of the eight trees. For each pass, a conclusion was added to working memory and the case run on the eight KBS again. The cycle stops when no new value can be added to working memory. In a later study with the RIMCRDR approach, there were eight conclusion types and at each inference stage conclusions were added if a single conclusion was given for that type (Compton et al., 1998). When required, rules could be added to give a correct conclusion or block incorrect conclusions. So only one conclusion was given for a conclusion type. It is claimed that the system performs satisfactorily on test data over 4000 cases of ion chromatography.

In summary, RDR has been demonstrated as a solution to configuration tasks. The same problem solving method as in classification tasks can be applied here. The only difference is that a configuration task requires multiple inference cycles. Each of which uses output from a previous cycle to fill in missing parts of the configuration. The cycles continue until a configuration solution is found.

Resource Allocation Task

Puppe defined the task as mapping objects onto other objects while considering preference, limited resources and other constraints (Puppe, 1993). Thus the main interest is the assignment of objects onto other objects. The problem is similar to configuration tasks, except for two additional constraints. Firstly, a resource has limited availability. Secondly, the order of object allocation matters.

Richards and Compton investigated a room allocation task SISYPHUS-I (Richards and Compton, 1999). In a typical room allocation task, two types of input cases

are considered; i.e., people cases and room cases. Here, Richards and Compton put more focus on people cases and their requirement for a room. A data case is passed to an MCRDR KBS in which a rule consists of features from both people cases and room cases. Upon each pass of the inference, a suggestion of suitable rooms is displayed with all other available rooms. The user is then required to pick a room for a person case. KA is required if the user disagrees with the room recommendation or no recommendation has been made.

In conclusion, the problem can be treated as a configuration task where people are objects with empty room slots. The typical MCRDR inference engine works well enough to map people to rooms. The only enhancement required is to track resources from a rule pathway.

Heuristic Search

RDR has also been used in heuristic search. Beydoun and Hoffmann developed heuristic searchers for chess by using NRDR to acquire human search knowledge (Beydoun and Hoffmann, 1997; Beydoun and Hoffmann, 1998). An NRDR KB was constructed as a hierarchy of concepts, i.e., each concept was an abstraction of other concepts which were eventually defined in domain dependent primitive terms. Each concept was built with an SCRDR tree. For any state of play, a number of SCRDR KB may fire rules in determining a move. If the determined move was not approved by experts, the KB was modified but the expert must decide which conclusion and therefore which particular KB should be modified. When required, new concepts could be added to represent the expert's explanation.

Another application investigated by Bekmann and Hoffmann was applying NRDR with genetic algorithm (GA) to detailed channel routing and switch box routing (Bekmann and Hoffmann, 2005). The GA uses a heuristic fitness function to guide their search in search space. The GA is stopped after each interaction and the expert writes rules to improve the fitness and mutation functions until they perform well enough .

Image Processing

RDR has been used for image processing tasks; in particular extracting lung boundaries from images. There are two different attempts in applying to this task. Firstly, Park et al. have applied MCRDR to classify and select region of lung from X-ray images (Park, Wilson, and Jin, 2000). After that, Misra et al. have extended MCRDR to extract lung boundaries in High Resolution Computed Tomography (HRCT) scans (Misra, Sowmya, and Compton, 2004). Unlike the work of Park et al., MCRDR here is extended to a new variant called *ProcessRDR*. The idea is that an image processing expert writes rules to control the edge detection, thinning and other algorithms which are assembled to give an overall image processing system.

3.3.7 RDR Problems and Discussion

It has been suggested that RDR will result in redundancy and repetition. The rule redundancy is when two similar rules classify the same case, or a rule is subsumed by another. This problem can lead to additional knowledge acquisition sessions. The problem results from the mechanism of adding a new rule to the knowledge base, where the newly created rule is verified against cornerstone cases or all previous cases, rather than being syntactically compared against all existing rules. Even though the problem appears to have no significant impact on KBS performance (Compton et al., 1991; Compton et al., 1998), Suryanto addressed the problem by reorganizing KBS and reduced the size of the KBS by 10.5%, while the accuracy was preserved (Suryanto, 2005). He states further that this figure only reflects general size reduction. The more important perspective is that the reduction in number of nodes, which might be later refined, is 27%. This figure suggests that future KA requires the expert to add 3 rules, rather than 4 rules.

Any knowledge based system has a problem when the world changes and some of its knowledge is no longer relevant. RDR is less prone to this problem because correction rules can still be added; however this results in unused knowledge. Yoshida et al. have suggested a minimum description length approach to dynamically removing unused knowledge in RDR knowledge bases (Yoshida et al., 2002;

Yoshida et al., 2004).

Another generic problem for all expert systems is brittleness. Expert systems are brittle because they do not realise the limits of their own knowledge. For example, an expert system with a very high level of knowledge about chemical pathology results might still diagnose a male as pregnant who has high levels of a pregnancy hormone, from a hormone-secreting tumor, because no one ever thought to tell it that only females get pregnant. The CYC project (Guha and Lenat, 1990) is an attempt at a solution to this problem by building a knowledge base of common sense, or general knowledge at the top the tree, as a foundation which other expert systems could be built on. Presumably, CYC would not allow the more specific KBS to conclude a male was pregnant. A variety of applications have used CYC knowledge base, for example, in directed marketing and database cleansing(Lenat, 1994).

Apart from the lack of commonsense knowledge, brittleness can also be characterised as a failure of the expert system to recognise when a case is outside its range of experience. Compton et al. proposed a mechanism called Prudence (Edwards et al., 1995; Compton et al., 1996) as a solution to the problem. The technique is based on the idea that expert systems should provide information to experts on their knowledge base's status, especially when it reaches its limit (Edwards et al., 1995; Compton et al., 1996). Hence,the RDR framework is extended with a warning policy which aims to warn users of new types of data cases, for which a new rule may have to be added. The warning mechanism makes use of profiles of attributes of seen cases, i.e., each rule independently maintains profiles of attributes of data cases which fire that rule. A warning is raised for a data case if a value of an attribute is different from the profile. In early experiments, Prudence worked reasonable well; i.e., it could correctly warn users for cases that needed further investigation but the false positive rate was quite high at about 15% on the Garvan data set (Compton et al., 1996).

This thesis focuses on incorporating the RDR framework in a dynamic problem domain such as computer network. The proposed methodology is based on an outlier detection algorithm, which will be discussed in the next chapter. The

method behaves in a similar fashion to prudence technique. The investigations have then been extended to the prudence application, with an aim to improve prudence performance; i.e., reducing the false positive rate of unnecessary warnings. These investigations are discussed later in Chapter 6.

Chapter 4

Detecting Outliers from Homogeneous Data

As mentioned earlier, this study aims at detecting anomalous behaviour in network traffic from discrepancies between observed traffic and profiles. Instead of learning a universal profile for general behavior, we propose to incrementally learn separate profiles for different behaviour. This means each profile is created for a particular situation, and can be added when a new situation is discovered.

The Ripple Down Rules framework, as discussed in previous chapter, can naturally partition a search space into smaller well defined sub-regions. This characteristic meets our requirement for partitioning the problem space of network traffic behaviour into regions of homogeneous traffic. The new partitions are created to cover discrete regions of the domain in which data is expected to be homogeneous, and if it is not, a new partition is created. We assume that these regions are sufficiently homogeneous that the data can be considered as uniformly distributed.

From these requirements, we need an algorithm that can start with no data and be fairly robust as it learns, because we allow a new partition to be added at any time, and it will contain only a single data item at the time it is added.

In the study, the proposed algorithm assumes that the data is uniformly distributed in some range; that all values in the range are equally probable. As will be discussed this assumption interacts with the requirement for the system to learn

on the fly. An outlier detection algorithm has been proposed for continuous attributes. Since the domain in which we are interested contains mainly continuous attributes. This will be explained in the next chapter. As a result, the algorithm is very simple and learns on-the-fly from observed data and flags outliers as they occur. It should be noted that the algorithm does not require a training session prior to its practical application to the domain; it can start from an empty state.

This chapter is organized as follows. Section 4.1 briefly discusses a background of outliers in statistics. Techniques for outlier detection in the literature are summarized in Section 4.2. The algorithm Outlier Estimation with Backward Adaptation (OEBA) is proposed in Section 4.3. An evaluation of OEBA is presented in Section 4.4. Although OEBA is intended for homogeneous data following a uniform distribution, it is also evaluated against data following a normal distribution, as shown in Section 4.5. The chapter is concluded in Section 4.6.

4.1 Outliers in Statistics

Statistical data analysis is widely used in a variety of domains to address decision problems under uncertainty, where probabilistic assessment of alternatives is made to support the process. The focus of early approaches was on analysis techniques, e.g., using various statistics, to solve the problem. No matter what techniques are considered, all of them are data-oriented. Provided all data in a set are valid, without errors, the analysis is likely to be accurate. However, this is not realistic. Some kinds of errors, e.g., measurement error or invalid data, are always present in the data set. We call them *outliers*.

Outliers are observations that appear to be inconsistent with the remainder of the collected data (Barnett and Lewis, 1987; Fallon and Spada, 1997; Iglewicz and Hoaglin, 1993). A standard definition of an outlier is from Hawkins:

> An outlier is an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism. (Hawkins, 1980)

The interpretation of the inconsistency covers both discordant observations and contaminants. A discordant observation is defined as an observation that appears surprising or discrepant to the investigator (Fallon and Spada, 1997; Iglewicz and Hoaglin, 1993). A contaminant is defined as an observation from a different distribution than the rest of the data (Fallon and Spada, 1997).

Possible sources of outliers are recording and measurement errors, incorrect distribution assumptions, unknown data structures, or novel phenomenon (Fallon and Spada, 1997; Iglewicz and Hoaglin, 1993). Recording and measurement errors are the most commonly suspected source of outliers. Secondly, data which does not fit well into the assumed distribution may fit well into a different distribution. Incorrect assumptions about the data distribution can lead to mislabeling data as outliers or accepting outliers as valid data, causing high rates of false positives and false negatives, respectively. Unknown data structures and correlations can cause apparent outliers. A data set could be made up of subsets, which are subject to different mechanisms and should be analyzed independently of each other. A data set indicative of a novel phenomenon or rare event can be often labeled as an outlier. For example, the network measurements during a conference or other activity, where there are extra users, indicating higher bandwidth consumption than usual can be thought to be outliers. This results in false alarms being raised, and if many false alarms are raised, genuine alarms may be overlooked.

To handle data outliers, one needs good record keeping for the observations and robust statistical methods for the analysis. To study the domain, all data should be maintained with any possible explanation or additional information. In analysis, the first step is to detect and label suspected outliers for further study.

Earlier works in outlier handling procedures can be distinguished as two main types, i.e., accommodation of outliers and discordancy testing. The dichotomy of approaches relies on differences in aims. When statistical methods are used to draw valid inferences about the sample population, so that the presence of outliers will not seriously distort those models, we classify these approaches as the accommodation of outliers. Conceptually, the methods used must be robust to small deviations from the assumptions (Barnett and Lewis, 1987; Huber, 1981).

On the other hand, any discordancy test determines whether an outlier should be retained or rejected.

In both approaches, we need to form two hypotheses, i.e., null hypothesis and alternative hypothesis. A null hypothesis explains the pure data set without contamination. An alternative hypothesis, on the other hand, accommodates outliers.

4.1.1 Accommodation of Outliers

In the accommodation of outlier approach, one needs to estimate the location and dispersion of the sample before any hypothesis is tested. The approach must make an assumption about the distribution where data come from, for example, a normal distribution. It is possible that data are actually from some other distributions. The approach is still considered robust if the actual distribution is not too dissimilar from the assumed one, but this does not guarantee that it will provide good robustness against outliers from contamination. It is necessary to accommodate outliers in an alternative model.

4.1.2 Discordancy Test

In the discordancy test approach, a distribution assumption is first made on the data set. Some statistical tests are performed to examine *whether a sample X_i is not only an extreme, but is also statistically unreasonable even when viewed as an extreme* (Barnett and Lewis, 1987). If it is, we say that X_i is a discordant outlier. An alternative model, where the presence of possible outliers is accounted for, is also constructed to assess the properties of the test.

4.1.3 Discussion

The choice of approach is interest-oriented. If the aim is to define the characteristics of the data set by a basic model (accommodation) then any test of discordancy is irrelevant. On the other hand, a test of discordancy is for situations where we wish to reject outliers as manifestations of contamination. However, no matter

what aim we set, the approach usually involves statistical tests on hypotheses of assumed distributions of sample data without outliers and with outlier contamination. The common distributions mostly assumed on sample data are the normal and exponential distributions (Barnett and Lewis, 1987).

In contrast, our study is intended for data following a uniform distribution because we apply the Ripple Down Rules framework to segment a problem space into smaller well defined sub-regions of homogeneous data which should follow a uniform distribution. This will be discussed later in Section 4.3. We chose a uniform distribution because we believe that system administrators implicitly use a uniform distribution. That is, they apply an implicit expected range to the data they observe. They do not care where data falls inside that range. If we were trying to determine the central tendency of network traffic, it might be more appropriate to use another distribution but our interest is rather in the range. However, we also provide simulation results on using our method with normally distributed data.

4.2 Outlier Detection

Outlier detection has been long studied in statistics. It was started as a by-product of the attempt to explain a data set. However, there was also a need to filter some few data points that behave differently, according to specific criteria, from the majority of the set. This introduced the study of outlier detection. The first and foremost technique used in the area is distribution-based, where an assumption on the underlying data set must be made to detect outliers. In addition to these purely distribution-based approaches, more complex alternatives have been proposed recently, including depth-based, distance-based, and density-based approaches. These approaches introduce new critical parameters to justify the likelihood of data being outliers, in contrast to early distribution approaches where the justification is done directly on data values. These techniques are now reviewed.

4.2.1 Distribution-Based Approach

With an aim to label suspected outliers for further study, some statistical tests are performed on suspected outliers or extreme values. Early detection approaches relied on a distribution assumption. The normal distribution is the most commonly assumed distribution amongst others. Using a normal distribution, one can derive a z-score or modified z-score to calculate the probability that a value is an outlier (Iglewicz and Hoaglin, 1993; Barnett and Lewis, 1987). Another simple method which does not depend on a distribution assumption is the boxplot (Tukey, 1977). All of the experimental observations are standardized and the standardized values outside a predetermined bound are labeled as outliers (Rousseeuw and Leroy, 1987; Fallon and Spada, 1997).

As mentioned above a combination of models might be assumed for the contamination, e.g., Yamanishi et al. used a gaussian mixture model and histogram density model to represent continuous and categorical variables, respectively (Yamanishi, Takeuchi, and Williams, 2000; Yamanishi and Takeuchi, 2001), They also introduced two discounting algorithms to update the two models and measure how much the models have changed after learning. A high score indicates a high possibility that the datum is an outlier.

The major problem is that a large amount of data is required to accordingly specify a distribution and this data should not contain outliers, but should not exclude values that are unusual but are not outliers. Most techniques are also univariate.

4.2.2 Depth-Based Approach

To address issues with distribution-based techniques, depth-based approaches have been proposed. The depth-based approach is based on the geometric characteristics rather than the distribution of the data set. Data are represented as points in a k-d space and organized in layers, with the expectation that shallow layers are more likely to contains outliers than deep layers. These depth-based methods can overcome the problem of distribution fitting, and conceptually can process data in

a multidimensional space.

However,in practice, there is a computational problem in the approach. To compute k-dimensional layers, the technique relies on the computation of k-dimensional convex hulls, which has a lower bound complexity of $\Omega(N^{\lceil \frac{k}{2} \rceil})$. Hence, depth-based methods are not expected to be practical for more than 4 dimensions for large data sets (Knorr and Ng, 1998). In fact, existing depth-based methods only give acceptable performance for $k \leq 2$ (Ruts and Rousseeuw, 1996).

4.2.3 Distance-Based Approach

The distance-based approach mainly aims at developing methods to detect outliers in an accumulated data set, rather than developing methods able to learn a model to predict outliers in new incoming data (Angiulli, Basta, and Pizzuti, 2006). It was introduced by Knorr and Ng (Knorr and Ng, 1997; Knorr and Ng, 1998) to address a problem in multidimensional data sets They define the notion of outliers as follows. *An object O in a data set T is a distance-based outlier if at least fraction p of the objects in T is further than distance D from O* (Knorr and Ng, 1997; Knorr and Ng, 1998).

The technique can handle k-dimensional data, for any value of k with complexity of $O(kN^2)$. An optimized variant of the technique is also proposed with complexity that is linear wrt N, but exponential wrt k. However, this optimized variant works best for $k \leq 4$.

4.2.4 Density-Based Approach

Both depth- and distance-based approaches suffer from the problem of local density in data sets. The density-based approach is an alternative to those techniques. It focuses on the local neighborhood density and is interested in addressing the problem of arbitrary regions in a data set. Instead of a binary decision, i.e., being an outlier or not, the paradigm introduces an outlier factor for each object. The higher the outlier degree, the more likelihood the object is an outlier. The outlier

factor is defined and determined differently in different techniques.

For example, Breunig, et al. (Breunig et al., 2000; Breunig et al., 1999) defined the notion of local outliers. Every object is assigned with a Local Outlier Factor(LOF), which is the outlier degree of an object that takes into account the clustering structure in a bounded neighborhood of the object. The neighborhood is defined by the distance to the k-th nearest neighbor. The technique performs very well with complexity of $O(N^2)$. However, issues have been raised with respect to the selection of k, e.g., k must be as large as the size of clusters, and the method exhibits some sensitivity on the choice of k (Papadimitriou et al., 2002).

Another density-based method is proposed by Papadimitriou, et al. (Papadimitriou et al., 2002). In their work, a Multi-granularity Deviation Factor(MDEF) is defined as the relative deviation of the local neighborhood density of a point P from the average local neighborhood density. They also incorporate probabilistic reasoning on decision-making while flagging outliers. An alternative to speed up the computation has also been suggested (Papadimitriou et al., 2002).

4.2.5 Problems and Discussion

There are two main measures of the efficiency of outlier detection; that is, false positive and false negative rates. The false positive rate is the proportion of valid instances that were erroneously reported as being invalid or outliers. The false negative rate is the proportion of invalid instances or outliers that were erroneously reported as valid. We would like to keep these two measure to a minimum as much as possible.

Another concern is computational complexity. Most of the time, the data sets we are dealing with are high dimensional. Unfortunately, most outlier detection techniques are not designed to cope with such data, except the density-based approach, which is able to address the problem of high-dimensional, large data sets with arbitrarily different regions. The analysis of high dimensional data typically requires high computational complexity and is too slow. Techniques in dimensionality reduction are usually applied to the data set prior to the actual analysis.

It should be noted that the trend in outlier detection has recently focused on the identification of outliers, rather than modeling or clustering the majority of data. The reason being is that constructing the data model is a highly computational task, but, for some applications or problem domains, the model may not be their goal.

The ultimate aim of this study is a solution to network intrusion detection. As we mentioned earlier in the previous chapter, we are implementing a traffic volume anomaly detection system, which essentially requires network traffic profiles in the analysis. The aim of our investigations in outlier detection is an online and low cost algorithm to detect outliers while the system is learning data behavior. Techniques in depth-based, distance-based or density-based approach are not suitable for our domain of interest, because they require all data to be present before analysis. In the domain, data comes in as a time series and is not complete.

However, our technique relies on a different assumption from other distribution-based techniques. Instead of taking the whole data set into account and constructing a baseline distribution, we partition data set into regions of homogeneous data and detect outliers in these regions. The technique is investigated in the next section.

4.3 Detecting Outliers while Learning

To detect an outlier, it is necessary for any algorithm to have some sort of model of expected behaviour to assess data against. Normally, a model is constructed upon a particular data attribute. We can classify an attribute into 2 main types, i.e., categorical or continuous, each of which requires a different modeling technique. However, in the network traffic domain, most attributes are real numbers. Hence, the focus of investigation is on modeling a continuous attribute.

We propose a probability-based algorithm to model a continuous attribute in a dynamic domain. The algorithm not only maintains a range of valid data, which have been observed, but adapts the model to changes in the domain as well as

flagging outliers when they are observed. We have made three assumptions in this algorithm:

1. All attributes are independent.

2. Provided there is a proper segmentation of the domain, data should behave similarly in each region, forming a cluster of homogeneous data.

3. For each region, homogeneous data follow a uniform distribution on an interval [a, b]. That is,

$$\begin{aligned} f(x) &= 0 && \text{for } x < a \\ &= \frac{1}{b-a} && \text{for } a \leq x \leq b \\ &= 0 && \text{for } x > b \end{aligned}$$

and

$$\begin{aligned} D(x) &= 0 && \text{for } x < a \\ &= \frac{x-a}{b-a} && \text{for } a \leq x \leq b \\ &= 1 && \text{for } x > b \end{aligned}$$

where $f(x)$ is the probability density function and $D(x)$ is the cumulative distribution function.

From the above assumptions, the probability of any value would fall into a range $[a', b']$ inside the interval $[a, b]$ is

$$\begin{aligned} P(a' \leq x \leq b') &= D(b') - D(a') \\ &= \frac{b'-a}{b-a} - \frac{a'-a}{b-a} \\ &= \frac{b'-a'}{b-a} \end{aligned}$$

where $a' \geq a$ and $b' \leq b$.

That is with a uniform distribution from 0 to 1, 50% of the data will fall between 0 and 0.5, 10% between 0 and 0.1. The function $P(a' \leq x \leq b')$ is the probability of the range [a', b'] for a single datum. However, we are interested in an event

where n independent data all fall into a particular region inside an interval, which is $P(a' \leq x_1 \leq b' \text{ and } a' \leq x_2 \leq b' \text{ and } ... \text{ and } a' \leq x_n \leq b')$. For convenience, we refer to this probability as *the Range Probability for n objects* (RP^n). From the probability of a sequence of independent events, this probability is determined by the product of the probability of individual events. Hence, the probability of n consecutive data points all lying in the region [a', b'] inside the interval [a, b] is

$$\begin{aligned} RP^n &= P(a' \leq x_1 \leq b') \times P(a' \leq x_2 \leq b') \times ... \times P(a' \leq x_n \leq b') \\ &= \left(\frac{b'-a'}{b-a}\right)^n. \end{aligned}$$

Let us assume that, from n observations, a is the minimum and b is the maximum. The interval [a, b] is the provisional range after n observations. The true boundary of the space might not be seen yet. Any object x that falls outside this interval should not simply be considered as an outlier, as possibly it is a new and unseen object that is part of the population. To justify the possibility of x being part of the group, we construct a temporary scenario that the outlier x actually belongs to the group. Hence, the interval [a, b] should be updated to either [a, x] if x is greater than b or [x, b] if x is less than a. This means the probability of the previous n objects falling in what is now the sub-region [a, b] is reduced from 1 to either $\left(\frac{b-a}{x-a}\right)^n$ or $\left(\frac{b-a}{b-x}\right)^n$, depending on whether x is greater than b or less than a, respectively.

Algorithm 1 Outlier Estimation

a: the minimum of the range after n observations
b: the maximum of the range after n observations
x: new observation
T: the least confidence level at which x is accepted as a value of the population
if $x \geq a$ and $x \leq b$ **then**
  n=n+1
else if $x > b$ and $RP^n \geq T$ **then**
  b=x, n=n+1
else if $x < a$ and $RP^n \geq T$ **then**
  a=x, n=n+1
else
  REPORT outlier
end if

We then assess whether x is an outlier against the RP^n. The less the RP^n, the higher the likelihood that x is an outlier. An x is accepted into the group only if RP^n of region [a, b] is greater than T, where T is an arbitrary threshold, which needs to be determined. The algorithm as shown in Algorithm 1 is named the *Outlier Estimation* (OE) algorithm.

A new model for each attribute is created every time a new region is partitioned. It is possible that an outlier is added to the model, especially when n is small in a new partition. Once an outlier is mistakenly accepted into the model, the range is updated. For example, when an outlier is above the range, the maximum is updated or if it is below the range, the minimum is updated.

When more data have been observed, any outliers that were misleadingly added to the model are likely to exhibit themselves as extremes in the model, and can be be rejected from the model. This means there should be a mechanism to inspect or assess the feasibility of the model after an update. This mechanism is named Backward Adaptation (BA). BA, as shown in Algorithm 2, works as follows. As soon as a range is updated, BA starts to assess the new range. If all observations following the update fall within a region and its RP^n is less than the threshold T, the model is contracted to that region.

Algorithm 2 Backward Adaptation

a: the minimum of the range after n observations
b: the maximum of the range after n observations
x: new observation
T: the least confidence level at which X is accepted as a value of the population
a': min(x, minimum observed since a was set)
b': max(x, maximum observed since b was set)
na: the number of population within the range after a was set
nb: the number of population within the range after b was set
if $RP^n(a, b') < T$ **then**
  $b = b'$, $nb = 0$
end if
if $RP^n(a', b) < T$ **then**
  $a = a'$, $na = 0$
end if

To simplify the following discussion we will consider only the maximum. In the OE algorithm, n is always increasing as all previous points are below the new

maximum. However, with the BA algorithm, n is reinitialized whenever a new maximum is set, as we wish to assess the probability that the sequence of data occurring after the new maximum is set all fall within a subrange. We count the data points since the last maximum and also note the actual maximum observed. After n points, and an observed maximum of $b' < b$ for those n points, we can repeat the same calculation that these n points all fell within the subrange. If the probability is less than a threshold T, we reduce the maximum to the observed maximum, and start again.

In summary, if an outlier accepted into the model is an upper bound, then if this outlier is transient, one would expect the sequence of observations following the outlier to all fall within a sub-range. Once enough data have been collected to identify the previous bound as an outlier, the range is changed.

4.4 Simulations

The OE and BA algorithms rely on one parameter, the threshold T. Here, we conducted simulations to test the performance of the proposed OE and BA algorithms and to determine the optimal threshold T.

The metrics for the performance were the estimated minimum, estimated maximum, false positive rate and false negative rate. The false positive rate is the proportion of valid instances that were erroneously reported as being invalid or outliers and is computed as

$$\text{false positive rate} = \frac{\text{number of valid instances identified as outliers}}{\text{number of valid instances}} \tag{4.1}$$

The false negative rate is the proportion of invalid instances or outliers that were erroneously reported as valid. It is computed as

$$\text{false negative rate} = \frac{\text{number of invalid instances not detected as outliers}}{\text{number of invalid instances}} \tag{4.2}$$

Simulations were set up for two different scenarios. In the first scenario, data

sets were noise free. This was to test the convergence capability of the algorithm to the true boundary of the data. In this phase, we focused on whether there would be false positives detected.

Secondly, data sets were contaminated with outliers. As this was to test the outlier detection capability of the algorithm in various situation of outlier contamination, we were more interested in the produced false negative rate.

4.4.1 Simulations on Noise Free Data Sets

The simulation started from the data set generation. We chose to generate, for a data set, 3,000 random numbers following the standard uniform distribution in the range of 0.0 to 1.0 (U(0,1)). To reduce bias that might occur from the generated data set, we generated 10 data sets for this simulation. We show average data in the results.

Data Set Generator

Simulations were implemented with the Java programming language. An instance of class java.util.Random is used to generate random numbers. The class uses a 48-bit seed, which is modified using a linear congruential formula. The method nextDouble(), which returns the next pseudorandom, uniformly distributed double value between 0.0 and 1.0, is used.

Experimental Results

The uniform distribution generator created 10 sets of 3000 random numbers. These series were separately fed into the modeling engine from a blind state, where it knew nothing about the data set. Statistics, e.g., false positive, false negative, expected minimum, expected maximum, were collected for each of 10 series and averaged. The results are shown in Table 4.1.

The simulation showed that the algorithm is able to converge from a blind state toward the true interval [0, 1) as shown in Table 4.1. When the threshold

T	Minimum	Maximum	FPR
$1.0E-99 \leq T \leq 1.0E-2$	0.0003	0.9997	0
$T = 1.0E-1$	0.0232	0.9768	0.1083

Table 4.1: Simulation results of the OEBA algorithm on noise free data. Metrics shown are final estimated minimum, maximum and false positive rate (FPR).

T is between 1.0E-99 and 1.0E-2, the algorithm can converge very close to the true limit. There is no evidence of false positives, that is inappropriate warnings for outliers in this range of T. However, when T is greater than 1.0E-2, the gap started to grow and false positives were introduced. That is, the algorithm is likely to refuse to include normal instances into the model and flag them as outliers, causing false positives. We do not show further details of false positives because the algorithm produced none, except for those with $T > 1.0E-2$. And in our later experiments, we used values that had 0 false positives in these simulations.

