MACHINE LEARNING FOR NETWORK BASED INTRUSION DETECTION

An Investigation into Discrepancies in Findings with the KDD Cup '99 Data Set and Multi-Objective Evolution of Neural Network Classifier Ensembles for Imbalanced Data

VEGARD ENGEN

JUNE 2010

A thesis submitted in partial fulfilment of the requirements of Bournemouth University
for the degree of Doctor of Philosophy

Copyright statement

Author's declaration

The work contained in this thesis is the result of my own investigations and has not been accepted nor concurrently submitted in candidature for any other award.

Abstract

For the last decade it has become commonplace to evaluate machine learning techniques for network based intrusion detection on the KDD Cup '99 data set. This data set has served well to demonstrate that machine learning can be useful in intrusion detection. However, it has undergone some criticism in the literature, and it is out of date. Therefore, some researchers question the validity of the findings reported based on this data set. Furthermore, as identified in this thesis, there are also discrepancies in the findings reported in the literature. In some cases the results are contradictory. Consequently, it is difficult to analyse the current body of research to determine the value in the findings.

This thesis reports on an empirical investigation to determine the underlying causes of the discrepancies. Several methodological factors, such as choice of data subset, validation method and data preprocessing, are identified and are found to affect the results significantly. These findings have also enabled a better interpretation of the current body of research. Furthermore, the criticisms in the literature are addressed and future use of the data set is discussed, which is important since researchers continue to use it due to a lack of better publicly available alternatives.

Due to the nature of the intrusion detection domain, there is an extreme imbalance among the classes in the KDD Cup '99 data set, which poses a significant challenge to machine learning. In other domains, researchers have demonstrated that well known techniques such as Artificial Neural Networks (ANNs) and Decision Trees (DTs) often fail to learn the minor class(es) due to class imbalance. However, this has not been recognized as an issue in intrusion detection previously. This thesis reports on an empirical investigation that demonstrates that it is the class imbalance that causes the poor detection of some classes of intrusion reported in the literature.

An alternative approach to training ANNs is proposed in this thesis, using Genetic Algorithms (GAs) to evolve the weights of the ANNs, referred to as an Evolutionary Neural Network (ENN). When employing evaluation functions that calculate the fitness proportionally to the instances of each class, thereby avoiding a bias towards the major class(es) in the data set, significantly improved true positive rates are obtained whilst maintaining a low false positive rate. These findings demonstrate that the issues of learning from imbalanced data are not due to limitations of the ANNs; rather the training algorithm. Moreover, the ENN is capable of detecting a class of intrusion that has been reported in the literature to be undetectable by ANNs.

One limitation of the ENN is a lack of control of the classification trade-off the ANNs obtain. This is identified as a general issue with current approaches to creating classifiers. Striving to create a single best classifier that obtains the highest accuracy may give an unfruitful classification trade-off, which is demonstrated clearly in this thesis. Therefore, an extension of the ENN is proposed, using a Multi-Objective GA (MOGA), which treats the classification rate on each class as a separate objective. This approach produces a Pareto front of non-dominated solutions that exhibit different classification trade-offs, from which the user can select one with the desired properties.

The multi-objective approach is also utilised to evolve classifier ensembles, which yields an improved Pareto front of solutions. Furthermore, the selection of classifier members for the ensembles is investigated, demonstrating how this affects the performance of the resultant ensembles. This is a key to explaining why some classifier combinations fail to give fruitful solutions.

Publications

The majority of the work in this thesis has been published or accepted for publication. The first empirical part of the thesis has been accepted for publication in the International Journal of Intelligent Data Analysis (Engen *et al.* 2011), which explores the discrepancies in the results reported in the literature with the KDD Cup '99 data set. The focus of the second part of this thesis, and another journal paper published in the International Journal of Knowledge-Based and Intelligent Engineering Systems (Engen *et al.* 2008), is on learning from imbalanced data. For this, an Evolutionary Neural Network was proposed, which successfully learns from imbalanced data due to the bespoke fitness functions that are employed. Two extensions to this approach have been developed in the third part of this thesis, proposing a method that performs **M**ulti Objective Evolution of **A**rtificial Neural Network Ensem**ble**s (MABLE), which has been published in the International Conference on Machine Learning and Cybernetics (Engen *et al.* 2009).

V. Engen, J. Vincent and K. Phalp. December 2008. Enhancing Network Based Intrusion Detection for Imbalanced Data. International Journal of Knowledge-Based and Intelligent Engineering Systems (KES), 12, 357–367.

V. Engen, J. Vincent, A.C. Schierz and K. Phalp. Multi-Objective Evolution of the Pareto Optimal Set of Neural Network Classifier Ensembles. In Proceedings of the International Conference on Machine Learning and Cybernetics (ICMLC), volume 1, pages 74–79, Baoding, China, July 2009. IEEE.

V. Engen, J. Vincent and K. Phalp. 2011. Exploring Discrepancies in Findings Obtained with the KDD Cup '99 Data Set. To appear in the International Journal of Intelligent Data Analysis, 15.

Acknowledgments

This PhD journey has certainly been a learning experience, on many levels, and there are many people who have had an impact in one way or another. There are so many people that deserve a 'thank you' here, so don't feel offended if your name is not mentioned. More importantly, this list is not ordered by importance. On a final note, since this is a personal part of the thesis, I *will* contract words and I'll leave in a 'blabla'![1]

To Jonathan: Thank you for continuing to supervise me even though you don't receive official credit for it. Your contributions have been invaluable and you never fail to impress me with your wealth of knowledge. I will miss working with you, but I expect there will be times for more interesting discussions.

To Keith: Thank you for your honesty and being there to talk to and giving me guidance whenever it was necessary.

To Christiane: Thank you for your encouragement and support throughout the PhD, even though the topic was actually banned. It was really interesting to finally learn what you were doing in the end! Your input has always been invaluable and it's been great to get a different perspective on things.

To Jo and Lindsay: Thank you for your support, positive outlook on life, honesty and for always having an open door and being up for a chat. You represent what I think is great about Bournemouth University.

To Sherry and Mel: Thank you for being such wonderful people, making the research centre something special. To me, you have been the glue that has kept things together, and you deserve more credit for the work you do.

To Laura: Thank you for your unconditional support, understanding and patience, particularly towards the end of the PhD when I was stressed and distracted by the research and other work.

To Mario: Thank you for all our interesting discussions, and for being such a close friend still despite the distance.

To Paul and Martine: Thank you for giving a balance to my life of science, offering your friendship and sharing your amazing view on life.

And last, but not least, to my family: Thank you for your unconditional love and patience since it went longer and longer between our talks. Thanks for all the emails and text messages asking if I was still alive! I would like to extend this thank you to Paul and Vivianne who have been like a second family to me here in England. You are such great people and I don't know of anybody more hospitable!

I would like to round this section up with a geeky comic (Figure 1), which I found to be very suitable.

[1] I have tended to write 'blabla' in cases where I couldn't immediately remember the correct word to use. This is all well, allowing me to continue with the flow of writing. However, I once forgot to change a 'blabla', which made it into the introduction of my transfer report for this PhD project.