4.4.2 Simulations on Outlier Contaminated Data Set

In this simulation, data sets were still generated following a uniform distribution in the range of 0.0 to 1.0 (U(0,1)). However, some outliers were randomly injected into each data set. Outliers were generated in bands both above and below the valid range of [0.0, 1.0]; that is, > 1.0 and < 0.0. For each band, its range is 0.5 wide.

There were two parameters controlling the behaviour of outliers, i.e., level of outlier contamination in the data set and distance of the outlier band from valid data.

Contamination Level

The contamination level denotes the quantity of outliers in the data set. The number of outliers is approximate because we also use a random number generator to decide when to produce an outlier. We use a percentage to represent a level, for example, a contamination level of 1% means there are approximately 30 outliers in a data set of 3,000 points. Five different levels, i.e., 1%, 5%, 10%, 15% and 20%

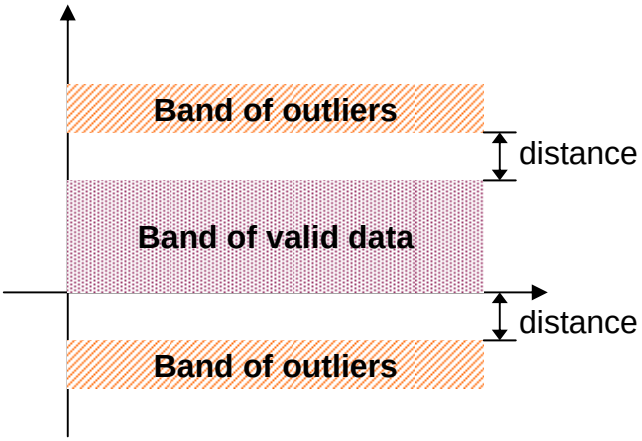


Figure 4.1: Distance between bands of outliers and valid data.

are used in this simuation.

Outlier Distance

The second parameter is outlier distance, which denotes the minimum distance or gap between the band of outliers and valid data, as shown in Figure 4.1. It controls the distance for both upper and lower bands. For example, the valid range of a data set is 0.0 and 1.0. The upper band of outliers with a distance of 0.1 is in the range of 1.1 and 1.6, where the lower band is in the range of -0.6 and -0.1. In this simulation, outlier distance was varied with 5 following values; 0.01, 0.05, 0.1, 0.5 and 1.0.

Data Set Generator

Data sets are generated in three bands as shown in Figure 4.1. We use an instance of the class java.util.Random to decide when to inject outliers into a data set, according to a predefined contamination level (*Level*) as shown in Figure 4.2. That is, if a generated value is less than *Level*, an outlier is injected at that point. Otherwise, a valid datum is generated.

To generate random numbers for a band of valid data, an instance of the class java.util.Random is used, as in the previous section. Outliers in two bands are generated by another instance of the class java.util.Random. Note the method

nextDouble() always returns a value between 0.0 and 1.0. Random numbers less than 0.5 are offset to the lower band, otherwise they are offset to the upper band. The offsetting is controlled by a specified *distance*, that is, a generated number $x < 0.5$ is offset to $x - 0.5 - distance$, and if $x \geq 0.5$, it is offset to $x + 0.5 + distance$. For example, given a random number for an outlier is 0.8 and *distance* is set to 0.05. This outlier is offset to 0.8 + 0.5 + 0.05 = 1.35.

```
public double[] generateUniform(
    int N,              // the number of outliers
    double Level,       // the contamination level
    double Distance     // the outlier distance   ) {

    Random valid = new Random();
    Random selector = new Random();
    Random outlier = new Random();
    double[] array = new double[number];

    for (int i=0; i<N; i++){
        if (selector.nextDouble() < Level) {
            double anOutlier = outlier.nextDouble();
            if (anOutlier < 0.5) {
                // Lower band
                array[i] = anOutlier - 0.5 - Distance;
            }
            else {
                // Upper band
                array[i] = anOutlier + 0.5 + Distance;
            }
        }
        else {
            array[i] = valid.nextDouble();
        }
    }
    return array;
}
```

Figure 4.2: Java code of data set generator with outlier contamination.

Experimental Results

Our interest for this simulation is on false positive and false negative rates; that is, how many valid data are flagged as outliers and how many real outliers are missed, respectively. With five different levels of contamination levels and five distances, we have 25 different data sets. For each set, a threshold T between 1.0E-99 and 1.0E-2 is used. A threshold of $\leq 1.0e-4$ eliminates all FP, so for simplicity, only results of thresholds between 1.E-5 and 1.E-2 are shown in table 4.2. We also group FPRs for data sets with the same *Distance* in the same table.

Simulations show interesting results for false negative rates (FNR). For convenience, we draw FNRs for data sets with the same *Distance* value in the same diagram. Results are illustrated graphically in Figure 4.3, 4.4, 4.5, 4.6, 4.7.

When threshold T is set too small, no outliers are flagged, e.g., in data set where $Distance = 0.01$ FNRs is 1.0 for all contamination levels, except for $Level = 0.01$. This happens with all other *Distance*s. However, when the threshold T is large enough, some outliers are correctly detected making FNRs drop. At some point, all FNRs reach zero. These results imply that the threshold T must be tuned to outlier characteristics to keep FNR at zero.

Table 4.3 summarizes, from the above simulations, the threshold T which produces zero FNR in different scenarios. Data shown in the table are the minimum thresholds T can be configured in each data set to have zero FNR. Hyphen "-" in the table means there is no possible value for threshold T not to produce false negative instances.

In general, the algorithm performs perfectly, except where outliers are very close to the valid data, i.e., 0.01. When the distance is 0.05 or higher, the algorithms can always detect all outliers no matter how many outliers are in the data set; i.e., up to 20% contamination level. We have not tested above 20% outliers, as what these tests concern is a constant background level of 20% outliers, i.e., can outliers be detected when there are always 20% outliers. We consider this an unrealistic situation. Of course there may be a burst of outliers much higher than 20%, but these will be detected when they commence as a transition from a previous range.

	Outlier Contamination Level				
T	1%	5%	10%	15%	20%
1.E-05	0	0	0	0	0
1.E-04	0	0	0	0	0
1.E-03	0.0001	0	0	0	0
1.E-02	0.0181	0.0006	0.0001	0.0002	0.0011

(a) Distance = 0.01

	Outlier Contamination Level				
T	1%	5%	10%	15%	20%
1.E-05	0	0	0	0	0
1.E-04	0	0	0	0	0
1.E-03	0.0001	0	0	0.0002	0
1.E-02	0.0331	0.0001	0.0004	0.0006	0.0001

(b) Distance = 0.05

	Outlier Contamination Level				
T	1%	5%	10%	15%	20%
1.E-05	0	0	0	0	0
1.E-04	0	0	0	0	0
1.E-03	0	0.00003	0	0	0.0001
1.E-02	0.0001	0.0013	0.0006	0	0.0002

(c) Distance = 0.10

	Outlier Contamination Level				
T	1%	5%	10%	15%	20%
1.E-05	0	0	0	0	0
1.E-04	0	0	0	0	0
1.E-03	0	0	0.00003	0	0
1.E-02	0.0003	0.0004	0.0001	0.0012	0.0005

(d) Distance = 0.50

	Outlier Contamination Level				
T	1%	5%	10%	15%	20%
1.E-05	0	0	0	0	0
1.E-04	0	0	0	0	0
1.E-03	0	0	0	0	0
1.E-02	0.0005	0.00003	0	0.0003	0

(e) Distance = 1.00

Table 4.2: False positive rates for uniformly distributed data. Each table shows false positive rates at different outlier contamination levels at specified distances.

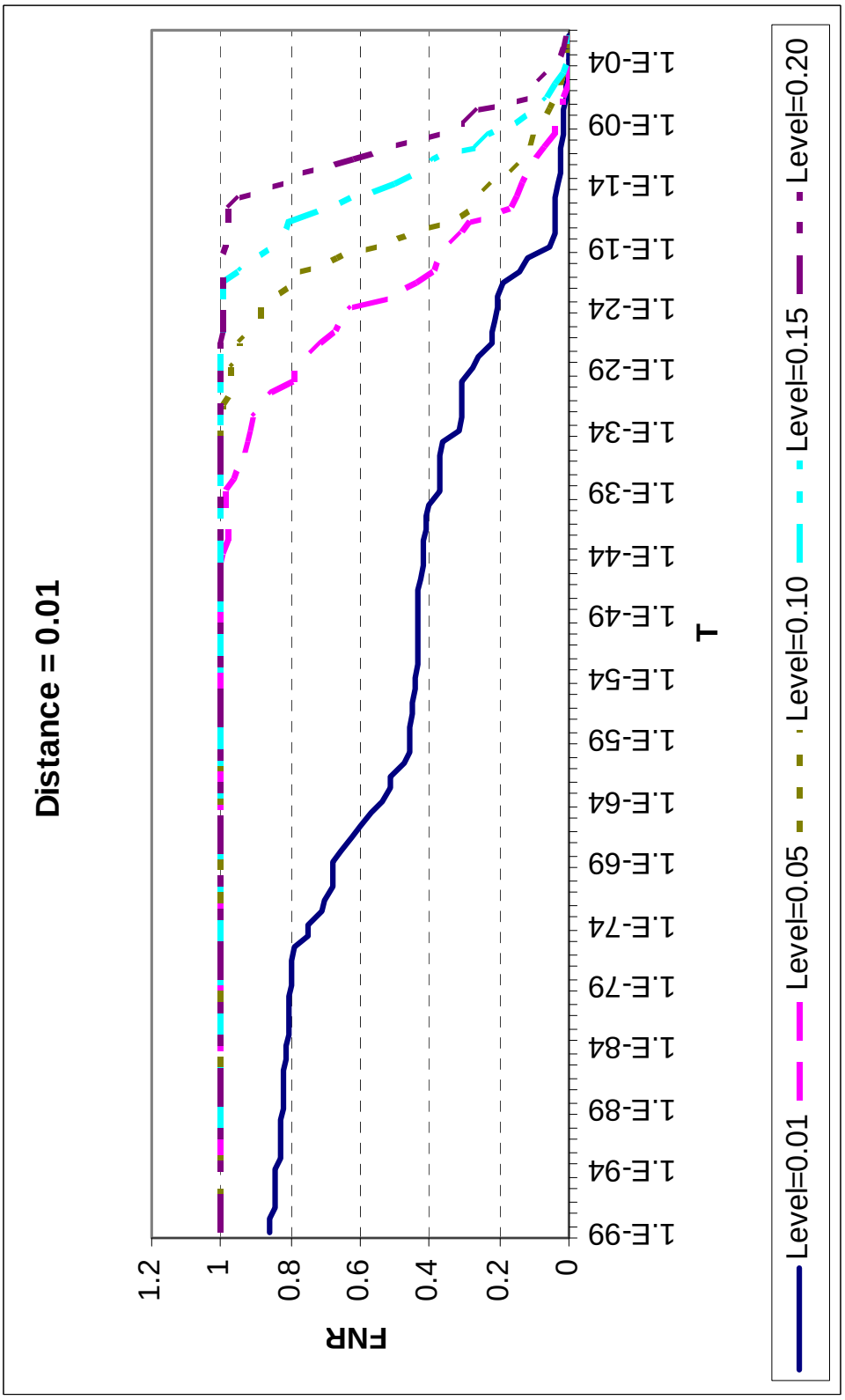


Figure 4.3: False negative rates of different contamination levels at the distance of 0.01.

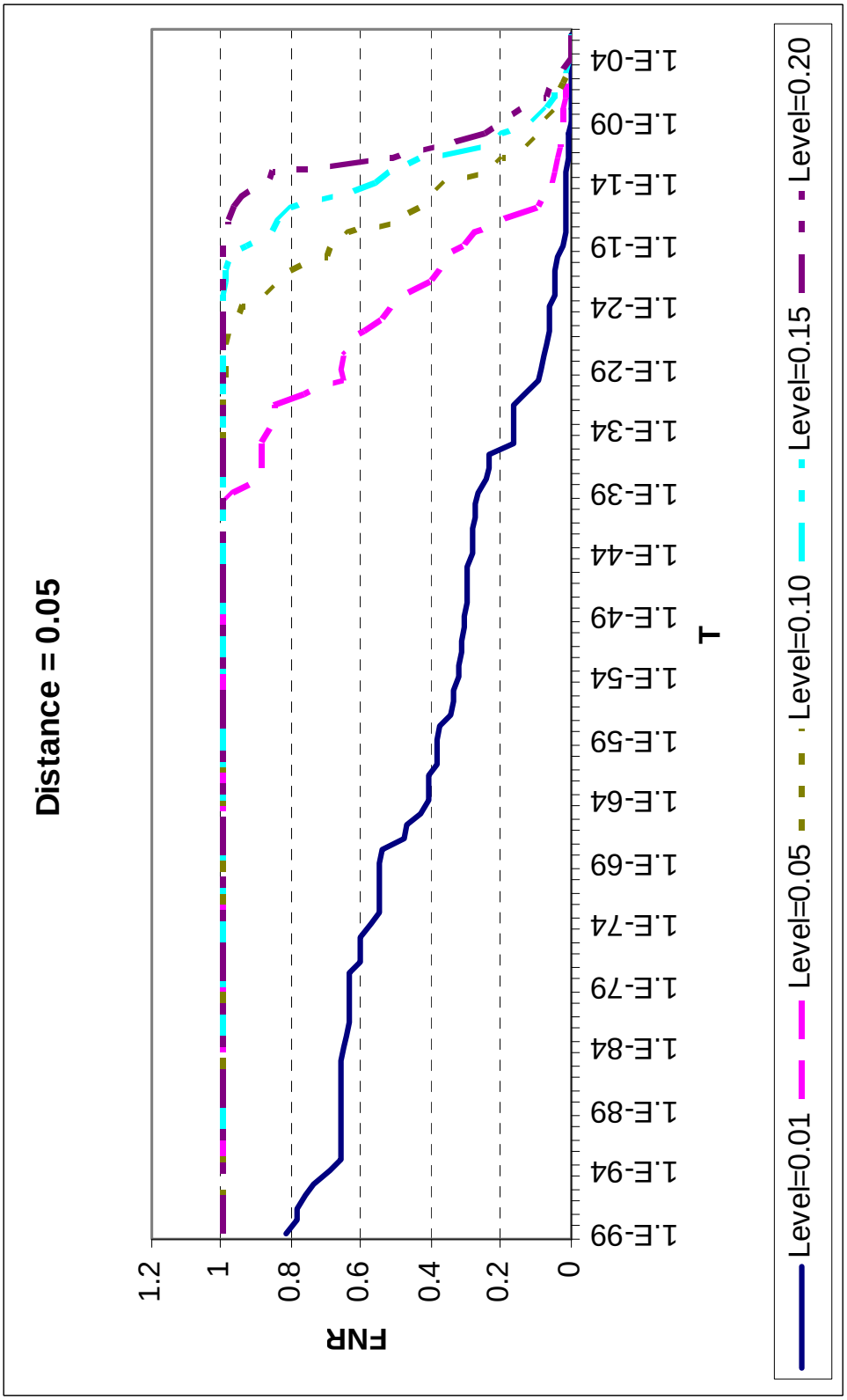


Figure 4.4: False negative rates of different contamination levels at the distance of 0.05.

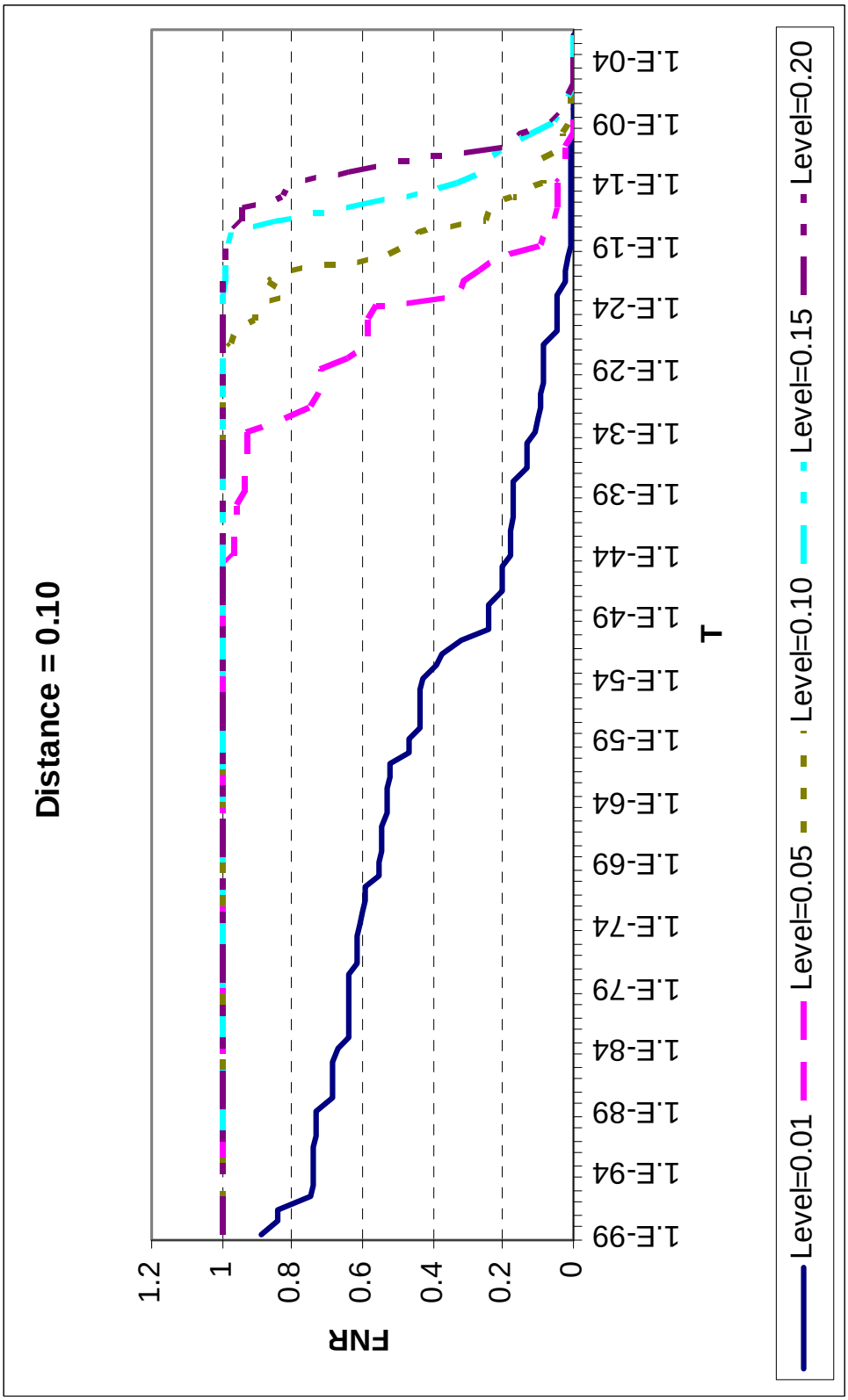


Figure 4.5: False negative rates of different contamination levels at the distance of 0.10.

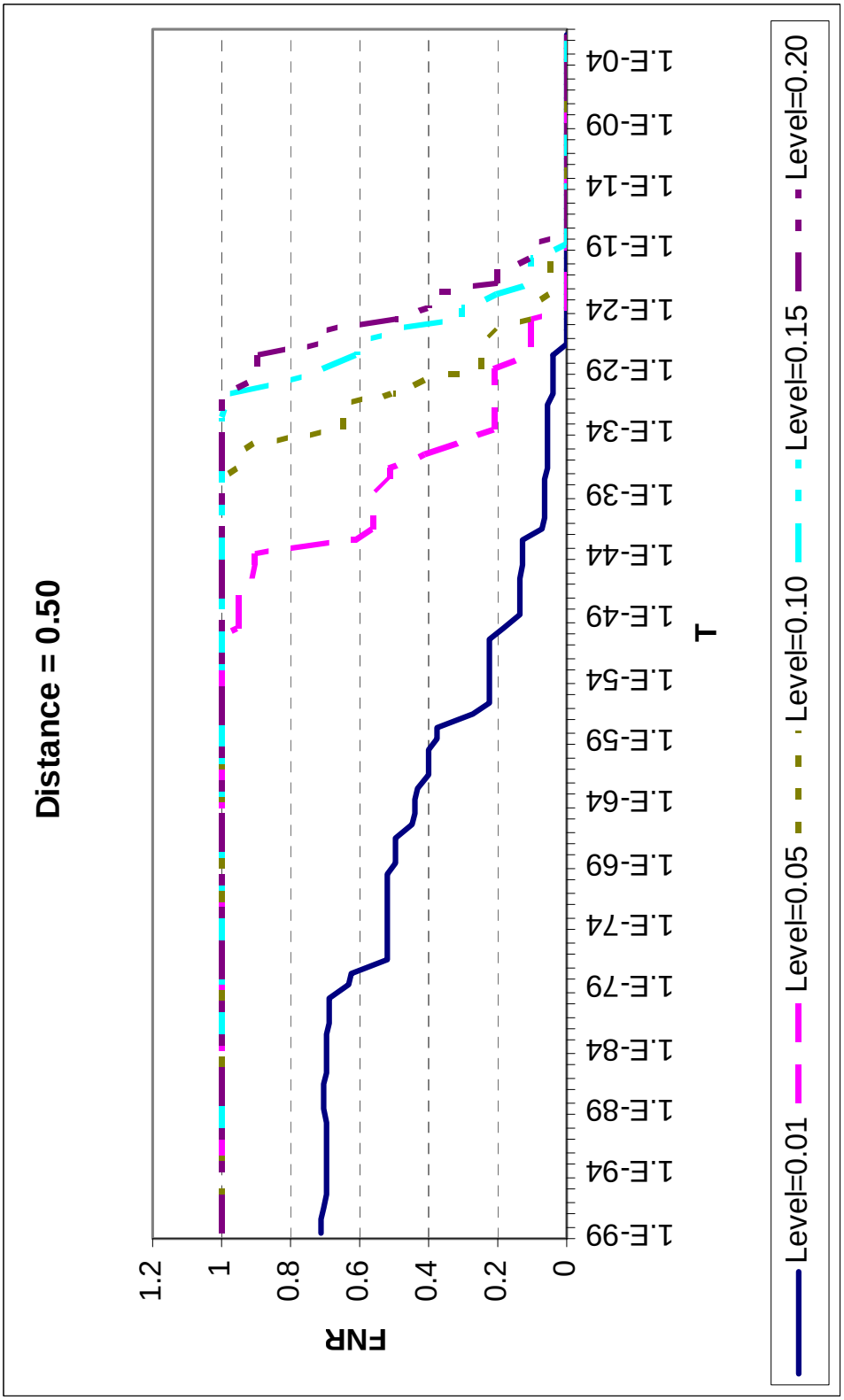


Figure 4.6: False negative rates of different contamination levels at the distance of 0.50.

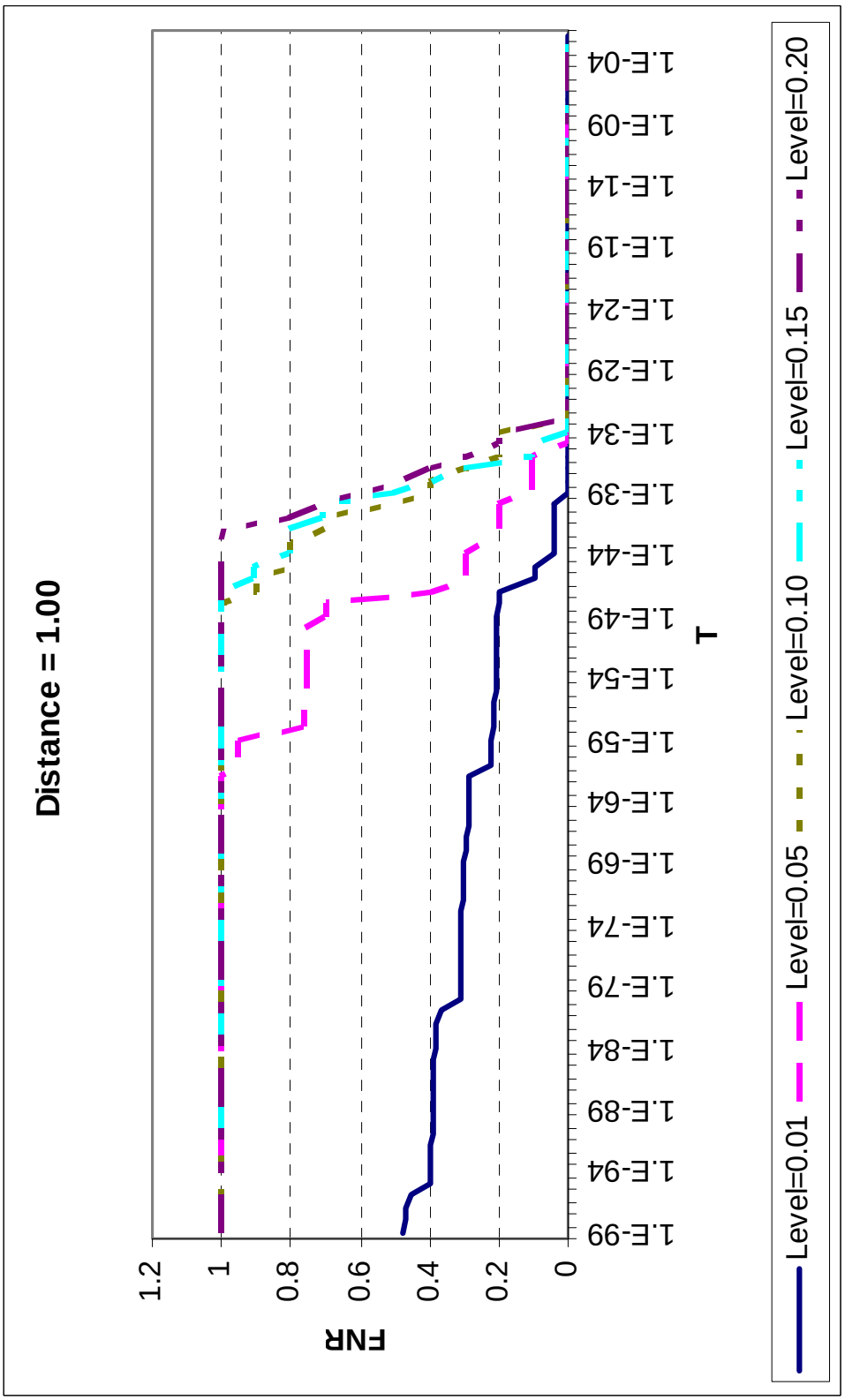


Figure 4.7: False negative rates of different contamination levels at the distance of 1.00.

	Outlier Contamination Level				
Distance	1%	5%	10%	15%	20%
0.01	1E-3	1E-3	-	-	-
0.05	1E-6	1E-3	1E-3	1E-3	1E-3
0.10	1E-10	1E-6	1E-5	1E-6	1E-6
0.50	1E-26	1E-24	1E-19	1E-19	1E-18
1.00	1E-39	1E-35	1E-33	1E-34	1E-33

Table 4.3: Minimum threshold with 0 false negatives. The number shown in this table is the minimum thresholds where there were no false negative instances for various outlier contamination levels and distances.

In the case where the outlier distance is 0.01, i.e., the outliers start only 1% above (and below) the expected range and so are almost continuous with the normal data, the algorithms can successfully detect all outliers only when the number of outliers is less than or equal to 5%. When the contamination level is higher than or equal to 10%, the algorithm cannot tolerate the contamination; i.e., some outliers are not detected, but are included in the model. We do not consider an inappropriate increase of 1% as a realistic problem. However, where our system would fail would be an attack when 10% of the data was 1% above the range and this was gradually stepped up 1% at a time.

In conclusion, all outliers can be detected as long as the outliers are not very close to the normal data (i.e., ≥ 0.05). To configure the algorithm with zero FPR and zero FNR, the threshold must be carefully configured, depending on outlier characteristics. The simulation suggests that the threshold level depends more on distance than contamination level. If outliers are 10% away from normal data, the threshold should be configured between 1e-6 and 1e-4. If outliers are 50% away, threshold can be between 1e-18 and 1e-4. Note from Table 4.2 all thresholds $\leq 1.0e-4$ gave zero FPs.

4.5 The Effect of A Normal Distribution

The proposed algorithm is based on the assumption of uniform distribution, but we have also evaluated its performance on normally distributed data. Simulations are conducted on a scenario where a data set follows a normal distribution. The

performance of the algorithm is evaluated by the false positive rate, as defined in equation 4.1

Here, a standard normal distribution, which is a normal distribution with a mean of zero and a variance of one, is used to generate a data set of 3,000 random numbers. Simulations are run separately on 10 data sets. The average FPRs over these 10 runs are presented.