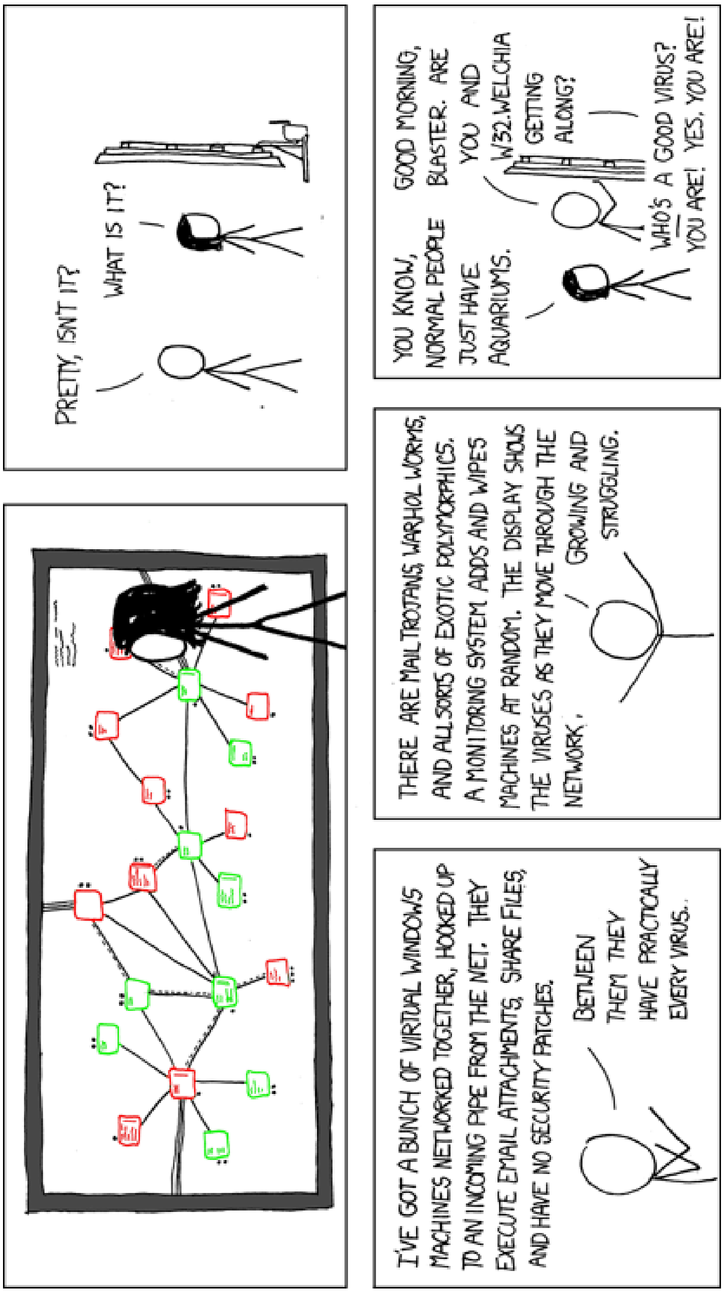

I'VE GOT A BUNCH OF VIRTUAL WINDOWS MACHINES NETWORKED TOGETHER, HOOKED UP TO AN INCOMING PIPE FROM THE NET. THEY EXECUTE EMAIL ATTACHMENTS, SHARE FILES, AND HAVE NO SECURITY PATCHES.
BETWEEN THEM THEY HAVE PRACTICALLY EVERY VIRUS.
THERE ARE MAIL TROJANS, WARHOL WORMS, AND ALL SORTS OF EXOTIC POLYMORPHICS. A MONITORING SYSTEM ADDS AND WIPES MACHINES AT RANDOM. THE DISPLAY SHOWS THE VIRUSES AS THEY MOVE THROUGH THE NETWORK,
GROWING AND STRUGGLING.
YOU KNOW, NORMAL PEOPLE JUST HAVE AQUARIUMS.
GOOD MORNING, BLASTER. ARE YOU AND W32.WELCHIA GETTING ALONG?
WHO'S A GOOD VIRUS? YOU ARE! YES, YOU ARE!
PRETTY, ISNT IT?
WHAT IS IT?

Figure 1: *Network* (Munroe 2007).

Contents

List of Abbreviations

ABN	Adaptive Bayesian Network
ACO	Ant Colony Optimisation
AdaBoost	Adaptive Boosting
AI	Artificial Intelligence
AIS	Artificial Immune System
ANN	Artificial Neural Network
AUC	Area Under the Curve
Bagging	Bootstrap aggregating
BN	Bayesian Network
BSM	Solaris Basic Security Module
CART	Classification And Regression Tree
CBR	Case Based Reasoning
CF	Confidence Factor
CHAID	Chi-squared Automatic Interaction Detector
CMAC	Cerebellar Model Articulation Controller
CV	Cross validation
DARPA	Defence Advanced Research Projects Agency
DDoS	Distributed Denial of Service
DNS	Domain Name Server
DoS	Denial of Service
DT	Decision Tree
ENN	Evolutionary Neural Network
FN	False Negative
FNR	False Negative Rate
FP	False Positive

FPR	False Positive Rate
FSM	Finite State Machine
GA	Genetic Algorithm
GP	Genetic Programming
HMM	Hidden Markov Model
HN	Hidden Neurons (the number of neurons in the hidden layer of an MLP)
IBL	Instance Based Learning
IDS	Intrusion Detection System
KBS	Knowledge Based System
KDD	Knowledge Discoverey and Datamining
k-NN	*k* Nearest Neighbour
LAN	Local Area Network
MABLE	**M**ulti-Objective Evolution of **A**rtificial Neural Network Classifier Ensem**ble**s
MA	Mobile Agent
MIMD	Multiple Instructions, Multiple Data
ML	Machine Learning
MLP	Multi Layer Perceptron
MOGA	Multi-Objective Genetic Algorithm
MSE	Mean of Squared Errors
NB	naïve Bayes
NCL	Negative Correlation Learning
NPGA	Niched Pareto Genetic Algorithm
NSGA	Non-dominated Sorting Genetic Algorithm
PCA	Principal Component Analysis
PSO	Particle Swarm Optimisation
RBFN	Radial Basis Function Network
RBF	Radial Basis Function
RBS	Rule Based System
REP	Reduced Error Pruning
ROC	Receiver Operator Curve
SIMD	Single Instructions, Multiple Data
SPMD	Single Program, Multiple Data
SSH	Secure Shell
SSL	Secure Socket Layer
SVM	Support Vector Machine

TNR	True Negative Rate
TN	True Negative
TPR	True Positive Rate
TP	True Positive
TTL	Time To Live
VEGA	Vector Evaluated Genetic Algorithm

CHAPTER 1

Introduction

This thesis considers the application of machine learning to network based intrusion detection. Section 1.1 outlines the background and motivation for the work presented in this thesis. The research questions are discussed in Section 1.2. The contributions and novelty of the work are discussed in Section 1.3, followed by an outline of the structure of the remainder of the thesis in Section 1.4.

1.1 Background and motivation

Computer security is important in our society, which has an ever growing use of computer technology, both for work and personal use. As of 2008, 65% of all households in the U.K. were connected to the Internet, which is approximately 16 million households (National Statistics 2008). Furthermore, the amount of computer malware[1] has increased rapidly in recent years; *"from about 333,000 in 2005 to 972,000 in 2006, and 5,490,000 in 2007" (Marx 2008).*

Anybody using a computer is at some risk of intrusion, even if the computer is not connected to the Internet or any other network (*i.e.* through physical access). If the computer is left unattended, any person can attempt to access and misuse the system. The problem is, however, far greater if the computer is connected to a network, particularly the Internet. Any user from around the world can reach the computer remotely (to some capacity) and may attempt to access private/confidential information or to launch some form of attack to bring the system to a halt or cease to function effectively.

An intrusion to a computer system does not need to be executed manually by a person. It may be executed automatically with engineered software. A well known example of this is the Slammer worm (also known as Sapphire), which performed a global Denial of Service (DoS) attack in 2003. The worm exploited a vulnerability in Microsoft's SQL Server, which allowed it to disable database servers and overload networks (Moore *et al.* 2003). Moore *et al.* refer to Slammer as *"the fastest computer worm in history"*, which infected approximately 75,000 computer systems around the world within 10 minutes. Not only did the Slammer worm restrict the the general Internet traffic, it *"caused network outages and unforeseen consequences such as canceled airline flights, interference with elections, and ATM failures*" (Moore *et al.* 2003).

[1] Malware is a term for malicious software; *i.e.*, software that is intentionally harmful to a computer or computer system.

A private person may not have much at stake if s/he is targeted by a ‘cyber attack’, but it is a serious threat to professional companies and government organisations. A survey by the Web Application Security Consortium (2008) revealed that 67% of attacks in 2007 were profit motivated. There are many examples in recent news of cyber attacks. For example, early in 2009, it was revealed that the US power grid had been infiltrated by an intruder, leaving malware that was capable of shutting down the entire grid (BBC 2009c). Later that year, a major spy network (GhostNet) was discovered (BBC 2009b). GhostNet was said to be mainly located in China, which is claimed to have infiltrated more than 1000 computers around the world, with victims such as foreign ministers and embassies. Another government related incident was reported in 2008, when the Russian military was accused of launching DoS attacks against Georgia during the war over South Ossetia (CNet news 2008). From these examples, it is clear that cyber attacks can threaten national security, prompting President Barack Obama to initiate a national cyber security body in the USA in May 2009 (BBC 2009d), followed shortly by the UK (BBC 2009a).

There are several mechanisms that can be adopted to increase the security in computer systems. Kruegel *et al.* (2004, pp. 11–17) consider three levels protection:

Attack prevention: Firewalls, user names and passwords, and user rights.

Attack avoidance: Encryption.

Attack detection: Intrusion detection systems.

Despite adopting mechanisms such as cryptography and protocols to control the communication between computers (and users), it is impossible to prevent all intrusions (Gollmann 2006, pp. 251–252). Firewalls serve to block and filter certain types of data or services from users on a host computer or a network of computers, aiming to stop some potential misuse by enforcing restrictions. However, firewalls are unable to handle any form of misuse occurring within the network or on a host computer. Furthermore, intrusions can occur in traffic that appears normal (Kruegel *et al.* 2004, Gollmann 2006, pp. 251–252). Intrusion Detection Systems (IDSs) do not replace the other security mechanisms, but compliment them by attempting to detect when malicious behaviour occurs.

The purpose of an Intrusion Detection Systems (IDS), in general terms, is to detect when the behaviour of a user conflicts with the intended use of the computer, or computer network, *e.g.*, committing fraud, hacking into the system to steal information, conducting an attack to prevent the system from functioning properly or even break down. Before the 1990s, the intrusion detection was performed by system administrators, manually analysing logs of user behaviour and system messages, with poor chances of being able to detect intrusions in progress (Kemmerer and Vigna 2002). This has gradually changed, with early works of Anderson (1980) and Denning (1987), by developing software to automatically analyse the data for the system administrators. The first IDS to achieve this in real-time was developed in the early 1990s (Kemmerer and Vigna 2002). However, due to the increased used of computers, the magnitude of data in contemporary computer networks still renders this a significant challenge.

A wide range of Artificial Intelligence (AI) techniques have been adopted in IDSs, as reviewed in Chapter 3. Initially, Rule Based Systems (RBSs) were the first to be employed successfully, and are still at the core of many IDSs. This allows for IDSs that automatically filter network traffic and/or analyse user data to identify patterns of known intrusions. Suspected intrusions can be reported to an administrator in a detailed, informative way, explaining the rules that lead to the intrusion alert. A general drawback of RBSs is that they are inflexible (due to the rigid rules), and, thus, cannot detect new intrusions, or variations of known intrusions (Lewis 1993, Owens and Levary 2006).

Another area of AI, machine learning / data mining, with techniques such as artificial neural networks and clustering, offers some desired flexibility, which has been of focus of much research in the past decade. These techniques are often employed as classifiers that learn to perform intrusion detection automatically from a training set with examples of user behaviour or network traffic. Hence, there is no need to extract knowledge from a human expert and formulate this knowledge into a rules that can represent attacks. A benefit of machine learning is that the techniques are capable of generalising from known attacks to variations thereof, or even detecting entirely new types of intrusion.

The type of intrusion detection referred to thus far is *misuse detection*, in which the IDS scans for known (or learned) attacks. Alternatively, machine learning allows for another method of detection, *anomaly detection*. The machine learning techniques can learn what constitutes normal behaviour, from which all unknown behaviour is considered a potential intrusion. Hence, such systems are capable of detecting entirely new types of attack. The trade-off, however, is an increased number of false alerts (false positives) (Dokas *et al.* 2002, Kruegel *et al.* 2004), as discussed further in Section 2.3.

Recent research focuses more on the hybridisation of techniques to improve the detection rates of machine learning classifiers. For example, Sabhnani and Serpen (2003) examine the performance of 9 machine learning algorithms on a commonly used data set, the KDD Cup '99 data set (The UCI KDD Archive 1999). First, they found that different techniques performed better on different classes of intrusion. Second, they found that combining the best techniques for each class improved the overall performance of the detector. However, there are discrepancies in the findings reported in the literature as to how well different techniques perform on the different classes of intrusion.

The KDD Cup '99 data set has been widely used to evaluate intrusion detection prototypes in the last decade. Although many researchers apply the same machine learning techniques to the data set, contradictory findings have been reported in the literature. Furthermore, some researchers have published criticisms of the data set[2], questioning the validity of the results obtained with this data (Bouzida and Cuppens 2006a;b, Brugger 2007a, Mahoney and Chan 2003, McHugh 2000, Sabhnani and Serpen 2004). Despite the criticisms, researchers continue to use the data due to a lack of better publicly available alternatives. Hence, it is important to identify the value of the data set and the findings from the extensive body of research based on it, which has largely been ignored by the existing critiques. This is the focus of the first part of this thesis, along with determining the underlying causes of the discrepancies, considered in Chapter 4.

Although there are discrepancies in the findings in the literature, all studies indicate that there is a significant problem in detecting two particular classes of intrusion: *User to Root (U2R)* and *Remote to Local (R2L)*[3]. There are methodological factors that affect these findings, which are uncovered in the first part of this thesis. However, the attacks remain challenging to detect, which is investigated further in the second part of this thesis. It is hypothesised that the class imbalance in the data set causes poor detection of these minor classes, which is a challenge to machine learning that has not been considered for intrusion detection previously.

Class imbalance is a problem in many real life applications, and has been considered in, for example, medical diagnosis (Cohen *et al.* 2006, Mazurowski *et al.* 2008, Mena and Gonzalez 2006), credit scoring (Huang *et al.* 2006), customer churn (Burez and van den Poel 2009, Xie *et al.* 2009), natural language processing (Kobyliński and Przepiórkowski 2008), lexical acquisition (Kermanidis *et al.* 2004) and text recognition (Stamatatos 2008). The general problem that has been observed in the cited literature, is that the minor class(es) are not classified well when there is a significant imbalance among the classes. Artificial Neural Networks (ANNs) and Decision Trees (DTs) have been popularly applied to intrusion detection, but both have been shown to be biased towards the major class(es) (Chawla 2003, Jo and Japkowicz 2004), which, in some cases, can lead to the minor class(es) even being 'ignored'. This corresponds with the observations in the literature on intrusion detection, in which ANNs have been reported to be unable to detect the minor class *U2R* (Bouzida and Cuppens 2006a;b). To verify this observation, an empirical investiga-

[2] The criticisms are mainly of the DARPA data (Lippmann *et al.* 2000a;b) from which the KDD Cup '99 data has been derived. These criticisms are discussed further in Section 4.2 on page 57.

[3] A description of these attacks is given in Section 2.2 on page 8.

tion has been conducted for the second part of this thesis. Furthermore, an alternative approach to training ANNs is proposed to better learn from imbalanced data.

It is hypothesised here that ANNs are capable of learning from imbalanced data, given more appropriate training. This thesis focuses specifically on Multi Layer Perceptrons, which are commonly trained by the backpropagation algorithm. This training algorithm aims to minimise the error of the classifier. Consequently, the algorithm can achieve a very low error by only correctly classifying the major class(es), which is why the minor class(es) can simply be ignored. The approach proposed in the second part of this thesis trains MLPs by evolving their weights with a Genetic Algorithm (GA), referred to as an Evolutionary Neural Network (ENN). Several fitness functions are examined, some of which are employed to demonstrate the issues related to learning by minimising the classification errors, which is essentially what the commonly used backpropagation algorithm does, and others that avoid this bias towards the major class.