Data Set Generator

An instance of class java.util.Random is used to generate random numbers. To generate a sequence of normally distributed double values with mean 0.0 and standard deviation 1.0, method nextGaussian() is invoked repeatedly until a sequence is filled up. In this case, we generate 3000 values.

Experimental Results

Simulation results are summarized in Table 4.4. This simulation focused on false positive instances produced by the OE and BA algorithms. In the table, false positive rates or FPRs are shown for different threshold T's. For convenience, we group together those thresholds which produced the same FPR. Results from Table 4.4 are graphically plotted in Figure 4.8. As expected, the smaller the threshold T, the lower the FPR.

T	FPR
$T \leq 1e-10$	< 0.0100
1e-09	0.0103
1e-08	0.0146
1e-07	0.0153
1e-06	0.0184
1e-05	0.0271
1e-04	0.0395
1e-03	0.0824
1e-02	0.1375

Table 4.4: False positive rates (FPR) of various threshold T of the OE and BA algorithms on data set following normal distribution.

In statistics, a hypothesis is most commonly tested with the significance level (type I error) $\alpha = 0.05$. The algorithms OE and BA achieve this rate when the

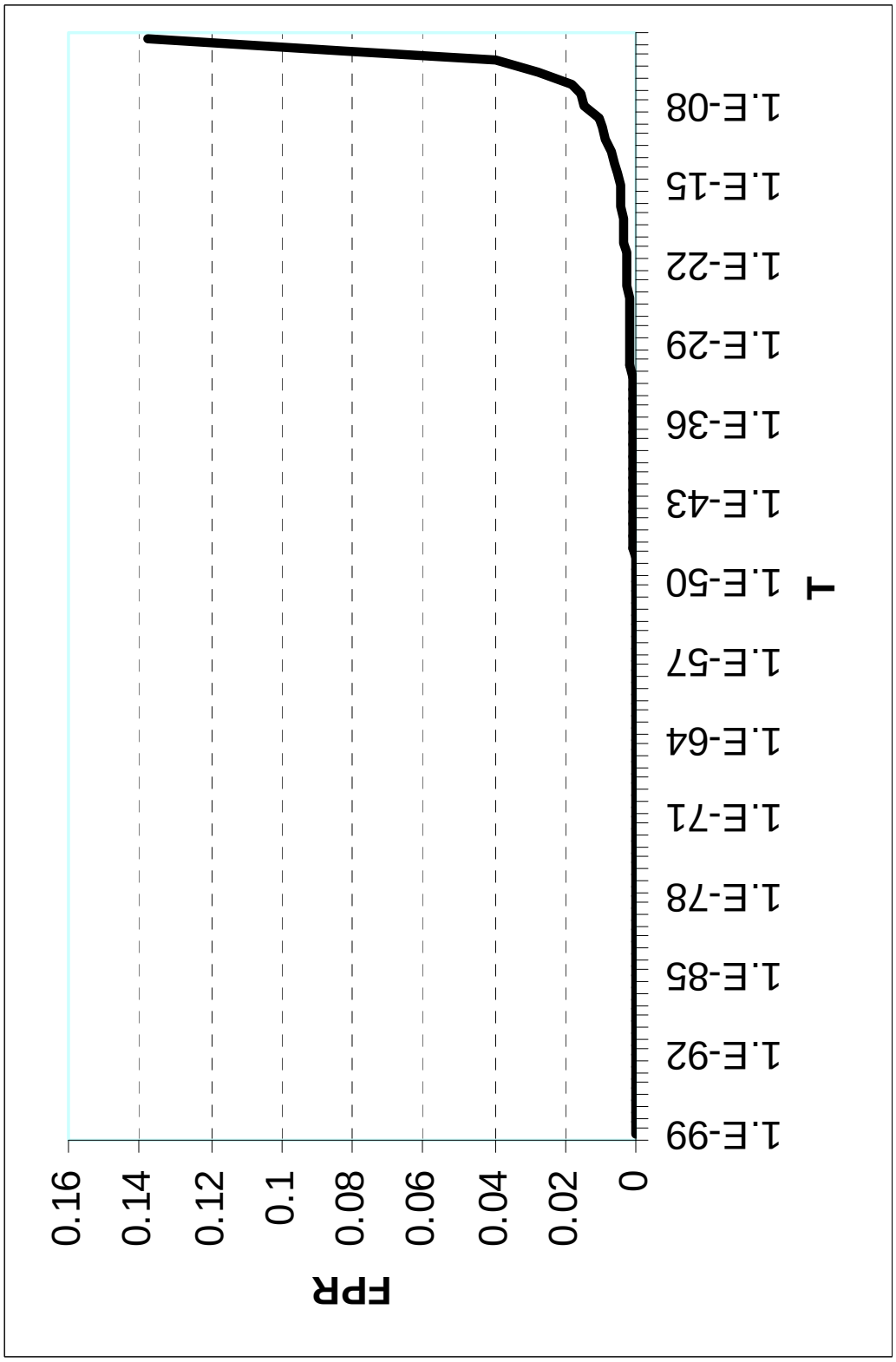


Figure 4.8: False positive rates for normally distributed data.

threshold T is less than 1e-4. The algorithms can also produce FPR less than 0.01, when the threshold T is less than 1e-10.

We note again that it would be important to use the most appropriate distribution function if we were trying to model the central tendency of the distribution. Rather what we are doing is trying to determine limits above or below which a value is suspicious. We believe this is how system managers operate. In our method, we are therefore not concerned whether the values observed in a subrange are bundled towards one end or more uniformly distributed. We are not interested in whether the probabilities we calculate are true probabilities, but rather whether we can find a cut off that minimizes FPs and FNs.

4.6 Conclusion

In this chapter, we show that the simple OEBA algorithm is able to learn the range of a continuous attribute in dynamic domain as well as detect outliers in the data set. There exists a range of a threshold for the algorithm to produce zero FPR and zero FNR. Although the method is based on a uniform distribution assumption, when the simulation is extended to run on data following normal distribution, the algorithm behaves reasonably. Following a normal distribution, the threshold $T < 1e-4$ produces FPR < 0.05 and $T < 1e-10$ produces FPR < 0.01.

In a network traffic domain, the distribution of network volumes has never been well defined. As by nature the usage varies from one site to another, it is not worthwhile to put much effort to define a data set that might not be useful to other sites. As will be explained in the next chapter, our technique partitions network traffic into sub-regions of homogeneous traffic, which are assumed to follow a uniform distribution. Traffic in each region will be learned by the algorithm OEBA. Generally, network attacks alter traffic volumes significantly from their normal levels. Our simulations looking at FPs and FNs suggest that a threshold between 1e-30 and 1e-4 can be used.

Although the assumption of a uniform distribution is not correct, the impor-

tant question is whether it introduces errors. In the experiments we have conducted there were no false negatives, so at the most the distribution assumption is introducing false positive warnings. False positive warnings are likely to occur when the real distribution has a long tail compared to a uniform distribution, and our simulation using a normal distribution showed a higher level of false positives as expected, so that the false positives in our experiments may be due to the distribution. On the other hand a different distribution model may introduce more false negatives than we observe. Ethernet traffic is known to be self-similar (Leland et al., 1994), so it may be worthwhile carrying out simulations of self-similar data, or exploring whether we can use a distribution that captures this self similarity. However, it seems unlikely to be an important issue as what we are concerned with is sudden changes, in network traffic. It is also worth noting that we have used median data from different time periods, so to some extent we are considering self-similarity.

Chapter 5

Network Traffic Anomaly Detection

Network administration commonly involves deployment, configuration, maintenance and monitoring of both hardware and software in a network. However, what this study concerns is the monitoring task. To keep a network free from being compromised, a network administrator constantly monitors network status, and rectifies the network if any break-in is detected.

One of the fundamental tasks in today's network monitoring operations is anomaly detection. Traditionally, anomalies such as DOS attacks, flash crowds, port scanning or worms are detected through the analysis of data inside packet headers, e.g. service, duration, src_bytes, dst_bytes of IP packets (Porras and Neumann, 1997; Smaha, 1988; Anderson, Frivold, and Valdes, 1995; Ilgun, 1993; Lunt et al., 1988; Axelsson, 2000; Cannady, 1998; B. Mukherjee, 1994; Ryan, Lin, and Miikkulainen, 1998; Yamanishi, Takeuchi, and Williams, 2000; Vaccaro and Liepins, 1989; Guan, Ghorbani, and Belacel, 2003), as all information regarding the communication is stored in these headers. More recently, the volumes of network traffic measurement, e.g. protocol byte count, link count, have been proposed as an alternative source of audit data (Barford et al., 2002; Brutlag, 2000; Krishnamurthy et al., 2003; Lakhina, Crovella, and Diot, 2004b; Mandjes, Saniee, and Stolyar, 2005; Soule, Salamatian, and Taft, 2005). Although traffic volume

contains less information than that inside packet headers, it has been proved to be meaningful enough to reveal such network anomalies as DOS attacks, flash crowds, outages and large file transfer(Barford and Plonka, 2001). Our focus is also on network measurement data.

Traffic volume data is a sequence or time series of bandwidth usage in a network, which can be collected by many network tools such as FlowScan (Caida, 2006) or RRDtool (Oetiker and GNU, 2006). Graphical visualization of traffic volume can show the status and performance of the network, e.g. heavily used, slightly used or whether an outage has occurred. When a network is attacked, traffic volume will significantly change from its normal state; either increasing or dropping for a period of time, depending on the attack type. This helps network administrators detect anomalous traffic. For example, FlowScan is used in (Barford and Plonka, 2001; Barford et al., 2002) to periodically maintain counters of attributes, e.g. IP protocol, and such services as FTP, HTTP. Time series data is recorded into a database for both archiving and visualization purposes. For graphical visualization, the aggregation over five minute intervals for a given time period is used instead of individual data points. Even though this level of aggregation is sufficiently coarse-grained that short time scale behavior will be missed, it has been shown that this does not suppress the manifestation of anomalous behavior (Barford and Plonka, 2001).

To detect anomalies from network measurement data, network tools are generally used to collected network measurement and generate visual graphs for network administrators to inspect. If an anomaly is detected (manually and visually), further investigations are performed to identify the root cause of such an anomaly and a remedy is provided, if necessary.

To automate such a task, an anomaly detector is implemented to filter suspicious data by comparing new observations with predefined system profiles of normal characteristics. There are two primary tasks in the process: 1) construction of normal system profiles, 2) differentiation of suspicious from normal traffic.

To construct normal behavior profiles, many approaches have been proposed to date. In early literature, statistics-based approaches, for example, the Holt-

Winters forecasting algorithm, and Auto-Regressive Integrated Moving Average (ARIMA), were used to represent traffic profiles(Brutlag, 2000; Krishnamurthy et al., 2003). Not only being used to model system behavior, they were used as methods to evaluate other approaches, such as wavelet analysis, and principal component analysis, which were proposed to detect traffic anomalies(Barford et al., 2002; Lakhina, Crovella, and Diot, 2004b). In these approaches, a single mathematical model is intended to cover the full range of normal behaviors sufficiently and precisely to exclude outliers.

The second task, i.e., differentiating anomalies from normal traffic, has been commonly implemented by using a threshold. The threshold is defined differently according to the implementation of the model. Between the two, the profile construction task is considered more sophisticated and time consuming.

As an alternate, we propose an RDR-based approach to learn, instead of a single universal model, a hierarchy of network behavior profiles, where each of which represents a subspace of homogeneous traffic. Our approach also uses a variable threshold based upon seen observations to define the confidence band. The more instances that have been observed, the more confident the algorithm. This works reasonably well (Prayote and Compton, 2006).

The rest of this chapter is organized as follows. Section 5.1 explains the actual manual anomaly detection task in the School of Computer Science and Engineering (CSE) at the University of New South Wales. Section 5.2 discusses the methodology we have used to automate traffic anomaly detection in CSE. This includes segmentation of the problem space, the modification of the Ripple Down Rule infrastructure and the overall architecture. The implementation of the methodology is explained in Section 5.3. This section also discusses the performance evaluation of the approach on real traffic data. Finally, the chapter is concluded in Section 5.5.

5.1 Manual Network Anomaly Detection

In CSE, one of the network administration routines is manual monitoring of network usage. CSE network usage is archived by the program RRDtool (Oetiker and GNU, 2006). It also generates graphs for the network administrator to inspect. If an anomaly is detected, further investigation is performed to identify the root cause of such an anomaly and a remedy is provided, if necessary.

5.1.1 Audit Traffic Data

RRDtool can be configured to meet variable requirements of users. Here, the administration team is interested in the bandwidth consumption of traffic of different service protocols, different servers, and different traffic directions.

Server Type. A server is classified by its location as either an *internal* or *external* server. Under the CSE policy, servers physically located within the school are classified as internal. All the rest are classified as external.

Traffic Direction. By its direction, traffic can be classified into two types, i.e., *incoming* or *outgoing*.

Service Protocols. CSE provides 7 service protocols to users. These are HTTPS, POP3, RSYNC, SMTP, SSH, TELNET and WWW. Not only is RRDTool configured to record network usage by these protocols, it is additionally set to keep track of the total bandwidth consumption of incoming and outgoing traffic. As a result, at every minute, RRDTool records 30 values for network usage, for example, a value for incoming traffic to the internal HTTPS server, a value for outgoing traffic from internal WWW server, etc.

Graphs that are generated by RRDtool are inspected by the network administrator in the following fashion. The administrator literally reads IP data first. If IP data does not follow patterns of past observation, they will perform further

investigations on particular protocols to identify such an anomaly. From past observation of IP data in our network, network usage typically starts to climb from 8am to its peak. The peak is from 2pm to 4pm. The usage slowly drops until 2am and it stays steady until 8am. This cycle is the basic pattern of the network usage.

5.1.2 Actions on Anomalies

It is worth noting that not all anomalies (manually) detected indicate intrusive behaviour; some of them might be just an ad-hoc and legitimate behavior, some are ad-hoc, but not legitimate. For example, there was a leap of bandwidth usage in one morning, far from normal, for about half an hour. From investigation, it was just a huge file transfer between a server and a machine in the network. It was ad-hoc and no harm was caused by this event; it was then considered legitimate.

Another example was similar, but a different conclusion was made. Graphs showed that traffic usage leapt every two hours, and lasted for half an hour. From investigation, it was an attempt to send an email to a Google website, which was rejected. After the cause was identified, the administrator terminated the process which kept sending the email.

The above are two actual scenarios that occurred in the CSE network. Actions in response to detected anomalies are determined after the root cause of anomalies is identified. In CSE, if an anomaly detected is considered not harmful, then the network administrator typically ignores it. This happens over and over again. We need to mention that the identifying task requires expertise of the administrator and, of course, some time to derive a conclusion. If there are too many detected anomalies without intrusive behaviour (or false positive instances), the network administrator might miss actual intrusions.

Our study aims at developing an automated system for anomaly detection; that is, the system is expected to detect all suspicious behaviour and send a report to the network administrator. The scope does not include identification and rectification. However, we wish to keep the false positive rate as low as possible.

5.2 Using RDR to Automate Traffic Anomaly Detection

Like other paradigms, there are two main tasks in an anomaly detection system, i.e., profiling and detection. The profiler is a component which controls the learning and the maintenance of profiles in the domain. The detector investigates audit data with maintained profiles and issues warnings or reports to users.

In our approach, Ripple Down Rules as reviewed in Chapter 3 is applied to operate in both profiling and detection. The framework is explained by these two tasks. Section 5.2.1 discusses our profiling component in detail. The detector component is explained in Section 5.2.2.

5.2.1 Network Traffic Behaviour Profiling

A typical intrusion detection system requires that profiles must be ready prior to its application. Under this requirement, most IDS's contribute training sessions to generate their profiles. In contrast in the RDR paradigm, an RDR-based system learns knowledge while in use; that is, an RDR-based system generally starts with an empty knowledge base and new knowledge can be added at any time appropriate. Here, we introduce a new variant of RDR methodology to incrementally learn profiles of network traffic. This section starts with an explanation of the underlying concept of network profiles.

Situated Network Traffic Profile

Our approach aims to incrementally learn profiles of network traffic. Each profile represents a particular situation in a network, which represents a particular pattern. Different situations manifest different traffic patterns. For example, every weekday at 6am, the network consumption leaps up to about 5 MB per minute and lasts for 30 minutes due to a backup; hence, this might be denoted as a "weekday-6am" pattern. The intention of RDR partitioning is to provide regions where the data is

homogeneous and the same behaviour occurs. We therefore assume each attribute is uniformly distributed within a partition. The adequacy of this assumption will be reflected in the performance of the final system.

Upon this assumption, we can apply the algorithm Outlier Estimation with Backward Adaptation (OEBA) to learn each attribute and detect outliers occurring in the situation. That is if we use n attributes, we will have n profiles maintained for the situation. These profiles work separately by only monitoring their corresponding attributes. For convenience, we denote profiles of all attributes of a situation as a *situated profile*.

Profile Construction and Organization

Like other RDR paradigms when new knowledge is added when required, each situated profile is created when a new situation is identified and a rule is added. Under the RDR paradigm, adding a new rule to a knowledge base requires human experts to choose new conditions and a conclusion for the new rule. In contrast, our approach requires only the new conditions identifying a new situation, no conclusion is necessary because rules are intended to identify an appropriate situated profile for a particular case.

Hence, a modification is made to the RDR structure. From a typical RDR rule structure of

IF (a condition) THEN (a conclusion).

Our approach modifies the rule structure to

IF (a condition) THEN (a situated profile).

We could say that the approach finds the most suitable profile for any particular case. With these changes, we name this version of the Ripple Down Rule based structure as Ripple Down Models or RDM.

5.2.2 Traffic Anomalies Detection

Generally, there are three main tasks in anomaly detection. First, a profile for a case must be retrieved. Second, the case is matched against its profile. Third, a conclusion is made whether the case is an anomaly.

Situated Profile Retrieval

In our paradigm, profiles are maintained in an RDM-based knowledge base. A situated profile for a particular case is retrieved by a typical RDR inference process. For example, if an RDM-based knowledge base is implemented in a binary tree, a situated profile of a case is taken from a last fired rule. On the other hand, if a decision list is implemented, a situated profile is taken from a first rule that fires.

Situated Profile Matching

Once a situated profile is retrieved from an RDM-based KB, a matching engine is used, based on the algorithm OEBA, to detect outliers. There is an important point to be noted here. A situated profile denotes profiles of all attributes of interest. If there are n attributes explaining traffic characteristics, a situated profile contains n OEBA-profiles. Consequently, there are n matching results from the matching process. These n results are then used to decide about a case.

Network Traffic Classification

To conclude whether a traffic situation is an anomaly, we employ a further RDR knowledge base, instead of simply summarizing from the results given. The reason is explained below.

First, there may be more than one attribute explaining traffic characteristics. For a particular situation, one attribute might be more important than others. That is experts might justify a situation based on some particular attributes, rather than all of them.

Second, although OEBA is efficient in outlier detection, there is a possibility

of false positive instances. To reduce the false positive rate, our technique allow experts to make their justification on top of OEBA outlier detection results. These justifications are maintained in an RDR knowledge base.

5.2.3 Discussion on RDR-based Anomaly Detection

In summary, our paradigm proposes a double knowledge base solution for network traffic anomaly detection. The first knowledge base maintains profiles of network behaviour, while additional expert justifications which draws conclusions from profiles are maintained in the second knowledge base. Both knowledge bases are implemented using Ripple Down Rule-based technology. Ripple Down Model (RDM) is used to construct the knowledge base of profiles, while the original RDR is used for the knowledge base concluding whether a case may be an anomaly.

Ad-hoc Network Events

An RDM-based profiler offers two advantages in network intrusion detection. First, no training session is required, which significantly reduces development time. Second, new patterns can be added at any time they are discovered without contaminating previous models or profiles. A profile is implemented with the algorithm OEBA. This algorithm is relatively simple because it is applied to relatively homogeneous sub-regions but it has to behave appropriately when a new region is identified and little data has been seen in this region.

Network traffic characteristics change on daily basis. Although, base patterns of network traffic may be stable, there often are events that are ad-hoc and deviate from these base patterns. Without a proper solution to handling these ad-hoc events, an IDS tends to cause a high false positive rate. RDM can easily address this problem by adding new events as new knowledge to a knowledge base. This will not affect base patterns.

Domain Segmentation

In the profiler, the RDR method is used to locate an appropriate profile for a situation in a multidimensional space. Network traffic is defined in a multidimensional space where there are many attributes defining its characteristics. A situated pattern, which is bounded in a smaller multidimensional sub-space, can be characterized by a certain range for each of the component attributes. Our approach utilizes the RDR functionality of segmentation to partition the problem space into regions of homogeneous data for particular situations.

Like the original RDR paradigm, the partitioning is performed through the addition of a new rule when required. As stated in the RDR segmentation section, the correction process of RDR allows further patterns to be added when required. Each subspace is formed through partitioning the problem space via interactive sessions with human experts while the system is running, without separate training sessions like other traditional approaches. That is, the system gradually takes over while being trained in the job like an apprentice.

5.3 System Implementation

We have implemented a traffic anomaly detection system for the school of Computer Science and Engineering (CSE), the University of New South Wales (UNSW), Sydney, Australia. The system reads input from IPflow archives generated by RRDTool (Oetiker and GNU, 2006). For any outlier detected, a warning is raised for further investigation by a network administrator in the school.

An architecture for the system is first described in Section 5.3.1. Network traffic, which is our audit data, has already been discussed in Section 5.1. However, under an RDR paradigm, audit input is considered case by case, Section 5.3.2 discusses how a case is formed from a network traffic stream and what attributes a case consists of. Finally, an implementation of both knowledge bases is outlined in Section 5.3.3.

5.3.1 System Architecture

A prototype system is implemented as illustrated in Figure 5.1. The system consists of a case generator, a profiler, an anomaly detector and two knowledge bases, i.e., a situated profile KB and a final decision KB. A case generator extracts features from network traffic and generates audit data cases. A profiler is responsible for four tasks; i.e., profile retrieval, profile matching, profile learning and profile construction. That is, a profiler is a component that gains access to and maintains an RDM knowledge base of situated profiles. Two other tasks, namely, case classifying and rule construction are controlled by an anomaly detector; it maintains the other RDR knowledge base of final decision.

The whole process is as follows. When a case is generated and passed to a profiler, the profile retrieval engine is first activated to find an appropriate situated profile for the case. It is worth noting that, under an RDR paradigm, there is always a default rule for any case; that is, any new observation might be fired by this default rule. Hence, a default profile is available.

A retrieved situated profile is passed along with the case to the profile matching engine. This engine matches a case against all its OEBA-based profiles. Matching results are next sent along with the case to the detector.

Once a case and its matching results are available to the detector, it starts the classification engine to conclude a case. A derived classification is presented to experts. The process should stop here, provided the classification is correct. Otherwise, an expert justification is fed back to the system; starting either the profile construction engine or the rule construction engine depending on the justification or if the warning was unnecessary, a false negative, the profiles for the case are updated.

To correct an incorrectly classified case in typical RDR, an expert is required to select some salient features from the case that make it different from a cornerstone case to form the conditions in a new rule. Then the new rule is added to a knowledge base. In our system, this process is slightly extended. If the retrieved situated profile is believed not appropriate for the case, a new profile should be added to

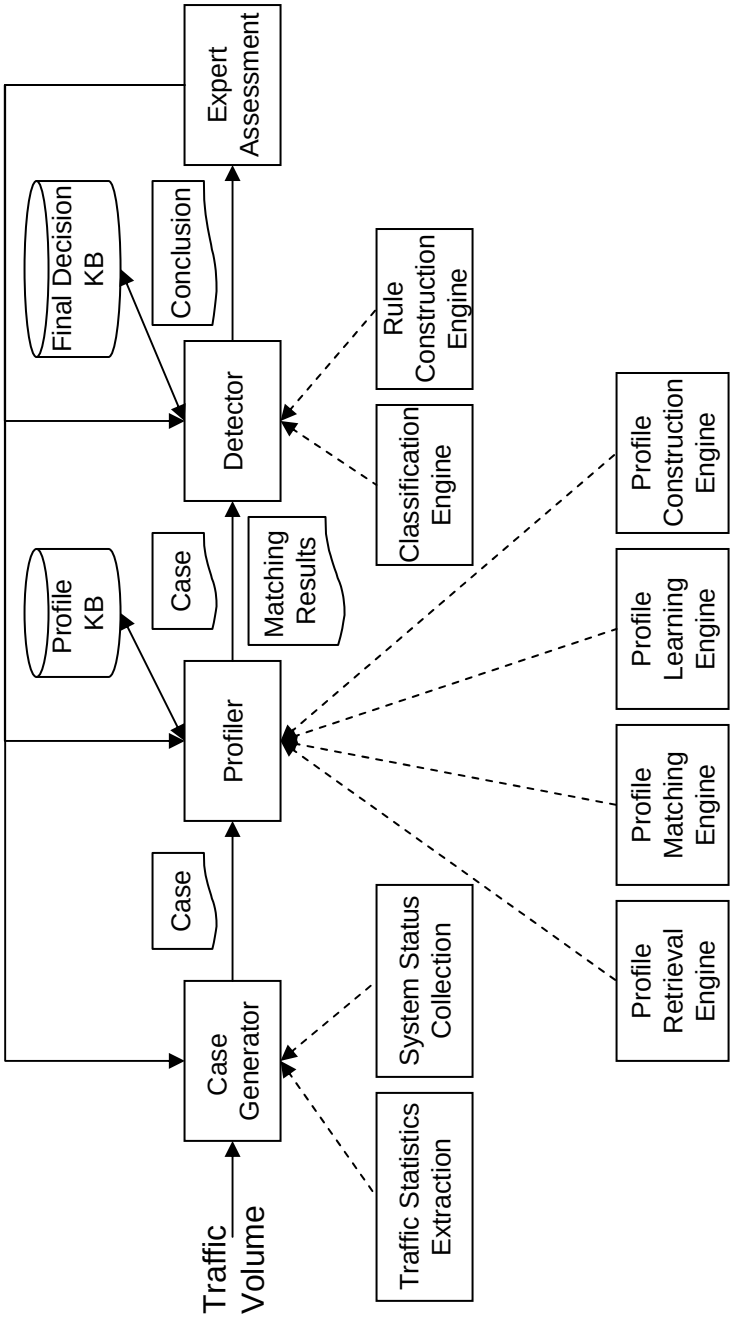


Figure 5.1: A system architecture of RDR-based network traffic anomaly detection.

the situated profile KB. The expert simply defines a new situation for the case, and a new rule and a new profile are automatically added to the KB. If the retrieved situated profile is appropriate for the case, a new rule is added to the expert justification KB and the expert needs to select salient features to compose a new rule, as usual.

5.3.2 Periodic Monitoring Cases

As mentioned in Section 5.1, the school of Computer Science and Engineering (CSE) maintains network usage by a program called RRDTool 5.1. RRDTool is configured to sample data every minute for every protocol used, resulting in a stream of network snapshots. In the RDR paradigm, problems are solved case by case. Inevitably, our RDR-based anomaly detection system analyzes traffic on the basis of cases. A case is a set of data that an expert considers in reaching a conclusion. For example, in the interpretation of pathology results, a case is a set of current and previous pathology results that are available for a patient plus other patient information such as age and sex.

In network monitoring, an automated system generally monitors a network in a constant and periodic fashion. For example, Barford et al. (Barford and Plonka, 2001; Barford et al., 2002) sampled data every 5 minutes. Krishnamurthy et al. (Krishnamurthy et al., 2003) also investigated network traffic with an interval size of 5 minutes. Mandjes et al. (Mandjes, Saniee, and Stolyar, 2005) implemented their work on a window size of 5 minutes. This interval size is not trivially chosen. It has been demonstrated that although a 5-minute interval is coarse-grained enough that short time scale behavior will be missed, this does not suppress the manifestation of anomalous behaviour (Barford and Plonka, 2001). We have also chosen a window size of 5 minutes for our monitoring intervals. That is, the monitoring process is periodically invoked every 5 minutes or every 5 records of data archived by RRDtool. Every time the monitoring process is invoked, a case is generated with attributes of interest to be further investigated. A next concern is what attributes a case should be composed of. As will be discussed, we then calculate

running statistics, such as median, mean, standard deviation, over 20 minutes and 60 minutes for each attribute. All of these make up the case.

Temporal Constraints

A network can have numerous situations with different traffic patterns. A key factor is access demand. When demand changes, so does traffic patterns. For example, in a situation where the demand is high, the network is highly utilized and less utilized when demand is low.

Demand to access a network is caused by users and system configuration for that network. Most system services are configured to execute at certain times, e.g., the network administrator of CSE configures RSYNC to run every morning from 6 am. till the backup is finished. During that period of time, network consumption would be exceptionally and instantly high, compared to the last hour. Without any background knowledge on this configuration, one might think that the network may be under an attack. Users in the organization directly affect network utilization. For example, network consumption in CSE is generally higher during a semester than a break. Patterns of behavior are also mainly temporal, e.g., working from 9am to 5pm on weekdays, and less on weekends, etc. That is the demand to access a network significantly depends on temporal constraints.

In our implementation, temporal attributes are extracted from time stamps as follows:

- TimeOfDay.

  In the CSE network, most traffic patterns occur repeatedly at the same time of a day, for example, the CSE network has low utilization every morning from 7am to 9am, etc. An attribute to give a description of the time when a case is monitored is required. We use the system of 24 hours a day to specify a value for the attribute TimeOfDay; i.e., from 00:00 to 23:59. In our prototype, an expert can specify a value of TimeOfDay to the granularity of

a minute.

$$\begin{aligned} TimeOfDay &\Leftarrow Hour : Minute \\ Hour &\in [0, 1, 2, ..., 23] \\ Minute &\in [0, 1, 2, ..., 59] \end{aligned}$$

- DayOfWeek. Not only does time of a day affect traffic patterns, network manifestation also varies on which day of a week, for example, CSE network utilization on Sundays is generally lower than other days. To provide choices, an attribute DayOfWeek is defined as follows.

$$\begin{aligned} DayOfWeek \in [&Monday, Tuesday, Wednesday, Thursday, Friday, \\ &Saturday, Sunday, Weekday, Weekend] \end{aligned}$$

  Not only days of a week are provided, Weekday and Weekend are two special abstractions of actual days in a week. Weekday covers Monday to Friday, while Weekend covers Saturday and Sunday. When a case is generated, DayOfWeek is only assigned with actual day of a week. The two abstractions, Weekday and Weekend, are provided to support an expert in making a new rule.

- Season. Another temporal factor that affects network usage is university season. During special times of the year, e.g., public holidays, the number of users accessing the university network significantly drops. Traffic patterns during this period are generally different from other times during the year. We, hence, provide a third temporal attribute for traffic; that is, Season.

$$\begin{aligned} Season \in [&Semester, Recession, PublicHoliday, \\ &Stuvac, Examination] \end{aligned}$$

  A case is assigned with Semester when a semester is on, Recession when it is a semester break, PublicHoliday when it is a public holiday, Stuvac when it is during a study vacation, and Examination if it is during an examination.

Traffic Statistics

The simplest statistic for network traffic is measured volume. However, using such raw data is at a risk of a high false positive rate because traffic volume is well-known for being highly fluctuating. Most systems in general have chosen to preprocess a vector by using a smoothing function, e.g., a moving average (Brutlag, 2000; Barford and Plonka, 2001; Barford et al., 2002), or they extract some statistical measures as representatives, e.g., sketches (Krishnamurthy et al., 2003), traffic variance (Mandjes, Saniee, and Stolyar, 2005), prior to an analysis.

In our framework, a traffic situation is understood by an administrator according to two main characteristics. First, an administrator looks at overall traffic consumption, not just a particular point. That is, there might be a spike or two in traffic (visualized in a graph), but this does not matter as it is typical for network traffic sampling. A spike would start to matter when it lasts for a certain period of time, for example, 10 or 15 minutes. With this point of view, we decided to use a central tendency, i.e, mean or median, to represent overall network consumption at a specific time. A second characteristic is traffic fluctuation.

Central Tendency

The mean and median are equal in symmetric distributions. The mean is a good measure of the central tendency for roughly symmetric distributions but can be misleading in skewed distributions since it can be greatly influenced by values in the tail. Therefore, the median is more informative for distributions such as reaction time or family income that are frequently very skewed. In other words, the median is more robust against outliers. In network environment where traffic fluctuates and contains much noise, the median is more suitable for a central tendency.

In our implementation, we have chosen the median as a statistical measure to represent an overall traffic level at monitoring time. However, different window sizes convey different information. The smaller window is a good measure of rapid changes in traffic, but it is sensitive to noise, especially when noise is high. The large window is more tolerant of noise but it often misses small changes in the

traffic. Figure 5.2 illustrates a graph of network traffic and its two medians; one with 20-minute window, the other with 60-minute window.

From an example in the figure, the median with smaller window size is more suitable to represent current traffic as it better reflect changes in current traffic. On the other hand, the median with larger window size is smoother. It is more tolerant to fluctuation in network traffic; hence it is best used for long term traffic analysis. Here, we have chosen to implement two median statistics; one for a small window size, the other for a large window size. For a small window, we have tried sizes of 5 minutes, 10 minutes, and 20 minutes. Sizes for a large window has been varied between 30 minutes and 60 minutes.

Fluctuation (Serial Correlation)

The median only provides a general idea of traffic level and nothing else. Another important characteristic is how much traffic is fluctuating. For example, Figure 5.3 is an example of two series with same median, but they are different in terms of fluctuation. In our domain, these two series are considered different patterns.

The basic measures to represent dispersion of data are standard deviation and variance. These two measures can be effective in differentiating such series in Figure 5.3. However, these two measures do not take temporal relationship in a series into account; i.e., considering data in the set equally, no matter what order they are in. To simplify discussion, we denote S_i as a series where all values in the series are represented in a pair of curly brackets. For example, two series are:

$$S_1\{1, 1, 1, 1, 2, 2, 2\} \text{ and } S_2\{1, 2, 1, 2, 1, 2, 1\}.$$

Intuitively, S_1 is more stable than S_2. There is only one transition at the fifth value. S_2 obviously exhibits more fluctuation. However, the standard deviation and variance of these two series are the same. Using either of these two statistics as a fluctuation measure is acceptable only in those domains where temporal relationship does not matter. In our domain, these two series are different. The simplest technique to differentiate these two series is to take the transition of adjacent values into account.

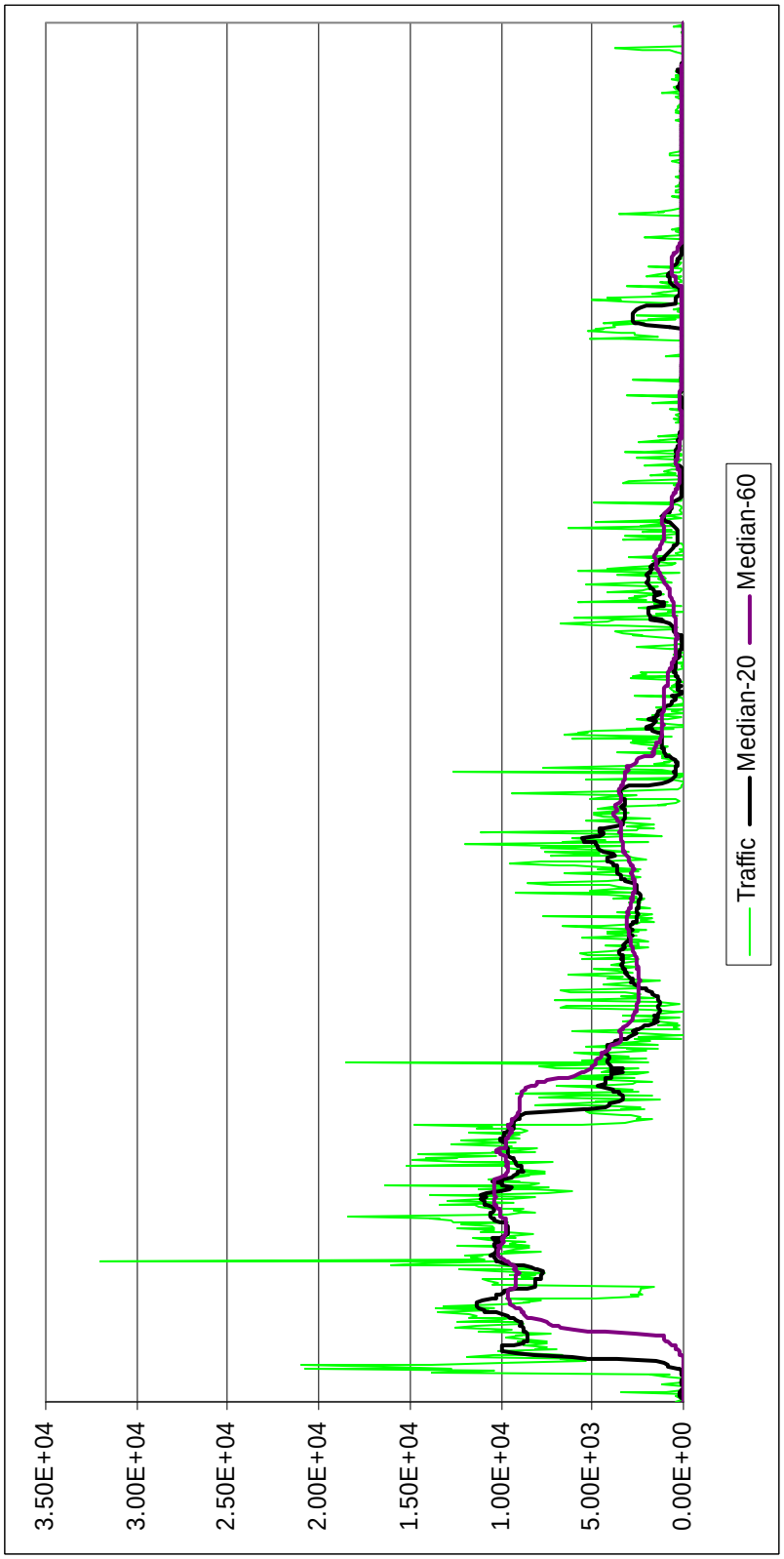


Figure 5.2: An example of network traffic and its medians. There are 2 medians drawn in this graph; i.e., a median with a window size of 20 minutes and a median with a window size of 60 minutes

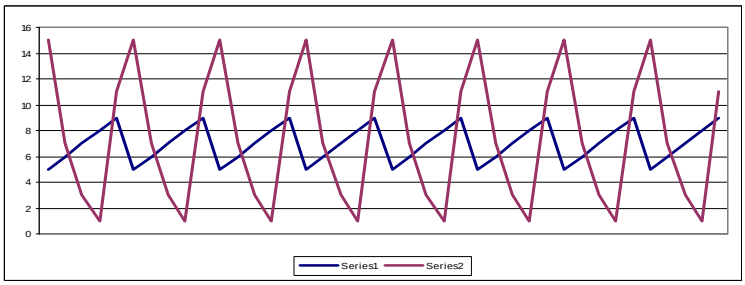


Figure 5.3: An example of two series with same median but different fluctuation.

We denote T as an average transition function which is defined as follows.

$$T = \frac{\sum_{1}^{N-1}(|s_i - s_{i+1}|)}{N-1} \tag{5.1}$$

From this definition, $T(S_1) = 0.167$, $T(S_2) = 1.0$. The average transition function can better reflect the fluctuation of time series than basic measures like standard deviation or variance. In our prototype, a transition function is implemented with the same window size as the small window of the median because we are interested in current or short-termed fluctuation of traffic rather than long-termed fluctuation.

System Status

In order to use the second knowledge base, a number of attributes related to the warnings that have been raised are deployed.

- PreviousState.

  This attribute provides information about a conclusion of the previous case. That is, whether there has been an outlier.

- WarningPeriod.

  If a case is concluded as an outlier, a warning is raised. So far, there has not been information on how long a warning has been raised, yet. We implement an attribute WarningPeriod to keep track of duration of a warning. WarningPeriod is 0 until a case is detected as an outlier, WarningPeriod will be updated incrementally. An increment stops when the system is back to

normal; that is no warning is raised any more.

- DurationFromLastWarning.
  Information on the duration of a normal state since last warning has been made is provided with an attribute DurationFromLastWarning. In contrast to WarningPeriod, this attribute is kept incremented until a warning is raised and is reset to zero when the system state is back to normal.

5.3.3 Knowledge Bases

As outlined, our approach uses a double knowledge base technique to detect traffic anomalies. The first knowledge base maintains situated profiles. The second one reflects expert decision about whether an outlier is of concern. In this section, an implementation of these two knowledge bases is discussed.

Situated Profile Knowledge Base

All situated profiles are maintained in this knowledge base using a Ripple Down Models (RDM) variant of RDR. As discussed earlier, inferencing in RDM is finding an optimal profile for a particular case in a vast search space. Each rule helps segment the search space into smaller sub-spaces with better defined situations or patterns. We have also discussed that a network pattern can be effectively identified by temporal constraints. Hence, in our prototype, only three temporal attributes, namely, TimeOfDay, DayOfWeek, and Season, are part of profile construction. That is an expert is required to specify particular values for these three attributes for a new pattern.

This technique is simple and effective. Firstly, we limit search space dimensions to only temporal attributes. Secondly,temporal patterns may change with deadlines, etc., but we are dealing with temporal changes. The paradigm allows us to introduce new temporal patterns to the knowledge base. Thirdly, other pattern attributes are automatically maintained and learnt by the algorithm OEBA in each profile. A discrepancy detected from a profile has a new profile generated

for a new pattern. That is a construction of a new profile is a refinement of the previous pattern. In this view, our technique gradually refines traffic patterns to their proper specialization.

Final Decision Knowledge Base

On top of matching results from those OEBA profiles, experts make a final conclusion about whether a system administrator should be alerted. The experts can select any salient features for a particular case as they wish.

5.3.4 Coding and GUI

Our system is implemented using Java in all modules, including the graphical user interface. To facilitate network administration, our system implements an attribute of a case as a Java interface; that is, at any time, users can add new attributes to a case by add a new attribute plug-in to the system. One requirement is that a new attribute must be explicitly registered in a configuration file of case attributes.

Figure 5.4 illustrates the main window of our prototype. The main window consists of 3 panels; a graph panel, summary panel, and case panel. A graph of audit traffic is drawn in the graph panel on the top part of the window. A summary of all audit cases is listed on the summary panel, on the bottom left part of the window. To view details of a particular case, a user can click on that case in the summary panel and its details will be shown on the case panel on the bottom right of the window.

As discussed earlier, an expert can choose between profile construction, i.e., to create a new profile for a new pattern, and rule construction for a new justification to correct a misclassified case. Figure 5.5 is a window when an expert chooses to create a new profile. A list of temporal attributes is listed for expert selection.

On the other hand, a window for new rule creation is shown in Figure 5.6. There are more attributes for an expert to choose from. These are all attributes composing a case. An expert can create a rule using any combination of these

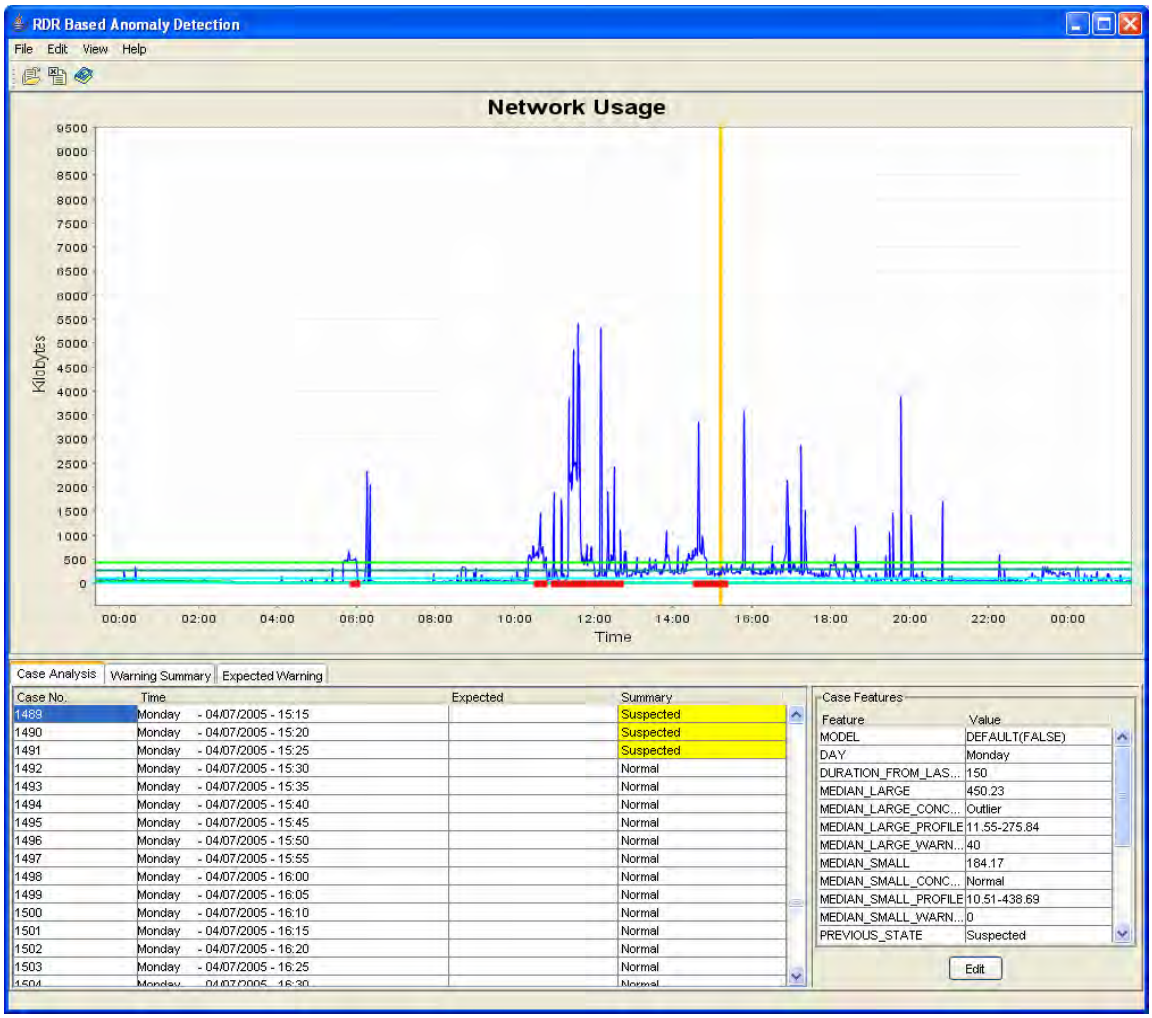


Figure 5.4: GUI of the main window

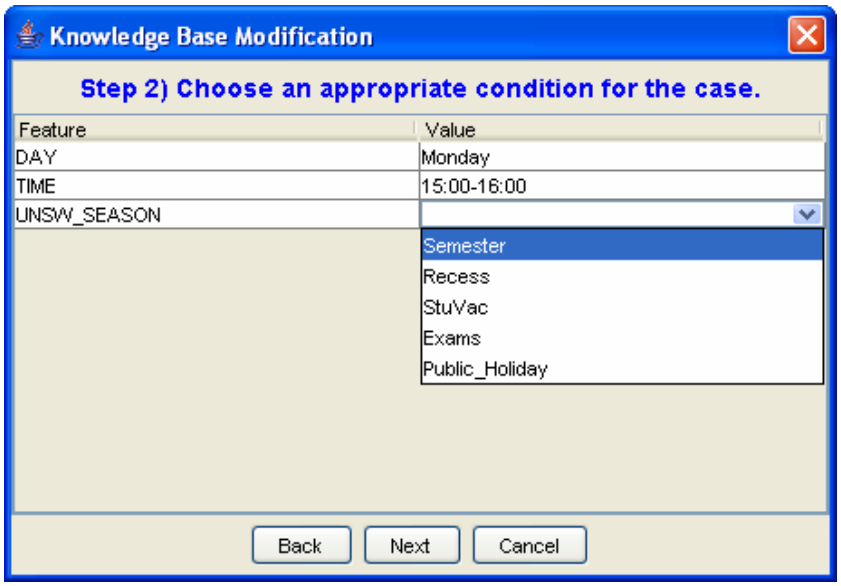


Figure 5.5: GUI of the profile construction Window.

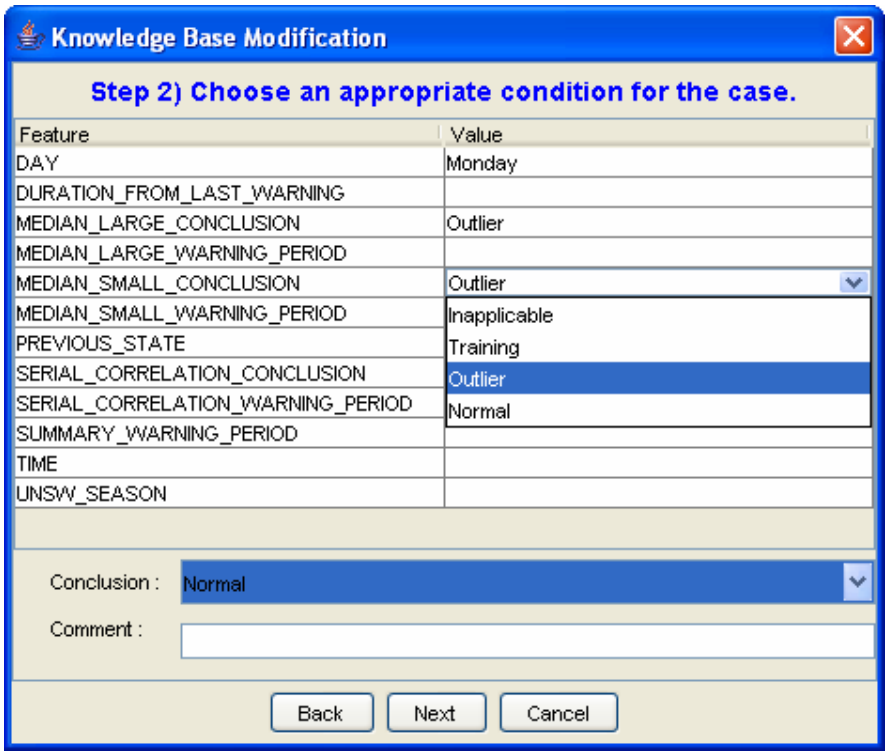


Figure 5.6: GUI of the rule construction window

attributes at will.

We have another additional option to profile and rule construction. That is profile update. This option is provided for the case where an expert judges that the observed pattern should belong to the profile although it was rejected by the OEBA algorithm.

5.4 Traffic Anomaly Detection Experiment

The system was test run on five data sets from RRDtool IP flow archives in CSE. Each archive contains data for seven day traffic with marked anomalies. To evaluate system performance, we use two basic metrics; i.e., false positive rate (FPR) and false negative rate (FNR). The false positive rate is the proportion of normal traffic that were erroneously reported as being invalid or anomalous, where the false negative rate is the proportion of anomalous traffic that was erroneously reported as normal.

5.4.1 Traffic Data Sets

For convenience, we refer to these data series as T_i, where i is 1, ..., 5. These five data series are archived in chronological order, i.e., traffic in T_1 is archived before T_2, T_3 and so on. Another characteristic of these series is these five data series are from different university seasons. This is relevant to the interpretation of the results below. T_1 is archived during the examination period and recess week. T_2 and T_3 are during recess weeks. Series T_4 is archived when a semester just has just started, hence the beginning part of the series still represents recess behavior. T_5 is during semester. For convenience, Table 5.1 summarises details of the university seasons for each data set.

Traffic Series	University Seasons
T_1	Examination
T_2	Recession
T_3	Recession
T_4	Recession + Semester
T_5	Semester

Table 5.1: University seasons of each data set

Anomalous traffic in these five series is labeled by an experienced network administrator in CSE. Traffic graphs were firstly given to the administrator. Suspicious patterns were manually labeled as anomalies. Figure 5.7, 5.8, 5.9, 5.10 and 5.11 illustrate these five traffic series. These data sets are real data taken CSE network. Anomalies are scattered throughout this traffic series; however in T_2, anomalous traffic also occurs at the start of the series. Anomalies in each data set is summarized in Table 5.2.

Traffic	Anomalous Events	Time Span(hr:min)
T_1	1	0:15
T_2	6	6:45
T_3	8	16:05
T_4	3	13:43
T_5	7	20:10

Table 5.2: Anomalies in data sets. This table shows the number of anomalous events and time span in each data set. These anomalies were marked by a CSE network administrator.