The ENN successfully learns from imbalanced data, which demonstrates that MLPs are capable of detecting the minor classes with more appropriate training. A third part of this thesis extends the work in part two, proposing a new approach that has two pragmatic aims: (1) to improve on the performance of the ENN, and (2) to offer the user a range of solutions with different classification trade-offs. The latter is desirable since the ENN only offers one solution that may have an inadequate trade-off in performance, *e.g.*, too many false positives, although a high true positive rate is achieved. This is identified as a general problem with existing machine learning algorithms that only produce a single solution. Different solutions may be obtained by changing the training data or configuration parameters, or assigning weights to the classes *a priori*. However, this is generally an *ad hoc* process, which, unless optimal weights are known, is not likely to offer the user the ideal trade-off. The approach taken here is to train the MLPs with a Multi-Objective Genetic Algorithm (MOGA), treating the classification rates of each class as a separate objective. The MOGA is capable of evolving a set of MLPs that exhibit different classification trade-offs, offering the user several solutions that are non-inferior with respect to each other.

Extending the ENN to perform multi-objective optimisation does not necessarily improve the performance of the MLPs. However, since we obtain a population of MLPs with different classification trade-offs, this can be exploited to perform classifier combination. The rationale for a combination is that the aggregated knowledge in the pool of base classifiers is important, which can lead to better results compared to the single best classifier (Kuncheva 2004, Quinlan 1996, Yao and Liu 1996). Contrary to current methods of creating classifier ensembles, which generate only a single ensemble, the approach taken here evolves a set of ensembles with different classification trade-offs. A single ensemble may improve on the performance of a small set of base classifiers, but is likely to be inferior to a large number of other base classifiers that exhibit a better classification trade-off for a given application. A MOGA is employed in this phase, which optimises the selection of base classifiers to form ensembles. This gives a new set of solutions (classifier ensembles), which improve on the performance of the base classifiers. Furthermore, the approach taken here gives a novel perspective on the analysis of the selection process for classifier ensembles.

1.2 Research questions and constraints

As the research presented in this thesis developed, it naturally formed three main parts: analysis of the KDD cup '99 data set, ENNs for improved performance on imbalanced data, and multi-objective evolution of ANN classifiers and classifier combinations. Specific aims and objectives related to each of these parts of the thesis are presents in their respective chapters, whilst general research questions are presented here at a high level.

- What has caused the contradictory findings with the KDD Cup '99 data set reported in the literature?

- In light of criticisms of the KDD Cup '99 data set in the literature, can it be used in the future to give valuable contributions to the intrusion detection and machine learning domains?
- Is the poor detection of some classes of intrusion due to issues with learning from imbalanced data?
- How can one utilise machine learning to better learn from imbalanced data?
- Can classifier combination be adopted to better learn from imbalanced data?
- How does the selection of base classifiers affect the performance of the resultant ensemble?

The scope of this thesis has been determined by a number of pragmatic constraints, which have been applied to ensure a focused investigation without compromising the ability to answer the research questions. This is necessary since the thesis considers several large research domains. The majority of the constraints apply to intrusion detection. First, the empirical work in this thesis only considers network based misuse detection. However, the review of the domain considers all the main methods of intrusion detection. Although some challenges and concepts apply to fraud detection and fault localisation, these applications are excluded. Furthermore, since the focus of this investigation is on machine learning, other, conventional, techniques applied to intrusion detection are not considered. However, the review does consider a broad range of other AI techniques applied to intrusion detection.

There are several aspects of intrusion detection that are not considered here, although they would require attention when developing an IDS to be deployed in real life, such as:

Architecture: the focus here is on what could be referred to as a detection module that would exist in a larger IDS framework. Especially in wireless and mobile *ad hoc* networks, the architecture is very important. This includes determining where to deploy the IDS, which is considered a general challenge (Kruegel *et al.* 2004, p. 27).

Data collection: since the KDD Cup '99 data set is adopted in this work, data collection is not required. It would, however, be necessary to collect data from the environment in which an IDS is to be employed. This also includes a process of labeling data for supervised learning.

Data preprocessing: some data preprocessing is necessary in this work, mainly for the MLPs that are adopted, enumerating and scaling feature values. However, the KDD Cup '99 data set has already undergone an initial preprocessing task by transforming the raw *tcpdump* from the DARPA data into a feature set suitable for machine learning. Although the availability of this data set is very convenient for researchers, criticism in the literature indicates that the transformation was not ideal (Bouzida and Cuppens 2006a, Bouzida 2006).

Performance: there are several mechanisms that can be adopted to help achieve a better performing IDS, in terms of detection rates, speed and memory usage, *e.g.*, feature selection and data sampling. Related to the data transformation discussed above, different transformations and feature sets may facilitate improved intrusion detection. There are also different ways of preprocessing the data for the MLPs adopted in this study, which may be considered better and could yield improved performance. However, the focus of this thesis is on the issues and challenges posed by the research questions, not to develop an optimal IDS prototype.

Other pragmatic considerations: detecting new intrusions will always be a challenge; there will always be new software, which inevitably have vulnerabilities that can be exploited. Therefore, re-training is necessary once new data is available. When and how this is done is not considered here. Neither is online training or unsupervised learning.

1.3 Contribution and novelty

As introduced above, there are three empirical parts to this thesis. These parts have made contributions to both the intrusion detection and machine learning domains. Although the focus of this thesis is on the application of machine learning to intrusion detection, several contributions have been made to the general machine learning domain.

The first part of the thesis makes the greatest contribution to the intrusion detection domain. First, identifying discrepancies in the findings reported in the literature. This has led to an empirical investigation of the KDD Cup '99 data set, which has uncovered several underlying causes of the discrepancies. Furthermore, an important contribution of this part of the thesis is a discussion of, and recommendations for, future research using this data set.

Learning from imbalanced data has been identified as one of the reasons for poor detection of certain classes of intrusion. This has not been considered an issue in this domain previously, and the empirical research conducted in the second part of the thesis demonstrates how commonly adopted techniques such as ANNs and DTs perform poorly for this reason. An alternative approach to training ANNs has been proposed, which demonstrates that the issues with learning from imbalanced data are due to using accuracy or a general measure of error as a performance metric in the training process. Offering a different approach to training, this research demonstrates that ANNs are capable of learning from imbalanced data, and, therefore, detecting more intrusions.

The proposed method of training ANNs is useful in any domain for which there is a class imbalance. However, a limitation to the approach has been identified. Although the approach is unbiased to the class balance, there is no control of the classification trade-off the resultant ANN obtains. This trade-off problem has not been considered previously in the literature in the current classifier and classifier combination approaches. Addressing this is, therefore, an important contribution to classification research. Furthermore, a novel approach to evolving ANNs and classifier ensembles has been proposed, which successfully learns from imbalanced data and offers the user a wide range of solutions that exhibit different classification trade-offs. From this, the user can select the solution that gives the best trade-off for the particular application.

1.4 Structure

The remainder of this thesis is organised as follows. Chapter 2 provides an introduction to the domain of intrusion detection, which is followed by a review of applications of artificial intelligence to intrusion detection systems in Chapter 3. This review uncovered discrepancies in the findings reported in the literature, which forms the focus of Chapter 4. Among the causes of the discrepancies, class imbalance was found to be an issue not previously considered in this domain, which forms the focus of the two consecutive investigations reported in this thesis. First, Chapter 5 seeks to demonstrate empirically that imbalance is a reason for poor detection of some intrusions, and proposes a novel approach to learning from imbalanced data. Thereafter, Chapter 6 improves upon this approach, considering some pragmatic limitations of the single-objective approach proposed in Chapter 5. Chapter 6 also considers classifier combination, which is widely employed in recent literature to improve upon the performance of single classifiers. Chapter 7 concludes and offers suggestions for further work.

CHAPTER 2

Intrusion detection

This chapter provides an introduction to intrusion detection. Related terminology and definitions that are used in this thesis are presented in Section 2.1. Section 2.2 presents an overview of potential intrusions to computers and computer networks. A taxonomy of intrusion detection systems is provided in Section 2.3, which includes a discussion of the main approaches to detecting intrusions. As an emerging area of research, intrusion detection in wireless and *ad hoc* mobile and sensor networks is discussed in Section 2.4. The main areas of research in intrusion detection are discussed in Section 2.5.

2.1 Definitions and terminology

Intrusion detection is the process of monitoring and analysing events that occur in a computer or networked computer system to detect behaviour of users that conflict with the intended use of the system. An Intrusion Detection System (IDS) employs techniques for modelling and recognising intrusive behaviour in a computer system. The term 'intrusive behaviour' is discussed further in Section 2.2.

When referring to the performance of IDSs, the following terms are often used when discussing their capabilities:

True positive (TP): classifying an intrusion as an intrusion. The true positive rate is synonymous with *detection rate*, *sensitivity* and *recall*, which are other terms often used in the literature.

False positive (FP): incorrectly classifying normal data as an intrusion. Also known as a *false alarm*.

True negative (TN): correctly classifying normal data as normal. The true negative rate is also referred to as *specificity*.

False negative (FN): incorrectly classifying an intrusion as normal.

The performance metrics calculated from these are:

$$True\,positive\,rate\,(TPR) = \frac{TP}{TP+FN} = \frac{\#correct\,intrusions}{\#intrusions} \tag{2.1}$$

$$False\,positive\,rate\,(FPR) = \frac{FP}{TN+FP} = \frac{\#normal\,as\,intrusions}{\#normal} \tag{2.2}$$

$$True\,negative\,rate\,(TNR) = \frac{TN}{TN+FP} = \frac{\#correct\,normal}{\#normal} \tag{2.3}$$

$$False\,negative\,rate\,(FNR) = \frac{FN}{TP+FN} = \frac{\#intrusions\,as\,normal}{\#intrusions} \tag{2.4}$$

Two additional performance metrics are also commonly used, referred to as *accuracy* and *precision*:

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} = \frac{\#correct\,classifications}{\#all\,instances} \tag{2.5}$$

$$Precision = \frac{TP}{TP+FP} = \frac{\#correct\,intrusions}{\#instances\,classified\,as\,intrusion} \tag{2.6}$$

Accuracy is also referred to as an *overall* classification rate, and according to Wu and Yen (2009), precision is also referred to as *recall*. Due to the direct nature of many intrusions, the terms 'intrusion' and 'attack' are used interchangeably.

2.2 Intrusions

In general terms, intrusive behaviour can be considered as any behaviour that deviates from normal, expected, use of the system. Intrusion detection shares many of the challenges of fraud detection and fault management/localisation. Although, these are not considered here, there is a natural overlap between these domains, especially for event correlation (Section 2.3.3.1 on page 12).

There are many types of intrusion, which makes it difficult to give a single definition of the term. Asaka *et al.* (1999) and Kruegel *et al.* (2004, p. 4) offer the following breakdown of a successful intrusion:

Surveillance/probing stage: The intruder attempts to gather information about potential target computers by scanning for vulnerabilities in software and configurations that can be exploited. This includes password cracking.

Activity (exploitation) stage: Once weaknesses have been identified in the previous stage, they can be exploited to obtain administrator rights to the selected host(s). This will give the intruder free access to violate the system. This stage may also include *Denial of Service (DoS)* attacks, as detailed further below.

Mark stage: After the exploitation stage, the attacker may be free to steal information from the system, destroy data (including logs that may reveal that the attack took place), plant a virus or spyware software, or use the host as a medium for conducting further attacks. After which, this marks the stage where the attacker has achieved his or her goal(s) of the attack (Asaka *et al.* 1999).

Masquerading stage: In this final stage, the intruder will attempt to remove traces of the attack by, for example, deleting log entries that reveal the intrusion.

The two first stages are further refined into an attack taxonomy that is widely adopted in the literature to classify attacks when evaluating IDSs, which considers four categories of intrusion (Kendall 1999, Lippmann *et al.* 2000a):

Probing (surveillance): Same as the first stage above.

Denial of Service (DoS): The general purpose of *DoS* attacks is to interrupt some service on a host to prevent it from dealing with certain requests. This may be a step in a multi-stage attack, such as the *Mitnick* attack which is described below, or to be destructive to 'crash' a host or prevent it from functioning properly, which is the purpose of the Slammer worm previously discussed in Section 1.1. Kendall (1999) describes three types of *DoS* attacks, those that (1) *"abuse legitimate features"*, (2) *"create malformed packets that confuse the TCP/IP stack of the machine that is trying to reconstruct the packet"*, or (3) *"take advantage of bugs in a particular network daemon"*.

User to Root (U2R): These attacks exploit vulnerabilities in operating systems and software to obtain root (administrator) access to the system. For example, Kendall (1999) describes the buffer overflow attack: "*Buffer overflows occur when a program copies too much data into a static buffer without checking to make sure that the data will fit. For example, if a program expects the user to input the user's first name, the programmer must decide how many characters that first name buffer will require. Assume the program allocates 20 characters for the first name buffer. Now, suppose the user's first name has 35 characters. The last 15 characters will overflow the name buffer. When this overflow occurs, the last 15 characters are placed on the stack, overwriting the next set of instructions that was to be executed. By carefully manipulating the data that overflows onto the stack, an attacker can cause arbitrary commands to be executed by the operating system.*"

Remote to Local (R2L): There are some similarities between this class of intrusion and *U2R*, as similar attacks may be carried out. However, in this case, the intruder does not have an account on the host and attempts to obtain local access across a network connection. To achieve this, the intruder can execute buffer overflow attacks, exploit misconfigurations in security policies or engage in social engineering (*i.e.*, obtaining data by tricking a human operator, rather than targeting software flaws) (Kendall 1999).

The four classes above may be used in an IDS for classifying intrusions, rather than only differentiating between 'normal' and 'intrusion'. This gives more information about the type of intrusion, which may affect the chosen method of reporting and acting on the suspected detection (see Section 2.3.4).

Single events can signify an intrusion, whilst other events are not considered an intrusion before they are observed in the context of one or more other events. This could be a repetition of the same event, as would be typical for *Probing* or *DoS* attacks, or a completely different event. An IDS should be able to recognise simple, single event, attacks as well as complex, multiple event, attacks (Benferhat *et al.* 2003). As an example of the former Benferhat *et al*. (2003) mention *ping of death*, which is a *DoS* attack where the attacker sends a too large ping package to a host, which may cause it to crash. As an example of the latter, they describe the *Mitnick* attack, which consists of the following steps:

1. An intruder floods the login port a host computer *H* so that it cannot respond to any other requests.
2. The intruder uses *H*'s IP address to send spoofed messages to a server *S*. *S* returns messages to *H*, and normally *H* would return messages to close the connection. However, since *H* is unable to respond, the connection remains open.
3. After being able to open the connection to S, the intruder can attempt further ingress to exploit the system.

Performing intrusion detection in wireless, mobile *ad hoc* and sensor networks have become more in focus in recent years. These networks pose additional challenges to IDSs, due to their distributed nature and *ad hoc* infrastructure (Ahmed *et al.* 2006, Brutch and Ko 2003, Zhang and Lee 2000). Additional security threats and operational considerations are necessary to take into account when deploying an IDS in such networks, which are discussed in Section 2.4 on page 14.

2.3 Intrusion detection systems

The specific architectures of IDSs are not discussed here, as these are diverse and continue to evolve with time. In general terms, Verwoerd and Hunt (2002) have identified the following common building blocks of an IDS:

Sensor probes: gather data from the system under inspection.

Monitor: receives events from a number of sensors and forwards suspicious content to a 'resolver'.

Resolver: determines a suitable response to suspicious content.

Controller: provides administrative functions.