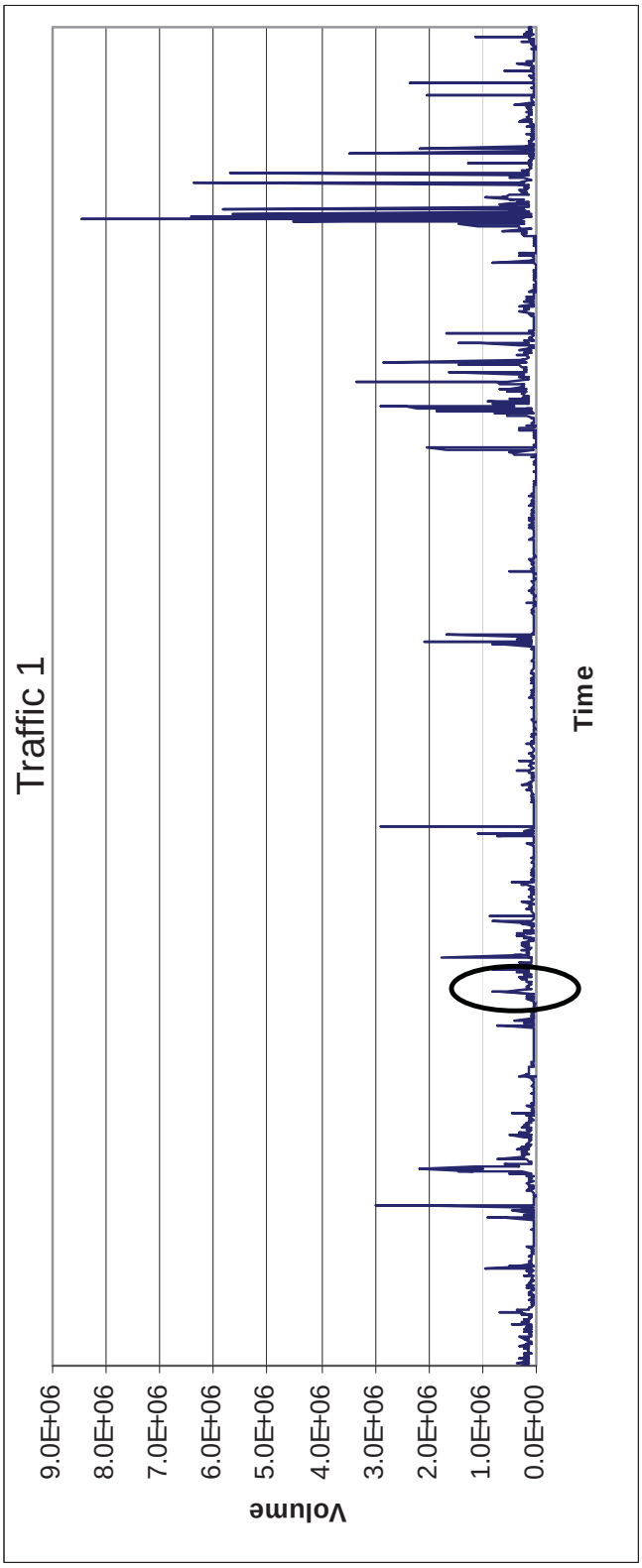


Figure 5.7: Traffic series T_1

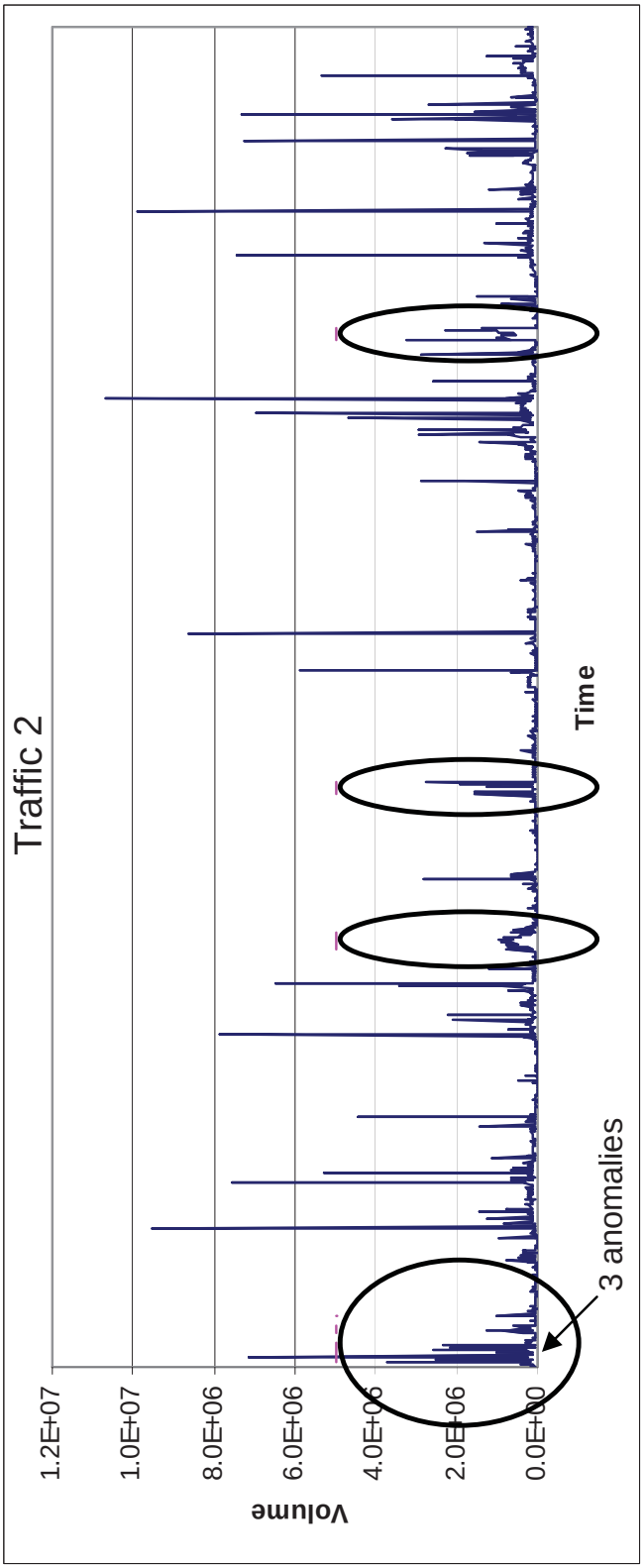


Figure 5.8: Traffic series T_2

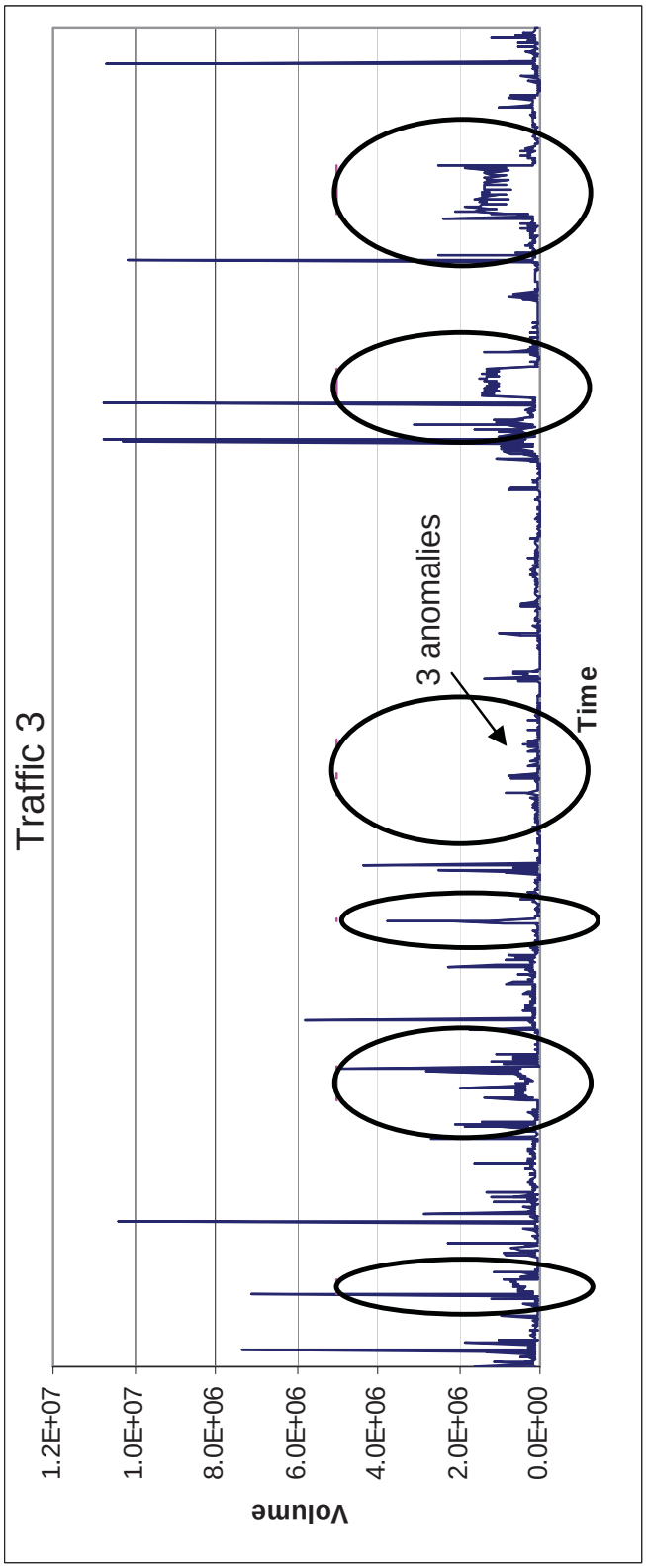


Figure 5.9: Traffic series T_3

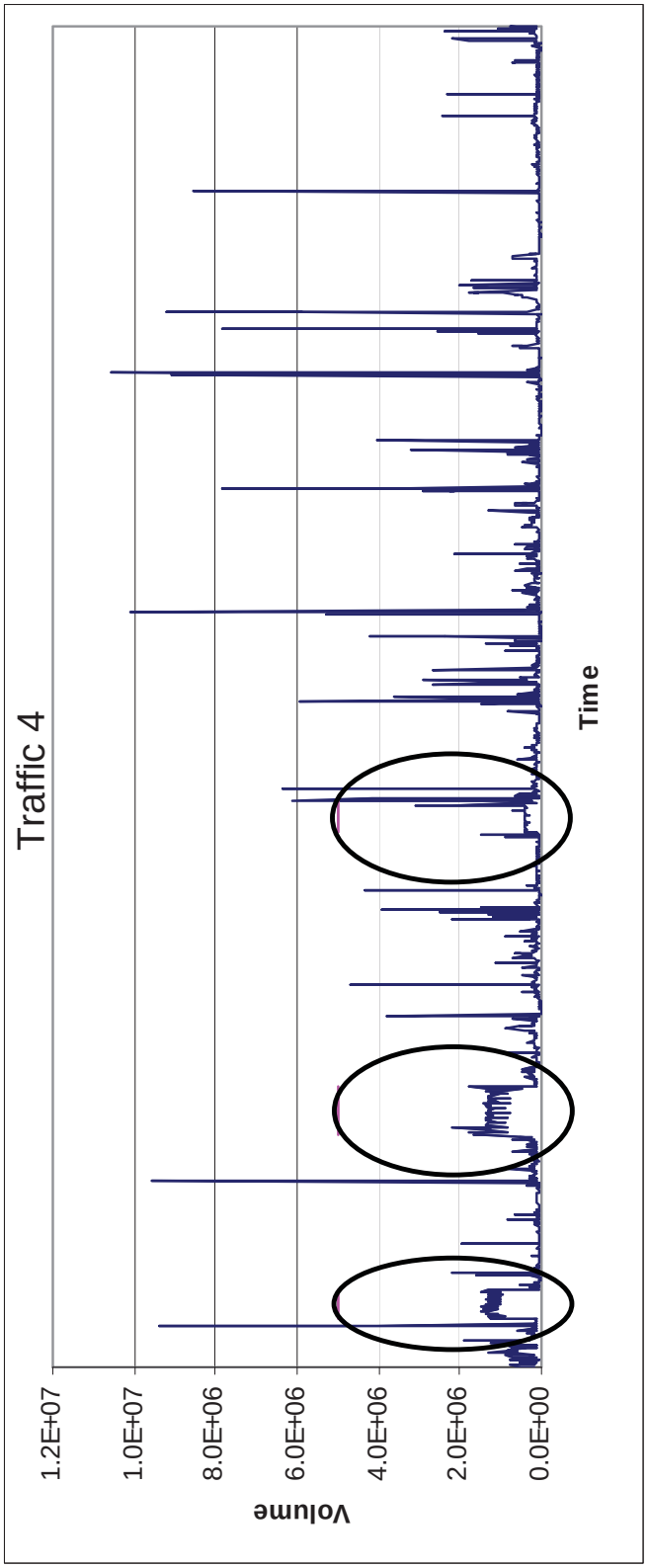


Figure 5.10: Traffic series T_4

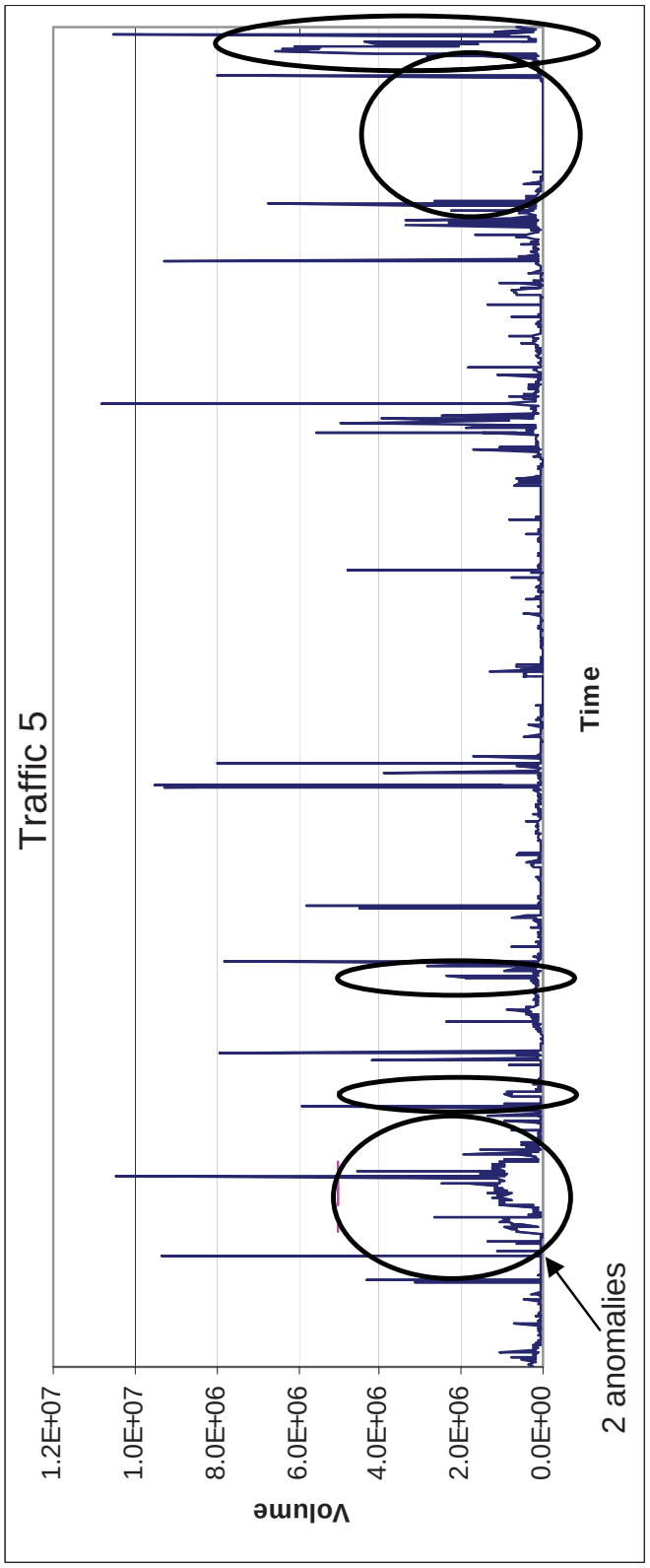


Figure 5.11: Traffic series T_5

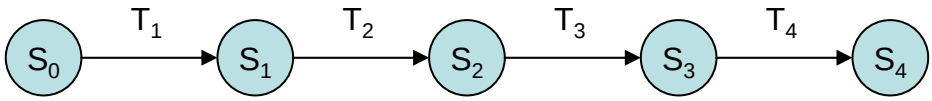


Figure 5.12: System states.

Once again, it is important to note that our approach proposes a method for anomaly detection, but does not include anomaly identification and remedy. Our interest lies on detecting whatever deviates from learnt behavior and expects experts to make a final conclusion on these suspicious patterns.

5.4.2 Experiments

The experiment is set as follows. The system is set to start from 5 different states of knowledge, denoted by S_0, ..., S_4. Figure 5.12 illustrates the transition of each state from S_0 to S_4. For each state, it is trained with different data sets and then tested on each of the remaining T_1,...,T_5 data sets individually. For example, S_3 is trained on T_1, T_2 and T_3. It is then used as a starting knowledge base and tested on T_4. Separately, it is also used as a starting knowledge base and tested on T_5. Table 5.3 summarizes trained data sets and test data sets for each system state.

System State	Trained Data Sets	Test Data Sets
S_0	-	T_1, T_2, T_3, T_4, T_5
S_1	T_1	T_2, T_3, T_4, T_5
S_2	T_1, T_2	T_3, T_4, T_5
S_3	T_1, T_2, T_3	T_4, T_5
S_4	T_1, T_2, T_3, T_4	T_5

Table 5.3: Data sets for training and testing in each system state. There is no trained data set for S_0, that is, it starts from an empty knowledge base. For each state, a data set is tested separately from other test data sets.

We followed this slightly convoluted design to try to investigate the effect of different sequences of cases, e.g., T_5 coming directly after T_1, T_2 and T_3, rather than after T_4. This enables us to explore a wide range of sequences rather than simply considering T_1,...,T_5 in order.

5.4.3 Results And Discussion

Although a warning is produced when a case is detected as anomalous, the expert will not be alerted case by case. The system considers all warnings produced by consecutive cases as a single warning and the expert is alerted once because an anomalous event in general lasts for a period of time, as shown in Table 5.2.

From section 4.4, these two metrics are calculated by

$$\text{false positive rate} = \frac{\text{number of valid instances identified as outliers}}{\text{number of valid instances}}$$

$$\text{false negative rate} = \frac{\text{number of invalid instances not detected as outliers}}{\text{number of invalid instances}}$$

However, in this experiment, we had to modify the calculation of the two rates to reflect how anomalies were defined, as follows.

$$\text{false positive rate} = \frac{\text{time span of incorrect warnings}}{\text{time span of normal traffic}}$$

$$\text{false negative rate} = \frac{\text{time span of anomalous events not detected}}{\text{time span of anomalous events}}$$

Calculated FPR and FNR are shown in the table 5.4. The diagonal in the table shows what would be the learning pattern of a system in routine use, where all previous data (except anomalies) have been used as training data. The other cells in the table represent less training data, taken from earlier periods, while the top row in the table shows each period with no prior training data.

From Table 5.4, the system in state S_0, with no prior training for any period tends to produce the highest rates of both false positives and false negatives. Once the system has learnt traffic T_1 for an examination week, it evolves to state S_1. For S_1, false negative rates are zero for all test data sets. However,only false positive rates for T_2 and T_3 drop, while those for T_4 and T_5 increase. The explanation for this increase is simply that the new profiles created for S_1 did not cover T_4 and T_5. Furthermore, these new profiles remove instances with high volume from the

Data Sets	S_0	S_1	S_2	S_3	S_4
T_1	6.1%				
T_2	2.3%	2.0%			
T_3	4.1%	3.6%	1.6%		
T_4	1.5%	5.3%	3.1%	2.1%	
T_5	7.5%	8.9%	7.2%	7.2%	7.2%

(a) False Positive Rate

Data Sets	S_0	S_1	S_2	S_3	S_4
T_1	0%				
T_2	27.6%	0%			
T_3	39.2%	0%	0%		
T_4	55.7%	0%	0%	0%	
T_5	0%	0%	0%	0%	0%

(b) False Negative Rate

Table 5.4: False positive and false negative rates of each test data sets in different system states. Numbers shown in table are percentages.

default profile as they are now covered by the new profiles. T_1 occurs during an examination week, while T_4 and T_5 involve traffic during a semester, but these patterns have not been defined yet. Although T_2 and T_3 are also different from T_1, their patterns are similar.

In the next state S_2, after patterns of an examination week and a recession week have been learnt, both false positive and false negative rates drop for T_3 and T_4, but not for T_5. This same result also shows in state S_3. These results show that when the system has learnt more data, false rates keep dropping, except for T_5. The explanation for this is that traffic behaviour during semester (T_5) is sufficiently variable that more profiles would need to be added over a longer period.

5.4.4 Building Up Knowledge Bases

As shown in Figure 5.12, the system was started from an empty state and gradually learned over 5 consecutive periods. During a KA session, the expert can choose between creating a new profile in the first knowledge base, adding a new final decision rule to the second knowledge base, and adding a current case to a profile.

The choice of adding a case to a profile is chosen when the case is a false positive to the profile. Table 5.5 shows KA sessions for the first 1,000 cases in a chronological order .

The first KA session was for creating a new profile the traffic of weekdays, during an examination week. The profile was created because the system raised a warning for the case number 12. The second KA session was for adding a false positive case number 86 to the above profile. The next session was on case 149. A new profile was created for traffic between 23:00 and 23:59 on weekdays during an examination week. This profile was an exception to the profile created for case 12. Similarly, the next session on case 295 created a new profile for traffic between 11:00 and 13:00 on weekdays during an examination week. Again, this profile was an exception to the profile created for case 12. A new profile for a new partition was created when case 442 had been observed. Case 442 was during a new season, i.e., UNSW_SEASON=Recess. The process continued like this until traffic was allocated to appropriate profiles.

Table 5.6 shows KA sessions from case 2000 to case 3000 in chronological order. The decision rule for case 2080 suppressed a warning produced from the profile of a window of 60 minutes if other profiles did not raise warnings. This is because a window of 60 minutes is influenced by high volumes in the previous period, i.e., before 21:00. The next rule was created for case 2382. A warning was produced from the serial correlation[1] profile and it lasted for 10 minutes, which the expert believed was too short to pay attention to. The rule suppressed a serial correlation warning if it did not last longer than 10 minutes. Again, the process continued until results from profiles produced correct conclusions.

Figure 5.13 shows the system gradually learning over 8,000 cases. The number of profiles, decision rules and false positive added to profiles for every 1,000 cases are summarized in Table 5.7.

The system is started with one default profile and three default decision rules. The default profile is for any traffic that cannot be allocated to any other profiles.

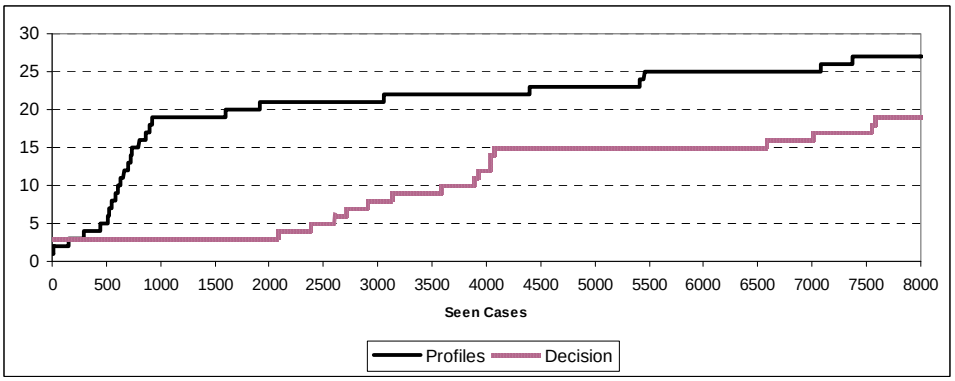
[1]Serial correlation was defined in Section 5.3.2, and is a measure of fluctuation.

Case	Action
12	New Profile: UNSW_SEASON=Exams & DAY=Weekday
86	Add the case to profile
149	New Profile: UNSW_SEASON=Exams & DAY=Weekday & TIME=23:00-23:59
295	New Profile: UNSW_SEASON=Exams & DAY=Weekday & TIME=11:00-13:00
442	New Profile: UNSW_SEASON=Recess & DAY=Weekday
512	New Profile: UNSW_SEASON=Recess & DAY=Weekday & TIME=05:30-07:00
527	New Profile: UNSW_SEASON=Recess & DAY=Weekday & TIME=07:00-09:00
553	New Profile: UNSW_SEASON=Recess & DAY=Weekday & TIME=09:00-11:00
579	New Profile: UNSW_SEASON=Recess & DAY=Weekday & TIME=11:00-14:00
601	Add the case to profile
611	New Profile: UNSW_SEASON=Recess & DAY=Weekday & TIME=14:00-16:00
635	New Profile: UNSW_SEASON=Recess & DAY=Weekday & TIME=16:00-18:00
659	New Profile: UNSW_SEASON=Recess & DAY=Weekday & TIME=18:00-21:00
670	Add the case to profile
695	New Profile: UNSW_SEASON=Recess & DAY=Weekday & TIME=21:00-23:00
726	New Profile: UNSW_SEASON=Recess & DAY=Weekday & TIME=23:00-23:59
730	New Profile: UNSW_SEASON=Recess & DAY=Weekend
799	New Profile: UNSW_SEASON=Recess & DAY=Weekend & TIME=05:30-07:00
858	New Profile: UNSW_SEASON=Recess & DAY=Weekend & TIME=10:00-13:00
861	Add the case to profile
895	New Profile: UNSW_SEASON=Recess & DAY=Weekend & TIME=13:00-16:00
924	New Profile: UNSW_SEASON=Recess & DAY=Weekend & TIME=16:00-18:00
953	Add the case to profile

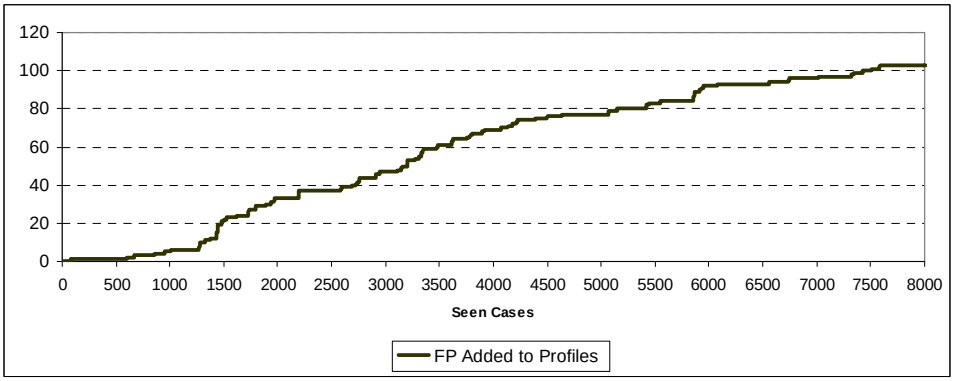
Table 5.5: KA sessions for the first 1,000 cases.

Case	Action
2080	New Rule: *Normal* if UNSW_SEASON=Recess & TIME=21:00-23:00 & SERIAL_CORRELATION_CONCLUSION=Normal & MEDIAN_LARGE_CONCLUSION=Outlier & MEDIAN_SMALL_CONCLUSION=Normal
2191	Add the case to profile
2192	Add the case to profile
2193	Add the case to profile
2196	Add the case to profile
2382	New Rule: *Normal* if UNSW_SEASON=Recess & TIME=22:30-23:59 & DAY=Weekday & MEDIAN_LARGE_CONCLUSION=Normal & SERIAL_CORRELATION_CONCLUSION=Outlier & SERIAL_CORRELATION_WARNING_PERIOD=10 & MEDIAN_SMALL_CONCLUSION=Normal
2585	Add the case to profile
2594	Add the case to profile
2595	New Rule: *Normal* if UNSW_SEASON=Recess & TIME=16:00-17:00 & DAY=Weekday & MEDIAN_SMALL_WARNING_PERIOD=10 & MEDIAN_LARGE_CONCLUSION=Normal & MEDIAN_SMALL_CONCLUSION=Outlier
2683	Add the case to profile
2707	New Rule: *Normal* if UNSW_SEASON=Recess & TIME=02:00-04:00 & MEDIAN_SMALL_CONCLUSION=Normal & MEDIAN_LARGE_CONCLUSION=Normal & SERIAL_CORRELATION_CONCLUSION=Outlier & SERIAL_CORRELATION_WARNING_PERIOD=15
2728	Add the case to profile
2740	Add the case to profile
2757	Add the case to profile
2761	Add the case to profile
2900	New Rule: *Normal* if MEDIAN_LARGE_CONCLUSION=Outlier & MEDIAN_SMALL_CONCLUSION=Normal& UNSW_SEASON=Recess & DAY=Weekend & SERIAL_CORRELATION_WARNING_PERIOD=10
2907	Add the case to profile
2909	Add the case to profile
2945	Add the case to profile

Table 5.6: KA sessions from case 2000 to case 3000.



(a) The number of profiles and decision rules in the two KBS.



(b) The number of FPs added to profiles.

Figure 5.13: The number of profiles, decision rules in the two KBS, and the number of false positive cases added to profiles against seen cases.

Cases	Profiles	Decision Rules	FP Added to Profiles
0	1	3	0
1000	19	3	5
2000	21	3	33
3000	21	8	47
4000	22	12	69
5000	23	15	77
6000	25	15	92
7000	25	16	96
8000	27	19	103

Table 5.7: The number of profiles, decision rules and profile updates for every 1,000 cases.

In contrast to other RDR frameworks, the final decision is initialized with three default rules. The first rule is a typical default rule: *IF (true) THEN (a case is suspected)*. The other two rules summarize results derived from profiles as follows. First, the first 3 cases since a profile has been created are concluded as *training*. Second, if all results from profiles are *normal*, a case is concluded as *normal*.

For the first 2,000 cases, new rules were added to the profile KB only. This is because the system was started when the profile KB was empty and warnings were produced by the default profiles. The expert then created appropriate profiles for traffic that was warned about. When it was further trained, i.e., for the next 2,000 cases, more rules were added to the final decision KB. This means profiles were appropriately created for traffic events but their results needed some tuning by rules that were added to the final decision KB. The graph in Figure 5.13 shows that the number of new rules added to both KBs are reduced when the system has progressed.

However, most of KA sessions were false positives which were caused when profiles were just created and the number of seen cases was small. When profiles had seen enough number of cases (i.e., > 5), the problem was eliminated.

In summary, from 8064 cases, 26 sessions were required for adding new profiles, 16 sessions for adding new final decisions, and 103 sessions for adding false positive cases to profiles. Although most KA sessions were false positives, there were in total 145 KA sessions for 8064 cases, which means that, on average, the system

was required to consult the expert every 55.6 cases or about every 4.38 hours. In other words, there were only 5.18 sessions required a day, which was very little. The graph also suggested that the more cases the system was trained, the fewer KA sessions were required for adding new rules to the two knowledge bases and the fewer false positive cases.

5.4.5 A Comparison with Another Method

Our technique has shown that it is capable of detecting outliers and alerting the expert when new patterns have been observed. However, the main contribution of our technique is that it is simple to add profiles for new patterns when required while in use, which is not possible in other approaches. Although, it is not our intention to surpass the accuracy of other systems, a comparison between our technique and the classic Holt-Winters (HW) algorithm using our data is provided.

In Holt-Winter algorithm, three parameters, i.e., α, β, γ, must be estimated before testing. Using traffic T_3, the three parameters were estimated as follows: $\alpha = 0.546, \beta = 1.0e-14, \gamma = 0.009$. The algorithm was then used to detect outliers from the same series, T_3. The reason that we ran the algorithm on the same series was to give the Holt-Winters method a better chance of success by avoiding bias from different seasons. The Holt-Winters algorithm could only detect 1 out of 8 anomalies in T_3 and produced one false positive warning. Although many systems aim to reduce false positives as much as possible, false negatives are unwanted. There is no point in training and testing our system on T_3 as it will necessarily be perfect so we compared our results on T_3 as an unseen test set with the system trained on T_1 and T_2. From Table 5.4, all 8 anomalies could be detected, with a false positive rate of 1.6%. We did not undertake more comprehensive comparisons with HW or other algorithms because as noted our aim was to produce a system that could easily learn any domain and adapt to further changes in the domain, while producing a good level of accuracy.