This section focuses on characteristics of IDSs, which elaborate on the first three points above. An IDS may be described according to four characteristics, according to the taxonomy adopted in (Kruegel *et al.* 2004, pp. 20–21):

Audit source location: host based or network based.

Detection method: misuse or anomaly detection.

Behaviour on detection: passive or active.

Usage frequency: real-time or off-line.

Also considered here is 'detection approach', which describes, at a lower level, the strategies used to detect intrusions. This is related to 'detection method', and is discussed to some degree in (Kruegel *et al.* 2004, pp. 20–21) within misuse detection systems. However, the detection approach is not constrained to misuse detection, and, thus, is treated as a separate characteristic here. The five characteristics are discussed in their respective sections below.

2.3.1 Audit source location

IDSs typically operate on one of two levels: on a *host* or a *network* (Debar *et al.* 1999, Kachirski and Guba 2003). A host based IDS monitors the local behaviour on a single host (computer), generally by analysing system status/performance and logs to determine *inter alia* access violations and system file modifications. Host based systems may monitor user and/or program/process behaviour, which is discussed further in Section 2.3.2 in the context of anomaly detection. This overlaps with a third level, *application* based IDSs, included in the taxonomy of Kruegel *et al.* (2004, pp. 26–27), which detect attacks against specific applications.

There has been a trend towards network based IDSs over the last decade (Kemmerer and Vigna 2002, Kim and Bentley 2001; 2002), which analyse network traffic. However, there are IDSs that support both host based and network based intrusion detection, which is promoted by Lindqvist and Porras (2001) as they see the two complimenting each other. Lindqvist and Porras identified some limitations (and challenges) of network based IDSs, which include:

- Not being able to catch all forms of intrusion, since some may not generate network traffic.
- Dealing with encrypted data via SSL (Secure Socket Layer) connections and SSH (Secure Shell).
- Must take into account different attacks on different operating system platforms.

Lindqvist and Porras also identified the following limitations of host based IDSs:

- Can fail to observe network activity that may be a part of an attack process on the host.
- If an attacker obtains root/administrator access, the system is particularly vulnerable, especially due to the possibility to turn off the IDS.

From the points above is it clear how the two types of IDS can complement each other, mainly with respect to achieving a broader coverage for detection.

2.3.2 Detection method

There are two main detection methods, referred to as misuse detection and anomaly detection (Endler 1998, Gollmann 2006, Kemmerer and Vigna 2002, Lee and Xiang 2001). These terms are also known as knowledge based and behaviour based intrusion detection (Debar *et al.* 1999, Debar 2000). The former attempts to encode knowledge of known intrusions (misuses), typically as rules, and use this to screen events (also known as a signature based IDS). The latter attempts to 'learn' the features of event patterns that constitute normal behaviour, and, by observing patterns that deviate from established norms, detect when an intrusion has occurred (Denning 1987). Some IDSs offer both capabilities, typically via a hybridisation of techniques, see for example (Depren *et al.* 2005, Endler 1998). However, a system may also be modelled according to both normal and intrusive data, which has become a common approach in recent research adopting machine learning techniques (Bouzida and Cuppens 2006a;b, Depren *et al.* 2005, Panda and Patra 2007; 2009, Sabhnani and Serpen 2003, Xiang *et al.* 2008, Zhang and Zulkernine 2006).

Misuse detection is successful in commercial intrusion detection, according to Gollman (2006, pp. 252–253), who states that *"at the time of writing* [2005]*, all commercial IDS products were based on misuse detection"*. However, this approach cannot detect attacks for which it has not been programmed, and, thus, it is prone to issue false negatives if the system is not kept up to date with the latest intrusions (Gollmann 2006, pp. 251–252, Lewis 1993). On the other hand, misuse detection systems generally produce few false positives (Kruegel *et al.* 2004).

The general perception about misuse detection, as presented above, is no longer entirely accurate. In recent years, researchers have incorporated techniques that allow misuse detection systems to be more flexible, being capable of detecting more variations of attacks. This has been made possible with machine learning techniques such as Artificial Neural Networks, which are built to be able to generalise their models of known attacks to classify unseen cases. This is also the case for rule based systems, which were deemed in the past to be unable to detect even slight variations of attacks due to rigid rules (Esmaili *et al.* 1996, Lewis 1993, Owens and Levary 2006). Rule based systems are now also capable of detecting variations of attacks, and may even be employed for anomaly detection, largely due to researchers incorporating fuzzy logic to define the rules (see Section 3.2 on page 19 for more details).

One of the benefits of anomaly detection is the ability to detect new attacks, since the system is modelled according to normal behaviour. The term 'behaviour' implies a host based IDS that analyses user behaviour, but it can also be an IDS analysing network traffic. In either case, modelling normal behaviour/traffic is an intensive task, which makes this approach prone to issuing *false positives* (Dokas *et al.* 2002, Kruegel *et al.* 2004).

Host based anomaly detection may be focused on user behaviour or process/program behaviour. With respect to the former, this relies on the belief that a user uses the system in a predictable way; that there is a pattern in the behaviour that is determined by habit. Therefore, it is assumed that it is possible to identify a user based on data, and if the system is used in a way that deviates from the habits of the respective user, then it is considered a potential intrusion (Debar *et al.* 1992, Ryan *et al.* 1998).

Debar *et al.* (1992) consider several levels of data sources to model anomaly detection on, which they classify as follows:

Keyboard level: which key that is hit, time since the last hit, *etc*.

Command level: which commands are used and the sequences of them. Researchers now also consider output parameters and arguments of system calls (Micarelli and Sansonetti 2007).

Session level: monitoring end-of-session events, which can produce data such as *"length of session, overall CPU, memory and input-output usage, name of terminal used, time of login, day of week..."* (Debar *et al.* 1992). However, as Debar *et al.* state, this approach is not likely to be able to perform real-time intrusion detection since the data is obtained only after the user has completed a session, by which time, the user may have completed the intrusion.

Group level: aggregating users into groups.

Based on any of these levels, an anomaly detection system may build up several profiles of users (hence, also known as *user profiling*). This can be implemented as either considering one profile per user, or as in the latter level, groups of users that may have particular rights in the system, *e.g.*, administrators, programmers, secretaries, *etc*. For further details, refer to Mutz *et al.* (2006).

A challenge of host based anomaly detection systems is keeping up to date with environmental changes. Retraining or continuous updating is required to avoid an increase in false alarms, referred to as behavioural drift (Balajinath and Raghavan 2001). It is possible to model/train an anomaly system over time, however, there is one particular issue with this: there is a danger of learning intrusive behaviour as well (Kruegel *et al.* 2004). If a user is aware that training of an anomaly detection system is commencing, s/he may gradually change her/his behaviour in such a way that a planned attack will not be detected (Gollmann 2006, p. 253).

2.3.3 Detection approaches

Two main approaches to detecting intrusions are considered here: *stateful* and *stateless*. Stateful approaches consider an attack as being composed of several events (stages), whilst stateless approaches attempt to classify single events as being an intrusion or not.

Event correlation is considered synonymous with stateful approaches here for simplicity, which has been subject to extensive research and is commonly adopted in commercial IDSs, *e.g.*, HP OpenView (Sheers 1996), Snort (Sourcefire Inc 1999), EMERALD eXpert and eXpert-BSM (Lindqvist and Porras 1999; 2001), and a well known open source IDS known as Snort (Sourcefire Inc 1999). Event correlation and stateless intrusion detection are discussed further in their respective sections below, followed by a discussion of the pros and cons of the two approaches.

2.3.3.1 Event correlation (stateful)

Event correlation, in a broad sense, refers to processing sequences of (typically low level) events, subject, possibly, to temporal constraints, in order to identify significant patterns that can be aggregated into a single higher level event. The purpose is typically to reduce the number of events and/or raise their semantic level, *i.e.*, aggregated events have more meaning than individual low level events. In the context of intrusion detection, event correlation is the process of identifying events as forming part of an attack pattern, in which the produced higher level event states (or suggests) which attack has been identified.