5.5 Conclusion

In this chapter, we have presented a Ripple Down Models (RDM) framework for network traffic anomaly detection. The RDM approach facilitates learning new traffic patterns of a network. Each pattern is learned by the algorithm: Outlier Estimation with Backward Adaptation (OEBA). This algorithm is able to learn system behavior from a blind state, adapting itself as more objects are observed and is intended to be robust at the initial stages of learning when relatively little data has been available. The results demonstrate the utility of partitioning data into more homogeneous sub-regions, rather than trying to learn a single model. Within the framework, another knowledge base is used to decide if outliers do in fact represent anomalous behavior.

We demonstrated the system using real Internet traffic data archived by the RRDtool. In the implementation, we have used median and variance of traffic. However, in the framework, network administrators could select other features to be monitored and modeled. The system could also be extended to deal with information from inside packet headers etc.

Chapter 6

Prudent Knowledge Bases

6.1 Introduction

Every expert system is aimed at possessing all the expertise from human experts in a particular domain. If we imagine all the world's knowledge as a tree, where general knowledge at the top and specific knowledge at the bottom, then an expert system covers only a small part or just a leaf of this tree(Lenat, 2002). Expert systems are brittle because they do not realise the limits of their own knowledge. For example, an expert system with a very high level of knowledge about chemical pathology results might still diagnose a male as pregnant who has high levels of a pregnancy hormone, because of a hormone-secreting tumour, because no one ever thought to tell it that only females get pregnant. The CYC project (Guha and Lenat, 1990) is an attempt at a solution to this problem by building a knowledge base of common sense, or general knowledge at the top the tree, as a foundation on which other expert systems could be built. A variety of applications have used the CYC knowledge base, for example, in directed marketing and database cleansing(Lenat, 1994).

Brittleness can be seen as a lack of common sense knowledge, but it can also be characterised as a failure of the expert system to recognise when a case is outside its range of experience. To build a complete knowledge base that contains all possible knowledge is not easy as some data patterns may never occur in practice and expert

justification is quite speculative when judging data patterns outside the expert's experience (Compton and Jansen, 1990; Compton et al., 1996).

One attempt to address the brittleness of expert systems is a technique called "prudence" in the Ripple Down Rule paradigm (Edwards et al., 1995; Compton et al., 1996). In this work, the system would warn of new types of cases for which a new rule may have to be added. This approach associated profiles of attributes of seen cases with each rule. There were two types of profiles, i.e., a list of seen values for a categorical attribute and a pair of upper and lower bounds for a continuous attribute. The profile was used when a case was evaluated against the rule; that is if a new value or a value outside the range occurred, a warning was raised. The approach worked well, but the false positive rate was about 15%, because of the simple way in which cases were compared to profiles.

This previous work is extended by model-based prudence (Prayote and Compton, 2006). Two models are studied, one for continuous attributes, the other for categorical attributes. For continuous attributes, we apply the probabilistic outlier detection technique, OEBA as discussed in Chapter 4. By doing this, the profile is able to understand the attribute better. For example, it can accept a case where a value of an attribute falls insignificantly outside the range, if the OEBA believes that this new value is not a real outlier. This is the same problem occuring in intrusion detection domain discussed in Chapter 5. A simulation is presented and the result is discussed in Section 6.3.

Furthermore, we develop another statistic technique based on the Negative Bernoulli distribution to accommodate categorical attributes known as Outlier Estimation for Categorical Attribute (OECA). This algorithm is explained in Section 6.4. Simulations are conducted to show improvement in performance when this new algorithm is used with OEBA.

We consider each attribute independently. As will be seen, this works well even when attributes are related because although relationships change across partitions, within a partition the relationship is less important. However, we also show in Section 6.6 that if the correlations are important, our approach can easily accommodate them.

Finally, we present a study on warnings produced by probabilistic profiles in Section 6.7. The study is aimed to distinguish correctly warned cases from false positives.

6.2 A Review on RDR with Prudence

Ripple Down Rules (Compton and Jansen, 1988; Compton et al., 1989)is an incremental approach in knowledge acquisition. As reviewed in Chapter RDR, it has been used extensively in a variety of tasks including classification (Compton and Jansen, 1988; Compton et al., 1989; Compton and Jansen, 1990; Preston, Edwards, and Compton, 1994; Compton et al., 2006; Edwards et al., 1993), resource allocation (Richards and Compton, 1999), configuration (Mulholland et al., 1993; Compton et al., 1993), heuristic search (Bekmann and Hoffmann, 2005), and image processing (Park, Wilson, and Jin, 2000; Misra, Sowmya, and Compton, 2004). Its first and major success is classification tasks, especially medical systems.

The approach adds a new rule to a knowledge base by means of correcting an incorrectly classified case. Initially, these misclassified cases were identified by manual inspection, which is a time consuming process. Compton et al. explained the situation where a case is incorrectly classified as the lack of recognition of reaching the limits of the system's knowledge. They attempted to address this problem by a technique called *Prudence*. Prudence was implemented by means of associating profiles of attributes of seen cases with each rule. When a case was evaluated against the rule, if a new value occurred, a warning was raised. The approach was simulated on three data sets, i.e., Garvan, Chess and Tic-Tac-Toe. A discussion on this study is presented below.

6.2.1 Audit Data Sets

The earlier work on prudent RDR in (Compton et al., 1996) was applied to three data sets from the UC Irvine Machine Learning Repository, i.e., Garvan, Chess and Tic-Tac-Toe data set.

Garvan

Garvan data set is an archive of thyroid diagnoses obtained from the Garvan Institute. There are several versions of these data. A version available on UC Irvine data repository consists of 9172 records from 1984 to early 1987. Each record looks like:

(29 attribute values), diagnoses [record identification]

The attributes are given in order and separated by commas. Unknown attribute values are indicated by question marks. For example, a record of a diagnosis is

41,F,f,f,f,f,f,f,f,f,t,f,f,f,f,f,f,f,?,f,?,f,?,f,?,f,?,t,11,other,-[84080104 2].

Among those 29 attributes, there are 7 continuous attributes and 22 categorical attributes.

However, the previous study in (Compton et al., 1996) used another version of Garvan data set. Cases in this set are part of a larger data set of 45,000 cases and a sequence of 21,822 consecutive records was chosen where the types of cases did not appear to be changing over time (Gaines and Compton, 1995). Induct was the ML in these studies. As Induct does not handle numbers, a set of derived categorical values was used, but warnings were based on the numerical values. In the present study, we used J4.8 and so used the numerical attributes as well as the non-numerical categorical attributes.

Chess

The Chess data set is from UC Irvine data repository. The data set contains 3196 cases of different scenarios of a White king and a White rook versus a Black king and a Black pawn on a7 of a chess board, i.e., the Black pawn is one square away from queening, when it is the White's turn to move. The target concept is whether the white can win this game, i.e., there are two classifications of "won" or "nowin". An assumption of wining in this game is that the White is deemed to be unable to

win if the Black pawn can safely advance (queening). A scenario for the board is described by 36 attributes. For example,

f,f,f,f,f,f,f,f,f,f,f,f,l,f,n,f,f,t,f,f,f,f,f,f,f,f,t,f,f,f,f,f,f,f,t,t,n,won

describes a winning scenario of the White.

Tic Tac Toe

This data set encodes the complete set of 958 possible board configurations at the end of tic-tac-toe games, where "x" is assumed to have played first. The target concept is whether x wins for a particular configuration, i.e., when "x" has one of 8 possible ways to create a "three-in-a-row". Hence, there are 9 attributes of each position on the board with a conclusion of the target concept. For each attribute, there are 3 possible values, i.e.,

- x. "x" means that square is taken by the player x.
- o. "o" means it is taken by another player o.
- b. "b" means it is not taken by any player.

There are 2 possible values for the target concept, i.e.,

- positive. A configuration is "positive" if the player x wins.
- negative. A configuration is "negative" if the player x cannot win.

For example, one configuration looks like this:

x,x,x,x,o,o,o,b,b,positive.

6.2.2 Simulation

To evaluate the Prudence mechanism, four metrics are used.

- False Positive (FP). A case is a false positive if a warning is produced unnecessarily, i.e., when the case is correctly classified.
- False Negative (FN). A case is a false negative if a warning is not produced when required, i.e., when the case is incorrectly classified.
- True Positive (TP). A case is a true positive if a warning is produced properly, i.e., when the case is incorrectly classified.
- True Negative (TN). A case is a true negative if a warning is not produced unnecessarily, i.e., when the case is correctly classified.

A prudence mechanism would be acceptable if it produced low false rates and high true rates. To add rules, a simulated expert is used instead of a human expert. A simulated expert is actually another expert system which is tuned to perform acceptably.

In this work, Compton et al. used Induct/RDR machine learning algorithm (Gaines, 1991) to construct a knowledge base for each data set. This knowledge base is used as the simulated expert for the data set. An RDR-ES with prudence is constructed as follows. Each case is run on the RDR-ES with prudence and assessed to see if a warning should be generated. In parallel, the case is run on the simulated expert and the warning assessed as true positive, true negative, false positive or false negative depending on whether there was a warning and whether the conclusions agree. If the conclusions disagree, a new rule is added to the constructing ES. The rule consists of conditions from the rule trace of the simulated expert that gave the correct conclusion and other features from the case. Different levels of expertise are simulated by different combinations of conditions from the rule trace and other features from the case. The next case is then run and the process repeated.

6.2.3 Previous Results

In this previous study, there were two types of profiles, i.e., a list of seen values for a categorical attribute and a pair of upper and lower bounds for a continuous attribute. These profiles were used when a case was evaluated against the rule.

Warnings were raised when unseen values were observed; that is, a new value for a categorical profile or an out-of-range value for a continuous profile.

Data Set	FN%	TP%	TN%	FP%
Garvan	0.2	2.4	83	15
Chess	0.3	1.3	91	7
Tic Tac Toe	1.5	3.8	81	14

Table 6.1: Final results of simulation of RDR with prudence in (Compton et al., 1996).

Results were summarized as shown in Table 6.1. This table shows the total number of all metrics for the whole data set shown as a percentage of the data set. With the developing RDR system, 2.6% of Garvan cases, 1.6% of Chess and 5.3% of Tic Tac Toe required rules to be added because the developing KB had not given the correct conclusion.

The goal of the study was to correctly warn about all errors. However, this goal was not reached. In the Garvan data set, warnings were missed for 8% of the cases that were incorrectly classified and required a further rule to be added. Missed warnings were higher in the other two data sets; i.e., 19% in Chess and 28% in Tic Tac Toe. Furthermore, there were quite high volumes of unnecessary warnings produced from those profiles. About 15% of all cases were incorrectly warned about in Garvan data set, and 14% in Tic Tac Toe. The unnecessary warnings were lower at 7% in Chess, though.

Although the technique did not reach its goal of efficiently warning about all errors being made in RDR, it demonstrated a reasonable approach to preventing brittleness in RDR KBS. The following are reasons why the technique could not reach its ultimate goal.

The only reason a warning could be missed is because attributes were not independent. So although individual values have been seen before, the combination is occurring for the first time. As will be seen, the false negative rate in our study is lower than the previous study. We suspect that in the previous study all cases were used, with the conclusion given by the simulated expert taken to be correct by definition. We excluded such cases and only considered cases where the simulated

expert was able to give the correct conclusion in the database. There are differences between the various simulated experts and the original conclusions because of the different ways in which pruning is done.

The second problem is false positives or unnecessary warnings. A warning was produced when a profile detected an unseen value. These profiles were implemented in a simple and easy to manage way, but were too simple. It is our intention to reduce the false positives using a probabilistic approach.

6.3 Probabilistic Profile for Continuous Attribute

As discussed in Chapter 4, the algorithm OEBA is designed for continuous attributes. It can separately and efficiently learn a model for a continuous attribute. For any new value that is out of the maintained range of a model, the algorithm computes the Range Probability for n objects, which is a simple probability of including this new value into the model, as follows.

$$RP^n = (\frac{b' - a'}{b - a})^n \tag{6.1}$$

The previous simulation of the algorithm OEBA was applied to randomly generated data sets to study the effect of different thresholds. Data were generated following two distributions; i.e., a standard uniform distribution and a standard normal distribution. It demonstrated promising results in applying the algorithm to practical outlier detection.

In this section, we study the OEBA performance against earlier work on Prudence, using the Garvan data set. We have not considered the Tic Tac Toe or Chess end game examples as they do not include numerical data. There are two types of attributes in the set, i.e., continuous and categorical. As our interest initially is on continuous attributes, we only enhance probabilistic profiles for continuous attributes. The original logging technique without probability is still used for categorical attributes.

6.3.1 Data Set

As mentioned earlier, there are a number of versions of the Garvan data set. In this simulation, we used a smaller sub set of 20470 cases from a larger set of 43472 cases. Previous studies (Compton et al., 1996; **?**) have suggested the types of cases seen changed over time and the 20470 cases represent a more homogeneous period. There are 8 continuous attributes and 22 categorical attributes including a class attribute. The class attribute has 60 classifications relating to Thyroid diagnosis. The extra continuous attribute compared to the UC Irvine data is because of an extra measured variable in later data.

We have noted missing values in many attributes and in many cases. The approach to missing values in most machine learning is to assume a default value. However, some of the Garvan conclusions relate specifically to missing values, so we added boolean attributes for missing values. These new attributes are flags for missing values.

6.3.2 Simulation of OEBA-based Prudence

Following the previous Prudence work, we used a simulated expert. We used a machine learning technique J48 as implemented by Weka (Witten and Frank, 2005) from the university of Waikoto, NZ, to construct a knowledge-based system for the simulated expert. Each case is evaluated against the simulated expert and an RDR ES under construction. If two conclusions agree, the simulation proceeds to the next case. Otherwise, a new rule is added to the RDR ES under construction. A new rule is derived from the inference trace of the simulated expert. Unlike the previous studies, we considered only the best expert, that is all the conditions in the rule trace.

To evaluate prudence performance, simulations were conducted for two prudence techniques, i.e., the original with simple profiles in (Compton et al., 1996) and the other with probabilistic profiles. The algorithm OEBA relies on a threshold. Following simulations in Chapter 4, we varied the threshold from 1.0E-10 down to 1.0E-90.

We used the same system of recording different types of warnings as mentioned in Section 6.2.2. Algorithm 3 shows how warnings are recorded. It is worth noting that for any case, if the simulated expert cannot give a correct conclusion, i.e., the conclusion stored in the data base, that case is ignored and not recorded in any of these four categories.

Algorithm 3 Recording false positives, false negatives, true positives, true negatives

```
if conclusion is correct then
  if warning is flagged then
    a case is false positive
  else
    a case is true negative
  end if
else
  if warning is flagged then
    a case is true positive
  else
    a case is false negative
  end if
end if
```

6.3.3 Results and Discussion

The results are shown in Table 6.2. From 20470 cases, the simulated expert could give correct conclusions to 20171 cases. The rest were not taken into account. This was simply to avoid any confusion in trying to take into account errors the expert might make. Our focus was to assess the prudence technique in the absence of complicating factors such as expert error.

From 20171 cases, the developing RDR gave 275 wrong conclusions, or 1.36% of all cases. 99.27% of these cases with incorrect conclusions (273 cases) were detected or warned about under the original Prudence technique, as used here. There were only 2 missed warnings. Similarly, OEBA-based prudence warned about 272 cases or 98.9% of 275 mis-classified cases. There were 3 missed warnings. The reason for the two missed warnings will be discussed later, but it can be noted now that only one was due to a correlation. Considering that the biochemical parameters in Thyroid disease are highly related, both inversely and directly, this is a very good

Continuous Profile	FP	FN	TP	TN
Original Prudence	2841	2	273	17055
OEBA 1E-10	1933	3	272	17963
OEBA 1E-20	1877	3	272	18019
OEBA 1E-30	1868	3	272	18028
OEBA 1E-40	1861	3	272	18035
OEBA 1E-50	1858	3	272	18038
OEBA 1E-60	1857	3	272	18039
OEBA 1E-70	1853	3	272	18043
OEBA 1E-80	1851	3	272	18045
OEBA 1E-90	1850	3	272	18046

Table 6.2: A comparison between original prudence and OEBA-based prudence on continuous attributes. Threshold of the algorithm was varied from 1.0E-10 down to 1.0E-90. Categorical attributes were maintained with the original prudence approach.

result. It supports the assumption we have made that RDR partitions the space into relatively homogeneous regions and correlations mainly relate to differences in classification across regions. However, in Section 6.6, we investigate extending the technique to deal with correlated variables. The explanation for the extra case that was detected by the original prudence, but not by the OEBA-based prudence is that an OEBA profile must have included an outlier into its model; hence, it failed to flag this case.

In the previous study, the false positive rate was 15%. In this simulation, the original prudence produced unnecessary warnings for about 14.08% of all cases. These results differ from the original study as we used a different learning algorithm for the simulated expert and removed cases that the simulated expert could not correctly classify. However, what we are concerned with here is to reduce the false positive rate when applying the OEBA-based profile. As shown in Table 6.2, the simulation demonstrated that the number of false positive cases was reduced when the threshold got smaller. However, the rate of reduction was also reduced when threshold got smaller, i.e., the number of false positive cases converged to 1850 or about 9.2% of all cases, while making 3 false negatives.

Although the number of false positive cases was reduced, the remaining rate was still not satisfactory. We have investigated further where the warnings are coming

from and the simulation was modified to record more information regarding sources of warnings. Table 6.3 shows the results.

	Total	Categorical Only	Continuous Only	Both
Original Prudence	2841	868	992	981
OEBA 1E-10	1933	1844	84	5
OEBA 1E-20	1877	1848	28	1
OEBA 1E-30	1868	1849	19	0
OEBA 1E-40	1861	1849	12	0
OEBA 1E-50	1858	1849	9	0
OEBA 1E-60	1857	1849	8	0
OEBA 1E-70	1853	1849	4	0
OEBA 1E-80	1851	1849	2	0
OEBA 1E-90	1850	1849	1	0

Table 6.3: Sources of incorrect warnings. This table shows the number of unnecessary warnings produced from different sources, i.e., categorical profiles, continuous profiles or from both profiles. The total number of false positive cases is also given.

The result is very interesting in that most warnings, 1849 from 1868 or about 98.98% at $T = 1E - 30$, or 1849 from 1850 or about 99.95% at $T = 1E - 90$, produced in the OEBA-based prudence were from categorical attributes. There were only 19 false positive warnings raised from the algorithm OEBA for continuous attributes at $T = 1E-30$ and 1 false positive warning at $T = 1E-90$. This result agrees with the reduced number of false positive cases in Table 6.2, that is the OEBA-based profile greatly reduced the false positive cases from that produced by the original approach. This improvement reveals the underlying problem of the high false positive rate for warnings from categorical attributes. In the next section, an algorithm to learn categorical attributes is proposed.

It should be noted that we have followed the protocol of the original experiment where all errors were corrected as they occured. This means that a false negative case will have rule added to correct it although no warning is raised. So the false negative cases here are rather the number of types of case that were missed rather than the total missed.

We have redone the experiments to count the actual FN when rules are *not* added to correct such errors. The number of FN increases from three to ten cases. That is, there are ten instances of the three types of case, giving an FN rate over the

20K cases of 0.05%. There is also a negligible decrease in the FP rate because rules were not added that might have increased the FP rate. Note that false positives are always the total number of actual cases that were incorrectly warned about.

6.4 Probabilistic Profile for Categorical Attribute

For categorical attributes, we have a similar aim, that is to develop a simple algorithm to learn profiles and detect outliers with a simple probability measure. The situation is similar, except that attributes are discrete. For example, patient samples in the Garvan data set came from 5 different hospitals. A new sample, e.g., the 100^{th}, is coming from a new hospital. What is the probability of a new hospital not being seen until the 100^{th} sample?

For convenience, we denote this algorithm Outlier Estimation for Categorical Attribute (OECA). We have made two assumptions in the algorithm OECA as follows.

1. An attribute is independent. This means the occurrence of all its values is independent from other attributes.

2. An attribute follows a uniform distribution. That is the probability of each value is equal, regardless of the number of occurrences for each value. The probability mass function of a uniform distribution is

$$f(x) = \frac{1}{v} \tag{6.2}$$

   v is the number of all different seen values.

A Bernoulli process is a sequence of independent identically distributed Bernoulli trials. There are only two possible values for each trial, which are generally labelled A and B. From a negative binomial distribution which is based on a Bernoulli process, the probability of k A's, $r-1$ B's in $k+r-1$ trials and that the $(k+r)^{th}$ trial is B is

$$f(x) = \frac{(k+r-1)!}{k!(r-1)!} p^r (1-p)^k \tag{6.3}$$

where p is the probability of the occurrence of B. A specialization of a negative binomial distribution, called a geometric distribution, is a case where $r = 1$, i.e., a series of A's has taken place before a first B occurs, and

$$f(x) = p(1-p)^k \tag{6.4}$$

From our assumptions and Equation 6.4, the probability of a new value B in a Bernoulli trial has not been seen until the $(k+1)^{th}$ trial, where the first k trials are A, is

$$f(x) = \frac{1}{v+1}\left(1 - \frac{1}{v+1}\right)^k \tag{6.5}$$

6.4.1 Outlier Estimation for Categorical Attributes

When a case is matched against a profile, we need to know how well it fits in the profile. Herein, M is denoted as a measure specifying a degree of matching between a case and a profile. When a value has been maintained, M should be 1.0. When a value is new to a profile, M should give a probability of having this new value after v values have been seen from k observations; i.e.,the probability in the equation 6.5 is used in this case. That is,

$$M(x) = \begin{cases} 1.0 & \text{if x has been seen,} \\ \dfrac{1}{v+1}\left(1 - \dfrac{1}{v+1}\right)^k & \text{otherwise} \end{cases} \tag{6.6}$$

However, the probability in Equation 6.5 very much depends on the value of k, i.e., when k is larger, the probability gets smaller. $M(x)$ varies greatly with k, so it is difficult to specify a simple threshold. Secondly, it seems reasonable to suggest that a new value after a long gap since the last new value was observed is more likely to be an outlier than the latest new value when new values are a regular occurrence.

To capture this, the algorithm compares the probability of a new value with the probability of a value that was last accepted into a profile. A ratio between

two probabilities gives a relative difference between them and reduces the effect of large k.

$$NewValueRatio = \frac{M}{M_{accepted}} \tag{6.7}$$

where M is the probability of a current value and $M_{accepted}$ is the probability of a value that was last updated into the profile. From the definition 6.6 , M is always less than or equal to $M_{accepted}$. The more these two probabilities are different, the less likely the case belongs to the profile.

For convenience, our algorithm to maintain categorical attributes is called Outlier Estimation for Categorical Attributes or OECA. The algorithm relies on $NewValueRatio$ to justify whether a case is an outlier. The higher $NewValueRatio$, the less the likelihood that a new value is an outlier. Similar to the algorithm OEBA, a threshold must be predefined. A warning is raised if $NewValueRatio$ is below the threshold. The algorithm OECA is shown in Algorithm 4.

Algorithm 4 Outlier Estimation for Categorical Attributes

```
x: an observation
T: a predefined threshold
if M(x) = 1.0 then
   ACCEPT x
else if M(x) / M_accepted > T then
   ACCEPT x
else
   REPORT outlier
end if
```

6.4.2 Simulation on OECA-based Prudence

To evaluate performance of the algorithm OECA, we ran simulations on the same Garvan data set as used in the previous section. Simulations were conducted by the same procedure, i.e., passing a case through both simulated expert and constructing ES to derive conclusions, and comparing two conclusions for further actions. Similarly, four types of cases were still recorded, i.e., false positives (FP), false negatives (FN), true positives (TP), true negatives (TN). To study the performance of the algorithm OECA, the algorithm OEBA was fixed at the threshold of 1.0E-50,

while varying the threshold of OECA.

Threshold	FP	FN	TP	TN
Original Log	1858	3	272	18038
9E-01	1529	3	272	18367
8E-01	1392	3	272	18504
7E-01	1260	3	272	18636
6E-01	1179	3	272	18717
5E-01	1080	4	271	18816
4E-01	1029	5	270	18867
3E-01	971	5	270	18925
2E-01	711	8	267	19185
1E-01	590	13	262	19306
1E-10	124	19	256	19772
No Profile	9	19	256	19887

Table 6.4: Simulations on categorical attributes. This table shows results when different profiles for categorical attributes are maintained. Continuous attributes are maintained by the OEBA algorithm at the threshold of 1.0e-50. First, they are maintained with a pure log in the original prudence approach. Second, the algorithm OECA is used with different thresholds. The last row shows a result when no profile of categorical attributes was maintained. The numbers shown in the table are the number of false positive(FP) cases, false negative (FN) cases, true positive (TP) cases, and true negative(TN) cases for each simulation.

Table 6.4 shows results of simulating different profiles for categorical attributes, while continuous attributes were maintained by OEBA-based profiles. In the first row, categorical attributes were maintained with pure logs as in the original approach (i.e., any unseen value would be warned about). There were 1858 false positive cases and 3 false negative cases. The algorithm OECA was applied to categorical attributes with various thresholds and results are shown in the next rows. Simulations were started off from a threshold of 9.0E-1 which produced 1529 false positives, i.e., false positive cases were reduced by 329. When the threshold was set lower, a number of false positive cases dropped constantly without an increase in false negatives.

However, when threshold was 5.0E-1, an extra false negative case occurred. With lower thresholds, the algorithm started to introduce extra false negative cases. To avoid false negatives, the simulation worked best at threshold 6.0E-1, where there were only 1179 false positive cases or 5.85% of all cases. A graph in Figure 6.1 shows relationship of false positives and false negatives for various thresholds.

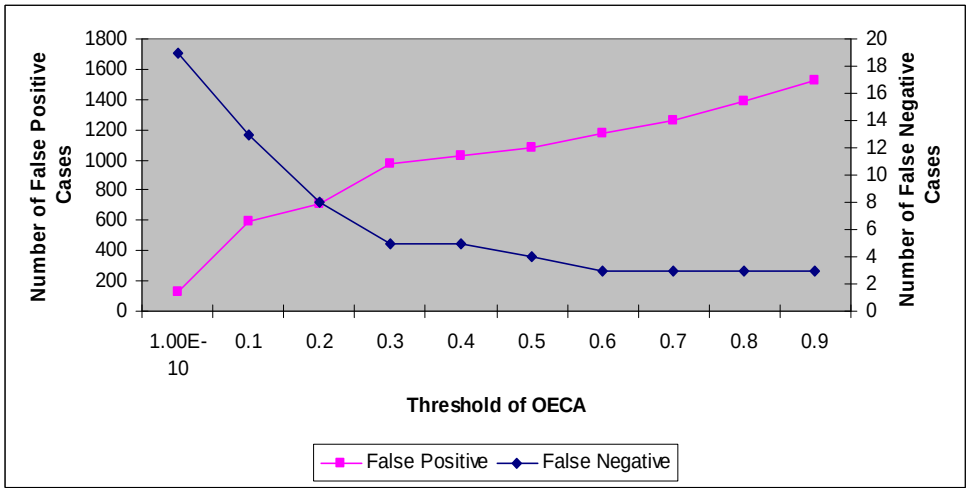


Figure 6.1: False positives and false negatives produced from different thresholds.

A simulation was also conducted for a situation where no profile was maintained for categorical attributes, i.e., there was no warning produced from these attributes, only the OEBA for continuous attributes was used. In this simulation, there were only 9 false positive cases, but the number of false negative cases was 19 cases. Note because our protocol for the simulation fixes all errors, these 19 cases correspond to 19 different types of case out of the 275 types of case the system handles.

To get the total false negative rate, we repeated the experiment without adding rules when a false negative occured. The results are shown in Table 6.5.

Threshold	FP	FN	TP	TN
8E-01	1386	10	270	18505
7E-01	1255	10	270	18636
6E-01	1174	10	270	18717
5E-01	1074	12	269	18816
4E-01	1022	15	268	18866
3E-01	964	15	268	18924
2E-01	709	23	266	19173
1E-01	589	28	264	19290
No Profile	9	44	256	19862

Table 6.5: Simulations when new rules are not added for false negatives. This table shows results for simulations with a different protocol from Table 6.4, that is, new rules are only added when warnings are correctly flagged. Other configurations are the same.

In summary, the 19 types of case that correspond to false negatives correspond to 44 actual cases in the 20,171 cases. One might surmise from this that false

negatives tend to be for fairly rare and unusual types of case.

6.5 Discussions of Probabilistic Profiles

Simulations of both OEBA and OECA algorithms on Garvan data set were very promising. The false positive rate was significantly reduced by approximately two-third. However, there were three false negative cases, where only one of the three was introduced by probabilistic profiles, while the other two were also missed by the original work.