Event correlation systems may analyse data both spatially and temporally, building deterministic and/or probabilistic models of intrusions (Jiang and Cybenko 2004). Spatial systems analyse events from different sources simultaneously, whilst temporal systems consider not only the order of events to be significant, but also the time between them. For example, event B must occur within 150 milliseconds after event A has occurred to qualify as intrusion X.

Rule based systems are commonly used for event correlation (Jiang and Cybenko 2004, Lindqvist and Porras 1999), which is reviewed in more detail in Chapter 3. This is referred to as a *signature based* approach, since the system will filter events according to a set of rules (signatures) that determine the pattern of intrusions. Another approach is model based systems, such as Bayesian networks which are also reviewed in Chapter 3. There are several non-AI techniques that have been adopted in the literature to perform event correlation, such as the Codebook approach (Yemini *et al.* 1996) and finite-state machines (FSM) and other state based approaches (Ilgun *et al.* 1995). In related domains, *i.e.,* network management and fault management/localisation, other model based approaches have been adopted, such as Petri nets (Fabre *et al.* 2004), dependency graphs (Kätker and Geihs 1997, Katzela and Schwartz 1995), and hyper-bipartite networks (Kumar and Venkataram 1995).

2.3.3.2 Stateless intrusion detection

Although Jiang and Cybenko (2004) argue that rule based systems are stateless, there is a significant difference between this and the stateless approaches discussed here. In short, a stateless IDS attempts to classify single events (*e.g.*, network connections) as being intrusive or normal. Thus, not taking account of how this event may relate to any previous events as event correlation / stateful approaches would.

Stateless intrusion detection is popularly adopted in the data mining / machine learning communities, treating the intrusion detection problem as a classification task. The raw data, such as *tcpdump* for network based IDSs, needs to be transformed into suitable feature vectors such as those in MADAM/ID (Lee and Stolfo 2000). The feature vectors may also include some *a priori* knowledge, such as the *count* feature in the KDD Cup'99 data set (The UCI KDD Archive 1999), which contains information about the number of connections from a particular user within the last two seconds.

A challenge is to obtain a feature set that is comprehensive enough to separate normal data from intrusive data, but also keep the size of this set as small as possible. Typically, the more features, the more difficult the problem is to solve. For many machine learning algorithms, increasing the number of features (the dimension of the problem), significantly increases the training time required to learn the intrusion task (also the run-time will slow down and memory requirements increase with more features), commonly referred to as 'the curse of dimensionality' (Bellman 1961, Duda 2001). Hence, much research has been devoted to developing efficient techniques to perform *feature selection*, as further discussed in Section 2.5.

2.3.3.3 Discussion

The greatest drawback of stateless intrusion detection is that multi-stage attacks, such as the Mitnick attack described in Section 2.2, cannot be detected (Kruegel *et al.* 2004, p. 22). Stateful/event correlation systems can detect multi-stage attacks, provided that the attack is known already and is sufficiently described. Despite this drawback of stateless intrusion detection, a significant benefit is that such systems are quicker and require less memory (Kruegel *et al.* 2004, p. 22), and, thus, are more likely to succeed in offering real-time intrusion detection. Furthermore, the event correlation can be complicated by the existence of many possible attack scenarios, and, thus, it is difficult to determine which, if any, attacks are in progress (Benferhat *et al.* 2003). Similarly for state transition approaches, the computational requirements are high as it is necessary to keep track of many events concurrently.

A benefit of many stateless approaches, if implemented with machine learning, is that intrusions are automatically learned from a data set (training), instead of conducting knowledge engineering to produce a rule base for event correlation. Moreover, many techniques, such as Artificial Neural Networks also offer some flexibility so that variations of an attack may be detected. Conventional rule based approaches need a rule for every single intrusion and variation thereof. Not only does this require a large knowledge base, but variations of known attacks may go by undetected (Kruegel *et al.* 2004, p. 22). However, as Kruegel (2004,

p. 23) notes, such systems are able to give accurate information when detecting an intrusion, contrary to many machine learning techniques, which are, for practical purposes, black boxes. Another benefit of state based systems is that they can execute responses for potential intrusions before they are completed.

A challenge to either approach is updating the system. For rule based systems, this involves adding new rules and potentially updating old rules. The drawback of rule based systems is that the knowledge base of rules may grow very large with time and does not scale well (Jiang and Cybenko 2004). For some machine learning techniques, updating may involve complete re-training, which may involve a process of having to gather data concerning new intrusions. Some techniques are able to learn continuously online. However, the danger is that intrusive behaviour is also learned, as discussed previously in Section 2.3.2.

Event correlation is effectively employed to perform misuse detection, whilst stateless approaches offer a broader opportunity to perform misuse and anomaly detection. However, due to the focus on classifying single events in stateless IDSs, such systems are prone to producing alert storms (Kruegel *et al.* 2004, p. 22). For example, with *ping of death* (*pod*), as described in Section 2.2, an attacker may send many attack packets to a host (or many hosts), all of which would produce an individual alert. State based systems are less prone to doing so as the focus may be on detecting the attacker. However, as Kruegel *et al.* (2004, p. 28) observe, there is a trend to correlate the alerts from different IDSs to give higher level intrusion alerts or aggregate alert storms into a single intrusion event.

It is clear that either approach exhibits certain pros and cons, and that neither approach can be said to be 'better' than the other. As with network based and host based intrusion detection, state based and stateless implementations compliment each other. Currently, no research has demonstrated that it is possible to combine the two by learning event correlation for intrusion detection. Evolutionary programming (Fogel 1964; 1999, Fogel *et al.* 1966) attempts to achieve this, but has been mainly applied to grammatical inference, see, for example, Zomorodian (1996) and Lankhorst (1995a, 1995b). However, Vincent *et al.* (2007) propose a notation for evolving trees with genetic programming, which can be mapped to finite state machines, where each tree represents a (possibly complex) correlation pattern. As yet, there are no empirical findings that demonstrate this working in practice.

2.3.4 Behaviour on detection and usage frequency

IDSs commonly report suspected intrusions to a system administrator. Some systems may, however, take action against the suspected intruder (Kemmerer and Vigna 2002). For a state based system, which is able to notify about (potential) attacks in early stages, this may give an opportunity to stop the attack before substantial damage has been caused. However, as Kemmerer and Vigna (2002) stress, this can be dangerous since a counter action may mistakenly be directed at an innocent user. Alternatively, a defensive mechanism may be implemented by making the resources under threat temporarily unavailable to the potential attacker. For more information, refer to Kabiri and Ghorbani (2005) for a survey on report mechanisms.

Usage frequency refers to when the IDS is active, *i.e.*, performing real-time intrusion detection or offline, analysing batches of historical data. Researchers strive to achieve the former, which is a challenge in large networks. The consequence of the latter is that it is not possible to detect and prevent intrusions that are in progress. Furthermore, such IDSs may fail to recognise attacks that did take place if they were masqueraded well.

2.4 Intrusion detection in wireless and mobile *ad hoc* networks

IDSs in wireless and mobile *ad hoc* and sensor networks can largely be characterised by the taxonomy discussed in previous sections. However, compared with wired networks, there are additional challenges that need to be taking into account due to their distributed nature and *ad hoc* infrastructure. Brutch and Ko (2003) discuss several challenges and necessary considerations for intrusion detection in *ad hoc* networks:

- The intrusion detection cannot rely on a centralised module to analyse the network data.
- Unlike in wired networks, where the traffic can be monitored at routers, switches and firewalls, there is a lack of key concentration points in *ad hoc* networks where detectors can be deployed.
- Mobile hosts (nodes) can be compromised by an intruder and rejoin the network as a Byzantine node. Han *et al.* (2007) offer the following definition of a Byzantine node: *"In a decentralized network system, an authenticated node is referred to as a Byzantine node, if it is fully controlled by a traitor or an adversary, and can perform destructive behavior to disrupt the system"*.
- Overhead considerations must be taken into account as there are battery, CPU and memory constraints on the nodes. This has an impact on the architecture chosen to perform intrusion detection, which is discussed further below.
- There are constraints on the wireless links, regarding transmission range and bandwidth.