An investigation of these three false negative cases was made. The three cases are shown in Table 6.6. Following is an explanation of the three cases.

Case 1

A high TT4 and low TSH together are indicators of thyrotoxicosis, or over treatment with thyroxine. In this case TT4 was towards the upper limit of the normal population range, and at the upper limit of the this OEBA profile for TT4, and although the TSH was diagnostically low it was within this OEBA profile for TSH. The difficulty of dealing with low TSH values is that they are at the limit of sensitivity of the analytical techniques, so that there is no significance difference between the level of 0.19 in this case and < 0.15 in case 2 below. < 0.15 simply means that value was below the sensitivity of the assay, which on this occasion was 0.15, while 0.19 probably has confidence limits of about +/- 0.15.

To apply the correlation analysis we have proposed, we had to normalize the TSH and TT4 values. The best normalisation was obtained by carrying out a log transform of TSH because of its very skewed distribution, and then normalizing values against the normal population range. This transformation was an arbitrary process and needs to be tested on a much larger set of data than a single case, but it probably always has to be based on domain specific knowledge about population distributions and correlations. In this case the normalisation resulted in a value outside the profile range, but the probability of this value was not less than the threshold, so the case was not flagged as an outlier.

recno	30401
ID N0	870813115
Age	31
Sex	F
Dr's Comment	hypothyroid
comment cde	8
Source	#SVHE
com etc code	00001000000
TSH	0.19
T3	?
TT4	147
T4U	0.92
FTI	160
TBG	?
FT4	?
diag rule	22310
diag code	1 A1

Elevated FTI & T4 consistent with thyrotoxicosis

recno	31280
ID N0	871102073
Age	61
Sex	F
Dr's Comment	hashimotos disease
comment cde	8
Source	#SJF
com etc code	00001000000
TSH	< 0.15
T3	2.2
TT4	177
T4U	?
FTI	?
TBG	?
FT4	?
diag rule	?
diag code	?

?

recno	35127
ID N0	880831020
Age	78
Sex	F
Dr's Comment	hypothyroid receiving supplements
comment cde	8
Source	#SVI
com etc code	00001001000
TSH	25
T3	0.7
TT4	92
T4U	?
FTI	?
TBG	?
FT4	?
diag rule	43420
diag code	18 G1&K1

Consistent with compensated hypothyroidism. Low T3 suggests a concurrent non thyroidal illness

Table 6.6: Three false negative cases.

Case 2

Patient 2 is a particularly interesting case. Patient 2 is a false negative because from the database and the simulated expert, the patient had no diagnosis, that is they are assumed to be normal, while the evolving RDR system had given a diagnosis of "Elevated T4, low TSH, normal T3. Consistent with toxicosis with a concurrent non-thyroidal illness". In fact the patient is clearly not normal, as there are clearly elevated levels of TT4 and suppressed levels of TSH, and the diagnosis given by the RDR system is more appropriate. This means it is not possible to decide what feature should have been flagged as an outlier so this more appropriate diagnosis was not given.

The simulated expert had learned to make the wrong diagnosis because of the inadequacy of some of the original classifications. What is interesting is that the original set of cases was developed by running Garvan cases through the original GARVAN-ES1 expert system [P. Compton, personal communication]. The reason for this was to ensure consistent classifications. This case must have fallen through the cracks of the original expert system; its rules did not cover this particular situation, although the results are clearly not normal. The correct conclusion would probably have been along the lines that the results were consistent with thyroxine treatment and a non-thyroidal illness or perhaps a more subtle comment on the possibility of transient elevated T4 with Hashimotos disease. From Wikipedia:

> Hashimoto's thyroiditis or chronic lymphocytic thyroiditis is an autoimmune disease where the body's own antibodies attack the cells of the thyroid. ... In many cases, Hashimoto's thyroiditis usually results in hypothyroidism, although in its acute phase, it can cause a transient thyrotoxic state. ... Treatment is with daily thyroxine. (Wikipedia, 2007b)

Case 3

The diagnosis for case 3 should be compensated hypothyroidism with a concurrent non-thyroidal illness, whereas the diagnosis give was simply compensated hypothyroidism. Clinically the differences are unimportant, with a relatively low

T3 the differentiating factor. The T3 for the case was well below the OEBA threshold, but because it was only the 16th case for this profile, the probability calculated was not low enough to be below the threshold. There would have needed to be about 340 cases for the probability to be low enough to be identified as an outlier.

In summary, prudence is a mechanism to let an expert or a user of an expert system know when an unusual case is being analyzed. In the original work, the system caused a high false positive rate. Our approach provides a solution to reduce this error rate. From simulations, the algorithm OEBA, implemented for continuous attributes, could reduce unnecessary warnings significantly, depending on a threshold used. As far as the algorithm OEBA could achieve, most of spurious warnings remaining were from new values for categorical attributes. We hence proposed the algorithm OECA for categorical attributes. Simulations showed that with a support from OECA, the system could further eliminate unnecessary warnings. However, configuring a threshold of OECA is critical, i.e., the smaller the threshold, the lower the false positive rate, but the false negative rate may also increase. OECA needs testing in other domains but is worth noting that it handled a range of categorical variables here with from 2 to 251 values, with the same OECA threshold for all variables. That is, the study does provide fairly wide evaluation.

6.6 An Extension on the Correlation between Attributes

As mentioned earlier, our approach detects an outlier by investigating each attribute separately. One might question correlations between attributes. This section proposes an approach to correlated attributes.

6.6.1 Correlations between Attributes

In statistics, a correlation is used to describe the degree of relationship between two variables. A number of different coefficients are used for different situations to specify correlations. The Pearson product-moment is the best known correlation

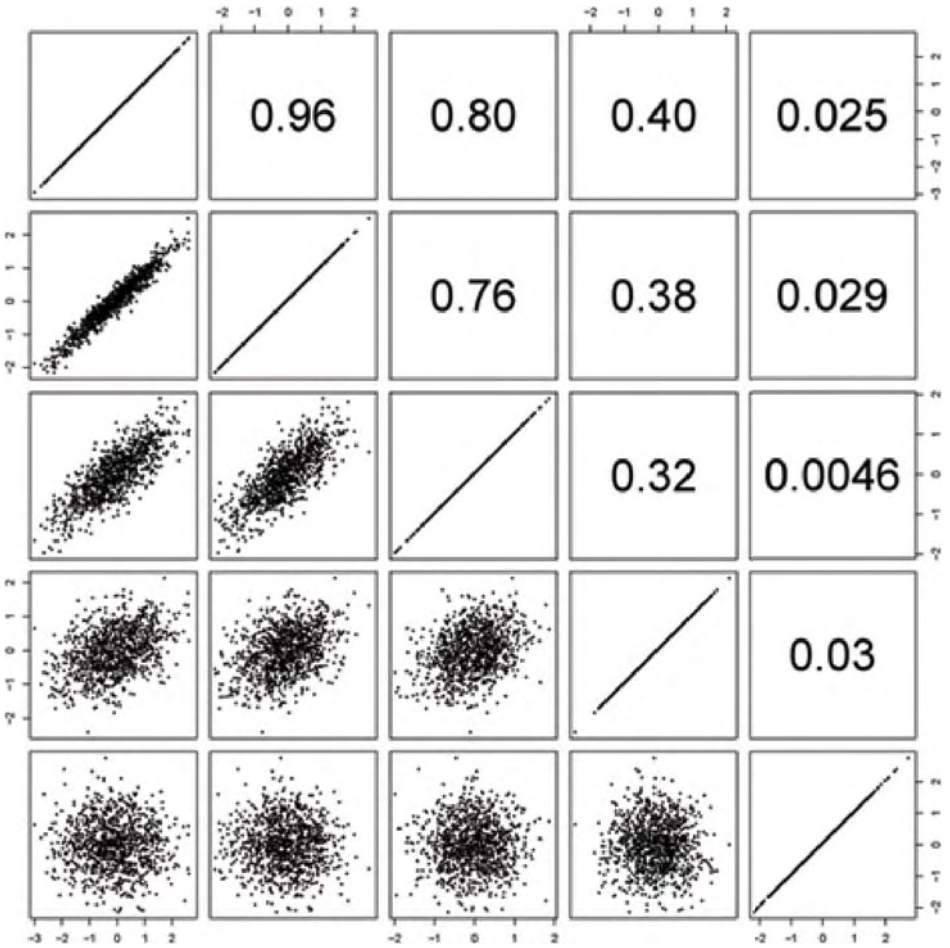


Figure 6.2: Examples of different correlations between 2 variables (Wikipedia, 2007a). Positive linear correlations between 1000 pairs of numbers. The data are graphed on the lower left and their correlation coefficients listed on the upper right. Each square in the upper right corresponds to its mirror-image square in the lower left, the *mirror* being the diagonal of the whole array. Each set of points correlates maximally with itself, as shown on the diagonal (all correlations = +1).

coefficient, which is obtained by dividing the covariance of the two variables by the product of their standard deviations (Wikipedia, 2007a). The Spearman rank order correlation (rho) or the Kendall rank order correlation (tau) are used when dealing with two ordinal variables. When one measure is continuous and the other is dichotomous, i.e., two-category, the Point-Biserial correlation is suggested. Visually, correlated data do not equally distribute across the whole space; that is, they are clustered in some regions, while other regions in the space are empty. Figure 6.2 shows some example different correlations between two continuous variables.

In medical domains, in particular, correlations between attributes are generally

the case. For example, the thyroid gland produces T4 and T3. But this production is not possible without stimulation from the pituitary gland (TSH), which in turn is also regulated by the hypothalamus's TSH Releasing Hormone. A low T4 level could mean a diseased thyroid gland or a non-functioning pituitary gland which is not stimulating the thyroid to produce T4. Since the pituitary gland would normally release TSH, if the T4 is low, a high TSH level would confirm that the thyroid gland, not the pituitary gland, is malfunctioning. On the other hand, if the T4 is low and TSH is not elevated, the pituitary gland is more likely to be the cause for the hypothyroidism.

The results so far support our contention that even in a highly correlated domain, attributes tend not be correlated within individual partitions. Nevertheless, a false negative case may occur because although each of its attributes falls within a maintained range, a combination of two attributes falls outside the region of their correlation. It is our intention to demonstrate that our approach can be applied to correlations between two continuous attributes.

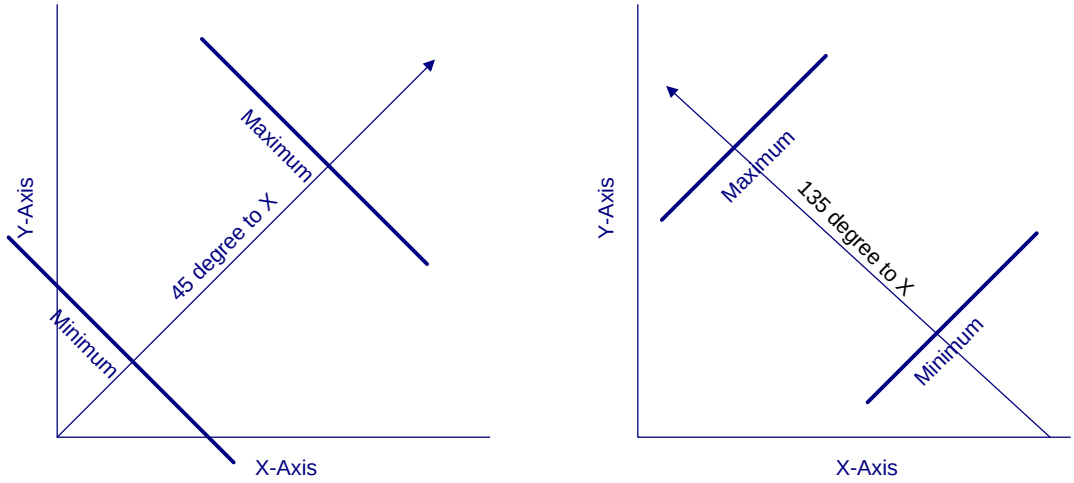
6.6.2 Correlation and the Algorithm OEBA

The algorithm OEBA separately maintains limits, i.e., the lower and upper bounds, of a continuous attribute. The limits of any two attributes define a rectangular boundary in a two-dimensional space. For example, the limits of attribute A_1 are 5 and 10. The limits of attribute A_2 are 3 and 20. In a two-dimensional space, where A_1 is mapped to X axis, and A_2 to Y axis, data are bounded by 4 linear lines as follows, $x = 5$, $x = 10$, $y = 3$ and $y = 20$. Provided two attributes are correlated, data are in a particular region within this rectangle. Our approach applies the algorithm OEBA to learn lower and upper bounds of this region in different directions, e.g., 45° or 135° to X axis as shown in Figure 6.3.

A straight line is in the form of

$$ax + by = c, \tag{6.8}$$

where a, b and c are coefficients such that a and b are not both zero. Lower and



(a) 45° to X axis

(b) 135° to X axis

Figure 6.3: Limits in different directions.

upper bounds of a direction are lines with a same slope, i.e., a and b are the same. For example, if a lower bound is $a_1x+b_1y = c_{min}$, a upper bound is $a_1x+b_1y = c_{max}$. Hence, the algorithm OEBA learn limits in a direction by maintaining c_{min} and c_{max}.

In our approach, we propose to maintain lower-upper bounds in two additional directions, i.e., 45° or 135° to X axis. In the direction of 45° to X axis, the slope is 1.0 and $a = -1.0$,$b = 1.0$. This means a profile for a correlation in this direction maintains limits of $c = y - x$. On the other hand, in the direction of 135° to X axis, the slope is -1.0 and $a = 1.0$, $b = 1.0$. This profile maintains limits of $c = y + x$.

This is a very simple approach. We are assuming that all correlations are linear and that we need to consider only pairwise correlations. Such a simple approach is reasonable when we have already demonstrated that considering attributes as independent gives us a almost perfect results. On the other hand, testing all possible multivariate correlations would greatly increase the false positive rate.

6.6.3 Simulation of Correlation Profile on Uncorrelated Attributes

Assuming that this approach may pick up some of the small number of false negatives, the key question is whether it will increase the false positives. A simulation was conducted to study additional false positives that might be increased by correlation profiles.

Simulations were set for three different domains, i.e., a domain with a single variable, a domain with two variables, and a domain with ten variables. In all domains, series of random numbers were generated following a mechanism used in Chapter [Outlier Detection], that is, each series contained 3,000 random numbers which were uniformly distributed in the range of 0.0 and 1.0.

In the univariate domain, there was only one OEBA profile created for the variable. In the bivariate domain (e.g., variables X_1 and X_2), there were four OEBA profiles created; two profiles for each variables X_1 and X_2, one for $X_1 + X_2$ and the other for $X_2 - X_1$. The number of profiles in the ten-variate domain was 100. That is 10 profiles for each variables, 90 profiles for correlations of each pair of variables. In the univariate and bivariate domains, simulations were run 10 times, while simulations were run twice for the ten-variate domain. The averages of false positive warnings produced was shown in Table 6.7.

Threshold T	1 var	2 var	10 var
1.0E-5	0	100.0	1654.5
1.0E-10	0	6.5	342.5
1.0E-15	0	3.2	74.0
1.0E-20	0	0.2	31.0
1.0E-25	0	0	12.0
1.0E-30	0	0	3.0
1.0E-35	0	0	0

Table 6.7: Simulations on OEBA profiles for correlation. The numbers shown in this table are the average of false positive instances for domains with one variable, 2 variables and 10 variables.

At $T = 1.0e-5$, there were 100 false positives produced in the bivariate domain and 1654.5 false positives in the ten-variate domain, while no warning was produced

in the univariate domain. However, the number of false positives dropped significantly when the threshold got smaller. It became zero when the threshold was as small as 1.0e-25 in the bivariate domain and 1.0e-35 in the ten-variate domain. This means that two additional profiles for correlation will not cause additional false positives when applied to uncorrelated data, provided the threshold is configured less than 1.0e-25 in a bivariate domain and 1.0e-35 in a ten-variate domain. This is well above the useful thresholds for the single attribute studies on the Garvan data. This is a particularly fortunate result as if there are n attributes, there will be $n^2 - n$ pairs of attributes greatly increasing the likelihood of false positives.

6.6.4 Simulation of OEBA and Correlations

Correlation profiles are now applied to continuous attributes in Garvan data set. This simulation was conducted on top of the simulation in Section 6.3, where the algorithm OECA was not taken into account. The results are shown in Table 6.8

Continuous Profile	FP	FN	FP+Correlation	FN+Correlation
Original Prudence	2841	2		
OEBA 1E-10	1933	3	2139	3
OEBA 1E-20	1877	3	1958	3
OEBA 1E-30	1868	3	1904	3
OEBA 1E-40	1861	3	1878	3
OEBA 1E-50	1858	3	1875	3
OEBA 1E-60	1857	3	1869	3
OEBA 1E-70	1853	3	1863	3
OEBA 1E-80	1851	3	1861	3
OEBA 1E-90	1850	3	1860	3

Table 6.8: Simulations of the algorithm OEBA for correlation between attributes.

The simulation demonstrates that when the algorithm OEBA was applied to correlations between continuous attributes, additional false positives were minor. For example, within the Garvan data, at $T = 1e - 30$, there were 36 extra false positives on top of 1,868 false positives from single attributes and when $T \leq 1e-70$, only 10 extra false positives on top of 1,853 false positives were introduced. However the approach failed to remove the third false negative. Using correlation profiles, the case was outside the profile but not below the threshold.

6.7 A Study on Warning Characteristics

The technique has been demonstrated as capable of efficiently detecting outliers and greatly reducing unnecessary warnings. False positives produced in an evaluation on Garvan data set were reduced from 14% to 5.85%, while true positives were 1.35% of all cases. That means 18.75% of all warnings were correctly produced. In our approach, a warning for a case is made if at least one of its profiles has raised a warning. No matter how many profiles have raised warnings, a case is simply flagged in the same way. It is our intention to study the characteristics of all warnings produced, as there might be some ways to distinguish correct warnings from false positives.

When a warning is produced for a case, additional information is collected as follows.

- The number of warnings produced from profiles.
- A vector of probabilities of warnings produced from profiles.

We spent considerable time trialling various methods of combining probabilities to get a single measure. We also used the J4.8 classifier to find if some combination of the measures we developed would be useful. We were unable to find any way of ranking cases that even approximately indicated the likelihood that a case was a true positive rather than a false positive. If such a measure could be found it would be of great value in suggesting priorities to the expert in scanning cases.

Chapter 7

Summary

Computer networks are increasingly critical in human lives. Many of our daily activities are involved with computer networks. Improving network performance is at the heart of computer network research. A major area is network security aimed at maintaining integrity, availability and confidentiality. One approach among others is intrusion detection systems (IDS).

Starting from a report by Anderson (Anderson, 1980) in 1980, through system prototypes implemented in many labs, e.g., IDES (Denning, 1987), Wisdom and Sense (Vaccaro and Liepins, 1989), MIDAS (Sebring et al., 1988), Haystack (Smaha, 1988), etc., the importance of IDS has been well realized. In 1987, Denning published a seminal model of IDS (Denning, 1987), upon which many later IDS's have been implemented.

In this thesis, a framework for a network intrusion detection system is proposed using a knowledge acquisition methodology called Ripple Down Models (RDM). RDM is a model-based variant of Ripple Down Rules. RDM incorporates a learning algorithm named Outlier Estimation with Backward Adaptation (OEBA) with each rule. The algorithm makes each rule capable of learning profiles of fired cases and detecting cases which differ from previous cases.

An attempt has also been made to improve *prudence* techniques in providing warnings to experts when a system is reaching the limits of its knowledge. A warning is produced when a new value for an attribute is encountered. In the thesis,

warnings are produced based on results from the algorithms OEBA and Outlier Estimation for Categorical Attributes (OECA), which are applied to continuous attributes and categorical attributes, respectively.

7.1 Thesis Summary

In this thesis, a brief history of IDS systems was presented in Section 2.2.4. From its functionality, an IDS is typically classified as either anomaly-based or misuse-based (or signature-based). A discussion of this classification was also presented in Section 2.2.4. A basic aim of an intrusion detection system is to detect anomalous behaviour occurring in a monitored network. The thesis identified four common tasks of IDS's, i.e., audit data storage, profile generation, intrusion detection, and policy execution.

The task of audit data storage mainly involves answers to these questions.

- What audit data is a system analyzing?
- How are they stored and retrieved?
- How big should the storage be?

Early systems detected anomalies from either log files or TCP connection information. Recently, network measurement, such as traffic volume, has been alternatively analyzed because it is easier to understand network performance from its utilization and it uses much less storage. Our study in this thesis analyzes traffic volume measured by the School of Computer Science and Engineering (CSE), UNSW. Traffic is monitored and stored by a network tool called RRDtool (Oetiker and GNU, 2006). The program is set to store the traffic utilization of monitored protocols for a maximum of 7 days. Details are in Section 5.1.1.

Profile generation is the heart of an IDS. Profiles are the knowledge of the system. They are retrieved from a knowledge base to match against new observations. System developers need to make a decision on what knowledge from audit data can be derived and represented. Depending on IDS's categories, there are two main

types of profile. The first one is a profile of normal behavior. The others are profiles of known attacks. Section 2.3.2 gives a detailed discussion of the task.

Intrusion detection is tightly coupled with profile generation. From statistical approaches, to other recent advances, many techniques have been proposed. Section 2.3.3 summarized some techniques used so far. A common idea among early techniques was the use of a generic model of a system. The disadvantages of this are the difficulty of defining a single universal model and the difficulty of handling novel and ad-hoc events.

In contrast, this thesis proposes a framework named Ripple Down Models (RDM) to partition a complex problem space with diversity of data into smaller sub-spaces of homogeneous data and separately learn a profile for each homogeneous region. RDM is a variant of a KA technique Ripple Down Rules (RDR). The philosophy of RDR in relation to cognitive science was reviewed in Section 3.3.1. The structure, functionality and terminology of RDR were reviewed in Section 3.3.2. RDR have been applied to many tasks, as reviewed in Section 3.3.6. In this study, RDR is applied to a segmentation task, i.e., partitioning a problem space into smaller sub-spaces (or regions) of data with the same classification.

Within a region, models are learned using the novel algorithm Outlier Estimation with Backward Adaptation (OEBA). The algorithm OEBA is an outlier detection algorithm that is capable of online learning and detecting outliers simultaneously. Section 4.2 presents a background on outlier detection, including some detection techniques used so far. OEBA is assumes that the data follow a uniform distribution; hence, it learns and detects outliers based on statistics calculated from past observations, as discussed in Section 4.3. Simulations and performance evaluation are presented in Section 4.4. OEBA shows very promising results for a learning algorithm; it produces very low false positive and false negative rates.

RDM modifies RDR as follows.

- It incorporates the algorithm OEBA with each rule.
- No classification is made from an inference in RDM. OEBA profiles are returned, instead.

- It does not maintain cornerstone cases. The OEBA profiles associated with a rule act as a cornerstone case.

Similarly to RDR, RDM adds a new profile for a region into its knowledge base when discrepancies between profiles and new observations are detected. A newly added region might be a new definition of a space that has never been covered or a refinement or an exception of an already defined space. Hence, any novel and ad-hoc events can be added to a knowledge base at any time when required.

A prototype has been implemented using the JAVA language and tested on traffic data sets from the school of CSE. Starting from an empty knowledge base, experts gradually add new definitions of traffic utilization while the algorithm OEBA successfully detects anomalous traffic with low false positive rate. Although the framework has been tested for network intrusion detection, we believe that it can apply to any complexed domain with a diversity of data.

It is worth noting that the algorithm OEBA is able to adjust itself to changes in the domain it is applied to. OEBA learns a range of values in a region point by point and justifies the likelihood of an outlier when a value falls out of range by a simple probabilistic technique based on past observations. A further application of the algorithm OEBA was a prudence technique for expert systems.

Most expert systems have a problem of brittleness, where a conclusion is made without knowing that the system has reached the limits of its knowledge. A classic attempt to avoid this problem is the project CYC (Guha and Lenat, 1990), which provides a foundation knowledge base of common sense, upon which other expert systems can be built. Prudence in RDR is another attempt at a solution to brittleness. It provides warnings to experts when a system is making a conclusion for a case beyond its experience. The previous study of prudence in (Compton et al., 1996) was successful in notifying experts of cases that required attention, but the false positive was high at 15%

Once OEBA is applied to prudence, unnecessary or false positive warnings are significantly reduced. However, false positive warnings cannot be eliminated because some warnings are produced from new values of categorical attributes.

Hence, we developed another learning algorithm for categorical attributes. This algorithm is called Outlier Estimation for Categorical Attributes (OECA). When both OEBA and OECA algorithms are used together, false positive warnings are significantly reduced down to 5.85%.

Both algorithms are simple and online. From an empty state, they incrementally learn data, while a calculation based on past observations is made to estimate the likelihood of any new value seen being an outlier. It is shown by both simulations and experiments on real medical data sets that they are efficient learning algorithms for outlier detection. We suggest that both algorithms can learn attributes for many domains and appropriately notify experts of unusual cases that are evaluated. In conjunction with the RDM framework, any complex problem space where data are diverse can be partitioned into smaller sub-spaces of more homogeneous data which can be separately learned by the two algorithms.

7.2 Discussion

Our goal was not to build a system that was necessarily more accurate than the best systems previously built for specific domains. Our aim was rather to develop a way of rapidly and easily customizing an IDS with a high level of accuracy for any domain and able to be easily customized further as the domain changed. However our approach does produce a very good level of accuracy compared to previous studies using volume data.

The system that is perhaps closest to our study is by Brutlag (Brutlag, 2000), where anomaly detection is performed by the Holt-Winters algorithm on traffic volumes archived by RRDtool. However, there are no false positive or false negative rates published anywhere.

Another system by Barford et al. (Barford et al., 2002) uses wavelet filters to analyze traffic volumes. In their work, the published result for anomaly detection is only that their technique can detect 38 out of 39 anomalies, in comparison to the Holt-Winters algorithm which can only detect 37 anomalies. The false positive

rate is not reported.

Lakhina et al. (Lakhina, Crovella, and Diot, 2004b), investigate their system with 3 data sets. In evaluation, they use two methods, i.e., Fourier and an exponential smoothing EWMA, to label anomalies in traffic volumes, against which their detection results are compared. As the two reference methods label anomalies differently, their results vary. With the Sprint-1 data set, the false positive rate is reported 1/999 when compared with Fourier labelled anomalies, and 6/1003 when compared with EWMA labelled anomalies. The false negative rate is reported as 0 when compared with Fourier labelled anomalies, and 1/5 when compared with EWMA labelled anomalies. However, when they investigate their system on synthetic anomalies, the false negative rate is reported at 7% on the Sprint data set, but the false positive rate is not reported.

The proposed method shows that all anomalies can be detected, i.e., a false negative rate of zero, while false positive rates are at about 2% when profiles for traffic are created. It should be noted that there are no training sessions before putting the system in use. The expert can easily add profiles and decision rules as required. This is a small addition to the normal network administration task.

In the proposed framework for intrusion detection systems, double knowledge bases (KB) are used. The first KB is used in anomaly detection, while the second KB maintains final rules on top of the previous anomaly detection. The expert is free to add a decision rule or a new profile - whatever they think is appropriate. Decision rules are not only defined to reduce false positive warnings, the expert can define rules for network attacks. That is, our system learns both normal and intrusive behaviour and is essentially a hybrid anomaly and signature detection system.

It is worth noting that our framework is open to any environment. Although the system is implemented to detect anomalies from traffic volumes, it could also analyze packet header information. Developers simply add features to cases for an analysis and experts can add profiles or decision rules for these features. We have not compared our approach with packet header systems as these are designed to classify anomalies while we simply detect them.

7.3 Future Research

7.3.1 An Interim Outlier Detection Algorithm

Both OEBA and OECA algorithms gradually learn from an empty state. Although the study shows that they work very well, it is worth noting that when the number of seen cases is small, they are more likely to accept outliers, causing false negatives. The reasons why the problem does not show in our study are as follows. We do not appear to have had many cases where an outlier occurred almost immediately after a new rule or model was added and new profiles developed even in multiple randomized simulations. On the other hand, we do not get later false negatives because the algorithm OEBA can adapt itself by cutting out outliers that were earlier mistakenly added into its model. However, we cannot guarantee that outliers will always occur after the algorithm is mature.

One solution would be to simply check all cases for new profiles for some period. However the deeper the refinement, the less likely a rule will cover other as yet unknown classes, so ideally the number of initial cases checked would vary with the rule location.

7.3.2 Combining Multiple Tests

Prudence is a different application to anomaly detection. Using the second knowledge base one can write rules about what sort of anomalies should be brought to the expert attention. In contrast, prudence is watching for new entities, that have been covered by an overgeneralization. Since the expert has overgeneralized, it would not make sense to expect rules about the overgeneralization in the second knowledge base. So for prudence, each attribute for the case is analyzed against the profile for that attribute and a warning is raised if necessary, and a warning for the case is flagged if any of those profiles raise warnings.

During this study, we attempted to determine relationships between combinations of warnings produced from profiles and cases that should be warned about.

If successful, experts should only look up at cases where supervision was really required, instead of reviewing all the cases that were warned about. Unfortunately, no relationship could be drawn from the investigation in section 6.7.

However, we believe there should be further investigation into combining warnings into a single overall warning. If it was possible to rank cases according to the likelihood that the warning was real, this would allow the expert under time pressures to decide how far down the list of cases they should check. This would be valuable even if the ranking was only approximate.

7.3.3 Redundancy of Partitions

In RDR, when a conclusion made for a case is incorrect, a new rule is formed by a conjunction of selected features that makes a case different from a cornerstone case of a currently fired rule and is added as an exception to the fired rule. The process is simple but there is no guarantee that rule redundancy may not occur. Suryanto investigated rule redundancy in RDR and proposed to re-organize RDR using Disjunctive Normal Form (DNF) representation to facilitate a removal of rule redundancy (Suryanto, 2005). It has been shown that the number of rules can be reduced by 10% and the number of future KA sessions is reduced by 27% (Suryanto, 2005).

Like other RDR-based systems, there is no guarantee that a redundancy does not occur when a new rule is added to RDM. It might be useful to use the profiles that are learned to help reduce redundancy building on the work of (Suryanto, 2005) and (Yoshida et al., 2002; Yoshida et al., 2004).

7.3.4 Correlations Between Classifications

Our framework has demonstrated that it is reasonable to omit correlations between attributes after a problem space is partitioned into regions of homogeneous data. However, the possibility of correlated data cannot be completely eliminated.

We have argued that it is not critical for our technique to handle correlations as

the problem space tends to get partitioned into homogeneous regions. The prudence studies on thyroid data provide strong support for this as although thyroid data is highly correlated, the fact that we only had three false negatives demonstrates the correlations are across partitions rather than within partitions.

We have developed a technique for handling two-variable correlations and have shown that it increases false positives by less than 1% for two variables. As expected and shown in Section 6.6, the false positive rate increases with more variables since the number of profiles kept is N^2 rather than N, where N is the number of variables. To use the technique for large numbers of variables, it would probably require a domain analysis to decide which variables should be considered. Some domain analysis is needed also to normalize the variables for pairwise consideration. We need to evaluate the approach on real domains where correlation might be a problem - despite the fact that the RDR approach minimizes correlations.

Bibliography

Allen, J.R. (1999). Driving by the rear-view mirror: Managing a network with cricket. In *Proceedings of the 1st Conference on Network Administration*.

Allen, J.R. (2003). The cricket reference guide. Retrieved December, 2006, from http://cricket.sourceforge.net/support/doc/reference.html.

Anderson, D., T. Frivold, and A. Valdes (1995). Next generation intrusion-detection expert system(nides). Technical report SRI-CSL-95-07, Computer Science laboratory, SRI International, Menlo Park, CA, USA.

Anderson, J.P. (1980). Computer security threat monitoring and surveillance. Technical report, James P. Anderson Co., Fort Washington, PA.

Anderson, R.J. (2001). *Security Engineering: A Guide to Building Dependable Distributed Systems*, chapter 18 Network Attack and Defense. John Wiley & Sons inc.

Angiulli, F., S. Basta, and C. Pizzuti (2006). Distance-based detection and prediction of outliers. *IEEE Transactions on Knowledge and Data Engineering* 18: 145–160.

Axelsson, S. (2000). Intrusion detection system: A survey and taxonomy. Technical report 99-15, Chalmers University.

B. Mukherjee, L. T. Heberlein, K.N. levitt (1994). Network intrusion detection. *IEEE Network* 8(3): 26–41.

Barford, P., J. Kline, D. Plonka, and A. Ron (2002). A signal analysis of network traffic anomalies. In *Internet Measurement Workshop 2002*.

Barford, P. and D. Plonka (2001). Characteristics of network traffic flow anomalies. In *In Proceedings of the ACM SIGCOMM Internet Measurement Workshop*.

Barnett, V. and T. Lewis (1987). *Outliers in Statistical Data*. Wiley Series in Probability and Mathematical Statistics. John Wiley & Sons, 2 edition.

Beal, V. (2006). The difference between a virus, worm and trojan horse. Retrieved November, 2006, from http://www.webopedia.com/DidYouKnow/Internet/2004/virus.asp.

Beale, J., J.C. Foster, J. Posluns, and B. Caswell (2003). *Snort 2.0 Intrusion Detection*. Syngress Publishing, Inc., Rockland, MA.

Bekmann, J. and A. Hoffmann (2005). Improved knowledge acquisition for high-performance heuristic search. In *Proceedings of the 19th international joint conference on artificial intelligence IJCAI*, pp. 41–46, Denver, USA.

Beydoun, G. and A. Hoffmann (1997). Nrdr for the acquisition of search knowledge. In *Australian Joint Conference on Artificial Intelligence*, pp. 177–186.

Beydoun, G. and A. Hoffmann (1998). Building problem solvers based on search control knowledge. In *11th Banff Knowledge Acquisition for Knowledge Based System Workshop*, pp. Share1/1 – Share1/6.

Bradshaw, J.M., J.H. Boose, and S.P. Covington (1987). How to do with grids what people say you can't. In *Journal of Knowledge Acquisition for Knowledge-Based Systems*, Vol. 1.

Breunig, M.M., H. Kriegel, R.T. Ng, and J. Sander (1999). Optics-of: Identifying local outliers. In *Proceedings of the 3rd European Conference on Principles and Practice of Knowledge Discovery in Databases*, Praque.

Breunig, M.M., H. Kriegel, R.T. Ng, and J. Sander (2000). Lof: Identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pp. 93–104, Dallas, Texas, US. ACM Press New York, USA.

Brockwell, P.J. and R.A. Davis (1996). *Introduction to Time Series and Forecasting*. Springer, New York.

Brutlag, J.D. (2000). Aberrant behavior detection in time series for network monitoring. In *Proceedings of the 14th Systems Administration Conference(LISA 2000)*, New Orleans, Louisiana, USA. USENIX Association.

Buchanan, G.G., R. Betchel, J. Bennet, W. Clancey, C. Kulikowski, M.T. M., and D.A. Waterman (1983). Constructing an expert system. In Hayes-Roth, F., D.A. Waterman, and D.B. Lenat, editors, *Building Expert Systems*, pp. 127–167. Addison-Wesley, London.

Caida (2006). Flowscan - network traffic flow visualization and reporting tool. Retrieved December, 2006, from http://www.caida.org/tools/utilities/flowscan/.

Cannady, J. (1998). Artificial neural networks for misuse detection. In *Proceedings of the 1998 National Information Systems Security Conference (NISSC'98)*, pp. 443–456.

Cao, T.M. and P. Compton (2005). A simulation framework for knowledge acquisition evaluation. In Estivill-Castro, V, editor, *Twenty-Eighth Australasian Computer Science Conference (ACSC2005)*, pp. 353–360, Newcastle.

Cartwright, D. (2007). Software application review: Snort 2.7 review. Retrieved November, 2007, from http://www.techworld.com/applications/reviews/index.cfm?reviewID=562&pagtype=all.

Catlett, J. (1992). Ripple down rules as a mediating representation in interactive induction. In *Proceedings of the Second Japanese Knowledge Acquisition for Knowledge Based Systems Workshop*, pp. 155–170, Kobe, Japan.

Center, CERT Coordination (1995). Cert advisory ca-1995-01 ip spoofing attacks and hijacked terminal connections. Retrieved January, 2006, from http://www.cert.org/advisories/CA-1995-01.html.

Center, CERT Coordination (2002). Spoofed/forged email. Retrieved January, 2006, from http://www.cert.org/tech_tips/email_spoofing.html.

Cheng, C., H.T. Kung, and K. Tan (2002). Use of spectral analysis in defense against dos attacks. In *Proceedings of Global Telecommunications Conference, 2002, IEEE GLOBECOM*, pp. 2143–2148. IEEE.

Clancey, W.J. (1991). Situated cognition: Stepping out of representational flatland. *AI Communications-The European Journal on Artificial Intelligence* 4(2/3): 109–112.

Clancey, W.J. (1993). Situated cognition: How representations are created and given meaning. In Lewis, R. and P. Mendelsohn, editors, *Lessons from Learning*, pp. 231–242.

Compton, P., T. Cao, and J. Kerr (2004). Generalising incremental knowledge acquisition. In Kang, B, A Hoffmann, T Yamaguchi, and W.K. Yeap, editors, *Proceedings of the Pacific Knowledge Acquisition Workshop 2004*, pp. 44–53. University of Tasmania Eprints repository.

Compton, P., G. Edwards, B. Kang, and al. (1991). Ripple down rules: Possibilities and limitations. In *6th Banff AAAI Knowledge Acquisition for Knowledge Based Systems Workshop*.

Compton, P., R. Horn, R. Quinlan, and Lazarus (1989). Maintaining an expert system. In Quinlan, R., editor, *Applications of Expert Systems*, pp. 366–385. Addison Wesley.

Compton, P. and R. Jansen (1988). Knowledge in context: a strategy for expert system maintenance. In *Seconde Australian Joint Artificial Intelligence Conference*, pp. 292–306.

Compton, P. and R. Jansen (1990). A philosophical basis for knowledge acquisition. *Knowledge Acquisition* 2: 241–257.

Compton, P., B. Kang, P. Preston, and al. (1993). Knowledge acquisition without analysis. In *European Knowledge Acquisition Workshop*. Springer-Verlag.

Compton, P., L. Peters, G. Edwards, and T.G. Lavers (2006). Experience with ripple-down rules. *Knowledge-Based System Journal* p. in press.

Compton, P., P. Preston, G. Edwards, and B. Kang (1996). Knowledge based system that have some idea of their limits. In *Proceedings of the 10th AAAI-Sponsored Banff Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff, Canada.

Compton, P., P. Preston, and B. Kang (1995). The use of simulated experts in evaluating knowledge acquisition. In Gaines, B. and M. Musen, editors, *Proceedings of the 9th AAAI-Sponsored Banff Knowledge Acquisition for Knowledge-Based Systems Workshop*, pp. 12.1–12.8, Banff, Canada, University of Calgary.

Compton, P., Z. Ramadan, P. Preston, and al. (1998). A trade-off between domain knowledge and problem-solving method power. In *The 11th Banff Knowledge Acquisition for Knowledge Based Systems Workshop.*

Compton, P. and D. Richards (1999). Extending ripple down rules. In *Proceeding of the Fourth Australian Knowledge Acquisition Workshop 99*, pp. 87–100, UNSW.

Compton, P. and D. Richards (2000). Generalising ripple down rules. In Deng, R. and O.Corby, editors, *Knowledge Engineering and Knowledge Management: Methods, Models, Tools*, pp. 380–386. Juan-les-Pins.

Corbridge, C., G. Rugg, N.P. Major, N.R. Shadbolt, and A.M. Burton (1994). Laddering: technique and tool use in knowledge acquisition. *Knowledge Acquisition* 6: 315–341.

Cordingley, E.S. (1989). Knowledge elicitation techniques for knowledge-based systems. In Diaper, D., editor, *Knowledge Elicitation: Principle, Techniques and Applications*, pp. 87–175. John Wiley & Sons.

Crawford, E., J. Kay, and E. McCreath (2002a). Iems the intelligent email sorter. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pp. 83–90. Morgan Kaufmann Publishers Inc.

Crawford, E., J. Kay, and E. McCreath (2002b). An intelligent interface for sorting electronic mail. In *Proceedings of the 2002 Conference on Intelligent User Interfaces*, pp. 182–183. ACM.

Datar, M. and S. muthukrishnan (2001). Estimating rarity and similarity over data stream windows. Technical report 21, DIMACS.

Debar, H., M. Becker, and D. Siboni (1992). A neural network component for an intrusion detection system. In *Proceedings of the 1992 IEEE Computer Society Symposium on Research in Security and Privacy*, pp. 240–250, Oakland, USA.

Denning, Dorothy E. (1987). An intrusion-detection model. *IEEE Transactions on Software Engineering* SE-13(2): 222–232.

Duda, R. and E. Shortliffe (1983). Expert systems research. *Science* 220: 261–268.

Durkin, J. (1994). *Expert Systems Design and Development*. Macmillan Publishing Company.

Edwards, G., P. Compton, R. Malor, A. Srinivasan, and L. Lazarus (1993). Peirs: a pathologist maintained expert system for the interpretation of chemical pathology reports. *Pathology* 25: 27–34.

Edwards, G, B Kang, P Preston, and P Compton (1995). Prudent expert systems with credentials: Managing the expertise of decision support systems. *Int. J. Biomed. Comput.* 40: 125–132.

Fallon, A. and C. Spada (1997). Detection and accommodation of outliers in normally distributed data sets. Retrieved June, 2005, from http://ewr.cee.vt.edu/environmental/teach/smprimer/outlier/outlier.html.

Feigenbaum, E.A. (1984). Knowledge engeering: The applied side of artificial intelligence. In *Proc. of a symposium on Computer culture: the scientific, intellectual, and social impact of the computer*, pp. 91–107, New York. New York Academy of Science New York.

Gaines, B. (1991). Induction and visualisation of rules with exceptions. In *6th AAAI Knowledge Acquisition for Knowledge Based Systems Workshop*, pp. 7.1–7.17, Banff.

Gaines, B.R. and P. Compton (1992). Induction of ripple down rules. In *Proceedings of the 5th Australian Joint Conference on Artificial Intelligence*, pp. 349–354, Hobart, Tasmania. World Scientific, Singapore.

Gaines, B.R. and P. Compton (1995). Induction of ripple-down rules applied to modeling large databases. *Journal of Intelligent Information Systems* 5(3): 211–228.

Gaines, B.R. and M.L.G. Shaw (1993). Eliciting knowledge and transferring it effectively to a knowledge-based system. *Knowledge and Data Engineering* 5(1): 4–14.

Gilbert, A.C., S. Guha, P. Indyk, Muthukrishnan, and M.J. Strauss (2001). Quicksand: Quick summary and analysis of network data. Technical report 43, DIMACS.

Goupillaud, P., A. Grossmann, and J. Morlet (1984). Cycle-octave and related transforms in seismic signal analysis. *Geoexploration* 23: 85–102.

Guan, Y., A.A. Ghorbani, and N. Belacel (2003). Y-means: A clustering method for intrusion detection. In *Proceedings of the 2003 Canadian Conference on Electrical and Computer Engineering*, Montreal.

Guha, R.V. and D. Lenat (1990). Cyc: A midterm report. AI Magazine.

Hawkins, D. (1980). *Identification of Outliers*. Chapman and Hall, London.

Hayes-Roth, F., D. Waterman, and D. Lenat (1983). *Building Expert Systems*. Addison-Wesley, New York.

Heberlein, L.T., G.V. Dias, K.N. Levitt, B. Mukherjee, J. Wood, and D. Wolber (1990). A network security monitor. pp. 296–303.

Huber, R.J. (1981). *Robust Statistics*. Wiley.

Hussain, A., J. Heidemann, and C. Papadopoulos (2003). Identification of repeated attacks using network traffic forensics. Technical report, USC/Information Sciences Institute.

Hussain, A., J. Heidemann, and C. Papadopoulos (2006). Identification of repeated denial of service attacks. In *Proceedings of INFOCOM 2006, 25th IEEE International Conference on Computer Communications*, pp. 1–15, Barcelona, Spain. IEEE.

Iglewicz, B. and D.C. Hoaglin (1993). *How to Detect and Handle Outliers*. American Society for Quality Control, Milwaukee, WI.

Ilgun, K. (1993). Ustat: A real-time intrusion detection system for unix. *IEEE Symposium on Security and Privacy* pp. 16–28.

Innella, P. (2001). The evolution of intrusion detection systems. Retrieved December, 2006, from http://www.securityfocus.com/infocus/1514.

Institute, SANS (2005). The twenty most critical internet security vulnerabilities (updated) the expert consensus, v.6.01. Retrieved January, 2006, from http://www.sans.org/top20.

Javvin Technologies, Inc. (2006). Port scan attack. http://javvin.com/networksecurity/PortScanAttack.html.

Jung, J., B. Krishnamurthy, and M. Rabinovich (2002). Flash crowds and denial of service attacks: Characterization and implications for cdns and web sites. In *Proceedings of the World Wide Web Conference*, Honolulu, Hawaii.

Kang, B., P. Compton, and P. Preston (1995). Mulitple classification ripple down rules: Evaluation and possibilities. In *The 9th AAAI-Sponsored Banff Knowledge Acquisition for Knowledge Based Systems Workshop*.

Kang, B., P. Compton, and P. Preston (1998). Simulated expert evaluation of multiple classification ripple down rules. In *Proceedings of 11th Banff Knowledge Acquistion for Knowledge-Based System Workshop*, pp. 1–19, University of Calgary.

Kelly, G.A. (1955). A theory of personality: The psychology of personal constructs.

Kivinen, J., H. Mannila, and E. Ukkonen (1993). Learning rules with local exceptions. In *European Conference on Computational Theory*, pp. 35–46. Clarendon Press, Oxford.

Knorr, E.M. and R.T. Ng (1997). A unified notion of outliers: Properties and computation. In *Knowledge Discovery and Data Mining*, pp. 219–222.

Knorr, E.M. and R.T. Ng (1998). Algorithms for mining distance-based outliers in large datasets. In *Proceedings of the 24th VLDB Conference*, pp. 392–403, New York, USA.

Kohavi, R. and F. Provost (1998). Glossary of terms: Special issue on applications of machine learning and the knowledge discovery process. In Kohavi, R. and F. Provost, editors, *Machine Learning*, Vol. 30. Kluwer Academic.

Krishnamurthy, B., S. Sen, Y. Zhang, and Y. Chen (2003). Sketch-based change detection: Methods, evaluation, and applications. In *IMC'03: Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, Miami Beach, Florida, USA.

Kumar, S. and E.H. Spafford (1994). An application of pattern matching in intrusion detection. Technical report CSD-TR-94-013, Department of Computer Sciences, Purdue University.

Lakhina, A., M. Crovella, and C. Diot (2004a). Characterization of network-wide anomalies in traffic flows. In *Proceedings of Internet Measurement Conference 04*, Italy. ACM.

Lakhina, A., M. Crovella, and C. Diot (2004b). Diagnosing network-wide traffic anomalies. Portland.

Leland, W.E., M.S. Taqqu, W. Willinger, and D.V. Wilson (1994). On the self-similar nature of ethernet traffic (extended version). In *IEEE/ACM Transactions on Networking (TON)*, pp. 1–15. IEEE Press Piscataway, NJ, USA.

Lenat, D. (1994). A brief list of the applications. Retrieved January, 2006, from http://www.cyc.com/cyc/technology/cycandd/brieflist.

Lenat, D. (2002). Artificial intelligence as common sense knowledge. Retrieved January, 2006, from http://www.leaderu.com/truth/2truth07.html.

Lizard, A. (2002). Using snort for intrusion detection. Retrieved November, 2007, from http://web.njit.edu/m̃anikopo/SNORT/snort_review1.html.

Lunt, T.F., R. Lee R. Jagannathan, S. Listgarten, D.L. Edwards, P.G. Neumann, H.S. Javitz, and A. Valdes (1988). Ides: The enhanced prototype, a real-time intrusion-detection expert system. Technical report SRI-CSL-88-12, Computer Science laboratory, SRI International, Menlo Park, CA, USA.

Lyle, M.P. (1998). Attacks and countermeasures- a study of network attact classes and security components to protect against them. Technical report, Recourse Technologies inc.

Mandjes, M., I. Saniee, and A.L. Stolyar (2005). Load characterization and anomaly detection for voice over ip traffic. *IEEE Transactions on Neural Networks* 16(5): 1019–1026.

Mcgraw, K.L and K. Harbison-Briggs (1989). Review: Knowledge acquisition for expert systeems.

Mills, T.C. (1990). *Time Series Techniques for Economists*. Cambridge University Press.

Misra, A., A. Sowmya, and P. Compton (2004). Incremental learning of control knowledge for lung boundary extraction. In Kang, B., editor, *Proceedings of the Pacific Knowledge Acquisition Workshop PRICAI 2004*, pp. 211–225, Auckland, New Zealand.

Mulholland, M., P. preston, B. Hibbert, and P. Compton (1993). An expert system for ion chromatography developed using machine learning and knowledge in context. In *Proceedings of the Sixth International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, Edinburgh.

Mycoted (2006). Laddering. Retrieved December, 2006, from http://www.mycoted.com/Laddering.

Oetiker, T. and GNU (2006). Rrd tool. Retrieved December, 2006, from http://oss.oetiker.ch/rrdtool/index.en.html.

Papadimitriou, S., H. Kitawaga, P.B. Gibbons, and C. Faloutsos (2002). Loci: Fast outlier detection using the local correlation integral. Technical report IRP-TR-02-09, Intel Research.

Park, M., L. Wilson, and J. Jin (2000). Automatic extraction of lung boundaries by a knowledge-based method. In *Visualisation 2000*, pp. 11–16, Sydney, Australia.

Porras, P.A. and P.G. Neumann (1997). Emerald: Event monitoring enabling responses to anomalous live disturbances. In *Proc. 20th NIST-NCSC National Information Systems Security Conference*, pp. 353–365.

Prayote, A. and P. Compton (2006). Detecting anomalies and intruders. In *Proceedings of AI 2006: Advances in Artificial Intelligence*, Hobart, Tasmania, AU.

Preston, P., G. Edwards, and P. Compton (1994). A 2000 rule expert system without knowledge engineers. In *The 8th AAAI-Sponsored Banff Knowledge Acquisition for Knowledge Based Systems Workshop.*

Puppe, F. (1993). *Systematic Introduction to Expert Systems.* Springer-Verlag, USA.

Richards, D. and P. Compton (1998). Taking up the situated cognition challenge with ripple down rules. *Int.J. Human-Computer Studies* pp. 895–926.

Richards, D. and P. Compton (1999). Sisyphus i revisited: An incremental approach to resource allocation using ripple down rules. In *12th Workshop on Knowledge Acquisition, Modeling and Management*, pp. 7–7.1 – 7–7.20, Banff, Canada. SRDG Publications.

Rousseeuw, P.J. and A.M. Leroy (1987). *Robust Regression and Outlier Detection.* John Wiley & Sons, New York.

Rugg, G., M. Eva, A. Mahmood, N. Rehman, S. Andrews, and S. Davies (2002). Eliciting information about organizational culture via laddering. *Journal of Information System* 12: 215–230.

Russell, S. and P. Norvig (1995). *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Inc.

Ruts, I. and P. Rousseeuw (1996). Computing depth contours of bivariate point clouds. *Computational Statistics and Data Analysis* 23: 153–168.

Ryan, J., M. Lin, and R. Miikkulainen (1998). Intrusion detection with neural networks. In Jordan, M.I., M.J. Kearns, and S.A. Solla, editors, *Advances in Neural Information Processing Systems*, Vol. 10. The MIT Press.

Scheffer, T. (1996). Algebraic foundations and improved methods of induction or ripple down rules. In *The Second Pacific Rim Knowledge Acquisition Workshop*, pp. 279–292.

Sebring, M., E. Shellhouse, M. Hanna, and R. Whitehurst (1988). Expert systems in intrusion detecion: A case study. In *Proceedings of the 11th National Computer Security Conference*, pp. 74–81, Baltimore, Mariland.

Shaw, M.L. and B.R. Gaines (1988). Kitten: Knowledge initiation and transfer tools for experts and novices. In Boose, J. and B.R. Gaines, editors, *Knowledge-Based Systems*, Vol. 2: Knowledge Acquisition Tools for Expert Systems, pp. 309–338. Academic Press, New York.

Siromoney, A. and R. Siromoney (1993). Variations and local exception in inductive logic programming. In Furukawa, K., D. Michie, and S. Muggleton, editors, *Machine Learning - Applied Machine Intelligence*, Vol. 14, pp. 213–234.

Smaha, S. (1988). Haystack: An intrusion detection system. In *Proceedings of 4th Aerospace Computer Security Applications Conference*, pp. 37–44. IEEE Computer Society Press.

Soule, A., K. Salamatian, and N. Taft (2005). Combining filtering and statistical methods for anomaly detection. In *IMC'05, 2005 Internet Measurement Conference*, pp. 331–344.

Suryanto, H. (2005). Learning and Discovery in Incremental Knowledge Acquisition. Ph.D. diss., Computer Science and Engineering, University of New South Wales, Australia.

Suryanto, H., D. Richards, and P. Compton (1999). The automatic compression of multiple ripple down rule knowledge based systems: Preliminary experiments. In *Knowledge-Based Intelligence Information Engineering Systems*, pp. 203–206, Adelaide, South Australia. IEEE.

Teng, H.S., K. Chen, and S.C. Lu (1990). Security audit trail analysis using inductively generated predictive rules. In *Proceedings of 6th Conference on Artificial Intelligence Applications*, pp. 24–29, IEEE Service Center, Piscataway, NJ.

Tukey, J.W. (1977). *Exploratory Data Analysis*. Addison-Wesley, Reading, MA.

Tversky, A. and D. Kahneman (1974). Judgment under uncertainty: Heuristics and biases. *Science* 185: 1124–1130.

Vaccaro, H.S. and G.E. Liepins (1989). Detection of anomalous computer session activity. In *Proceedings of the 1989 IEEE Symposium on Security and Privacy*, pp. 280–289. IEEE Computer Society Press.

Walfish, M., M. Vutukuru, H. Balakrishnan, D. Karger, and S. Shenker (2006). Ddos defense by offense. In *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, pp. 303–314, Pisa, Italy. ACM New York, NY, USA.

Wang, Y., Y. Sure, R. Stevens, and A. Rector (2006). Knowledge elicitation plug-in for protege: Card sorting and laddering. In Mizoguchi, R., Z. Shi, , and F. Giunchiglia, editors, *Asian Semantic Web Conference*, Vol. 4185 of *LNCS*, pp. 552–565, Beijing, China. Springer-Verlag Berlin Heidelberg.

Waterman, D. (1986). *A Guide to Expert Systems*. Addison-Wesley, Reading.

Wikipedia (2006a). Man-in-the-middle attack. Retrieved December, 2006, from http://en.wikipedia.org/wiki/Man_in_the_middle_attack.

Wikipedia (2006b). Principle component analysis. Retrieved January, 2006, from http://en.wikipedia.org/wiki/Principal_components_analysis.

Wikipedia (2006c). Spoofing attack. Retrieved December, 2006, from http://en.wikipedia.org/wiki/Spoofing_attack.

Wikipedia (2007a). Correlation. Retrieved January, 2007, from http://en.wikipedia.org/wiki/Correlation.

Wikipedia (2007b). Hashimoto's thyroiditis. Retrieved March, 2007, from http://en.wikipedia.org/wiki/Hashimoto's_thyroiditis.

Wikipedia (2007c). Snort (software). Retrieved November, 2007, from http://en.wikipedia.org/wiki/Snort_(software).

Winograd, T. and F. Flores (1987). *Understanding Computers and Cognition: A New Foundation for Design*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Witten, I.H. and E. Frank (2005). *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, 2nd edition.

Yamanishi, K. and J. Takeuchi (2001). Discovering outlier filtering rules from unlabeled data. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 389–394, Sanfrancisco California.

Yamanishi, K., J. Takeuchi, and G. Williams (2000). On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 320–324, Boston, Massachusetts, US. ACM Press New York, USA.

Yang, X., D. Wetherall, and T. Anderson (2005). A dos-limiting network architecture. In *Proceedings of the 2005 conference on Applications, technologies,*

architectures, and protocols for computer communications, pp. 241–252, Philadelphia, Pennsylvania, USA. ACM New York, NY, USA.

Yoshida, T., H. Motoda, T. Wada, and T. Washio (2002). Adaptive ripple down rules method based on minimum description length principle. In *Proceedings the 2002 IEEE International Conference on Data Mining (ICDM'02)*, pp. 530–537. IEEE Computer Society.

Yoshida, T., H. Motoda, T. Wada, and T. Washio (2004). Adaptive ripple down rules method based on minimum description length principle. *Intelligent Data Analysis* 8: 239–265.