Most of the points above raise a need for different IDS architectures than those employed in wired networks. Zhang and Lee (2000) conclude that the IDS should be distributed and cooperative, and that detectors should be deployed on each node. In practice, the constraints on battery life, CPU and memory power of the nodes may not allow for such a solution. Brutch and Ko (2003) refine this suggestion to only employ detectors on nodes that are capable of facilitating detectors. They also discuss a *hierarchical* architecture that takes into consideration such constraints, in which nodes in the networks can become heads of clusters of nodes. Such cluster heads may then perform intrusion detection for all nodes in the cluster. Ahmed *et al.* (2006) and Samad *et al.* (2005) employ such a model, in which the cluster heads cooperate in performing intrusion detection, largely influenced by the battery and power constraints of the nodes. Furthermore, the cluster heads can include detection procedures to locate and determine Byzantine nodes (Madhavi and Kim 2008).

Other architectures include *stand-alone* and *cooperative* systems. In the former, each node runs an independent IDS. These can only detect intrusions based on the information present at the node, which has the same general drawbacks of host-based intrusion detection. In a cooperative system, each node makes local decisions, but also cooperate globally. However, such an architecture can be at a higher risk of intrusion caused by Byzantine nodes. Recent research focuses on trust schemes to limit this risk (Madhavi and Kim 2008).

Anomaly detection is preferred for intrusion detection in *ad hoc* networks, since regularly updating nodes with new signatures induces more overhead (Brutch and Ko 2003, Zhang and Lee 2000). An additional challenge is that the IDS is required to detect attacks directed at the *ad hoc* routing infrastructure as well as the mobile nodes. Moreover, there is a greater risk compared to wired networks, that the intruder does not need to obtain physical access to execute an attack. Brutch and Ko (2003) discusses three types of intrusion in such systems:

Passive attacks: intercepting transmissions (eavesdropping).

Active attacks against nodes: manipulating transmissions, by, for example, deleting, modifying or inserting messages to a node.

Active attacks against wireless links: denial of service by jamming or draining the battery of a node. The latter is referred to as a *sleep deprivation* attack, in which a node is flooded with malicious packets (Ahmed *et al.* 2006).

Refer to Ahmed *et al.* (2006) and Djenouri *et al.* (2005) for a description of several specific attacks.

This section has addressed the main challenges and consideration necessary for intrusion detection in wireless/mobile *ad hoc* networks. For more information, refer to Brutch and Ko (2003), Djenouri *et al.* (2005), Zhang and Lee (2000) for general treatments of the domain, and (Ahmed *et al.* 2006, Elhdhili *et al.* 2008, Huang and Lee 2003, Liu *et al.* 2005, Madhavi and Kim 2008, Wang *et al.* 2009) for examples of applications.

2.5 Areas of research

There are several research niches in the domain of intrusion detection, all of which help move the field towards a set of ideal requirements for IDSs, as described by Debar *et al.* (1999):

Accuracy: no false positives.

Completeness: no false negatives.

Performance: real time detection.

Fault tolerance: the IDS not becoming a security vulnerability itself.

Timeliness: handle large amounts of data. Concerned with how quickly the IDS can propagate the information through the network to react to potential intrusions. Also referred to as *scalability*.

A large proportion of research on intrusion detection focuses on developing new system architectures and detectors to improve on the *accuracy and completeness* of the IDS. Event correlation is a very substantial research area, which has been established well for misuse detection. Related to system architectures, an emerging research area is intrusion detection in wireless/mobile *ad hoc* and sensor networks, as discussed above in Section 2.4.

As reviewed in Chapter 3, there is a trend in applying machine learning to intrusion detection, which offers flexible detectors and lends itself conveniently to anomaly detection. Moreover, it is now common to develop hybrid systems, which may combine misuse and anomaly detectors, host based and network based modules, and event correlation and stateless detectors. Consequently, concurrent with increasing research on hybrid IDSs, much recent research focuses on correlating effectively alerts between the different modules (Kruegel *et al.* 2004, pp. 28–33).

Related to alert correlation, alert aggregation is also the focus of recent research, which attempts to group similar alerts/events into a single generalized event. This can significantly reduce the false positive rates and the amount of alerts a system administrator is required to investigate (Al-Mamory and Zhang 2009). See Sadoddin and Ghorbani (2006) for a survey on alert correlation, and Al-Mamory and Zhang (2009), Corona *et al.* (2009), Maggi *et al.* (2009), Morin *et al.* (2009) and Zhou *et al.* (2007) for a selection of recent research papers on alert correlation and alert aggregating.

Much research addresses scalability by distributing/decentralising the IDS. This can be done with event correlation (Burroughs *et al.* 2002, Kruegel *et al.* 2001, Kruegel and Toth 2002) as well as with other detectors based on, for example, mobile agents and artificial immune systems. The latter are discussed in greater depth in Sections 3.8 and 3.9. Whilst event correlation is largely a passive process, researchers have proposed active probing as an alternative approach (Rish *et al.* 2005, Tang *et al.* 2005). Probes are issued to gather information about the performance of the distributed system. In active probing, Rish *et al.* (2005) utilise Bayesian reasoning to determine how many probes should be issued and the tests they should perform. The motivation for such a scheme is to reduce the computational costs, in order to achieve real-time diagnosis of the system.

Although there is value in discovering intrusions in hindsight, researchers strive to obtain real-time intrusion detection. Despite the drawbacks of stateless approaches, as discussed in Section 2.3.3.3, they can more readily achieve real-time intrusion detection. In conjunction with other state based intrusion detection modules, which is becoming more common, the stateless detectors may be great assets by detecting some share of intrusions in real time.

Related to achieving real-time intrusion detection, researchers have investigated several methods of performing *feature selection*. The primary benefit of feature selection is that the amount of data required to process is reduced, ideally without compromising the performance of the detector. In some cases, feature selection may improve the performance of the detector as it simplifies the complexity problem by reducing its dimensionality. See Chen *et al*. (2006) for a survey and taxonomy of feature selection algorithms, and Chebrolu *et al.* (2005), Fries (2008), Mukkamala and Sung (2002), Makkithaya *et al.* (2008), Sivagaminathan and Ramakrishnan (2007), Sung and Mukkamala (2003), Tsang and Kwong (2005a;b) and Tsang *et al.* (2007) for a selection of studies adopting AI-based feature selection techniques for intrusion detection.

A paradox with intrusion detection is that the IDS itself may become a security vulnerability. Kruegel *et al*. (2004, pp. 22–23) argue that state based IDSs are more prone to attacks than stateless approaches, as they can be flooded with events that prevent them from functioning efficiently enough. However, security is compromised for stateless machine learning approaches as well, which is subject to recent research referred to as *adversarial learning/classification*. It is applications of machine learning algorithms that learn over time that are particularly vulnerable to adversarial attacks. As previously discussed in Section 2.3.2, there is a danger that an adversary may manipulate the training process, by gradually changing his/her behaviour over time so that a planned attack will not be detected (Gollmann 2006, p. 253). Barreno *et al.* (2006) give an overview of threats to learning algorithms, and discusses ways to protect against, and detect, an adversary. See Biggio *et al.* (2008; 2009), Dalvi *et al.* (2004), Lowd and Meek (2005) and proceedings of the NIPS workshop on adversarial learning (Laskov and Lippmann 2007) for more information on this topic.

Other research topics include detecting masquerades (Kim and Cha 2004, Schonlau *et al.* 2001, Seo and Cha 2007, Yung 2004), and developing correlation languages for event correlation (Albaghdadi *et al.* 2001, Cuppens and Ortalo 2000, Eckmann *et al.* 2002, Michel and Mé 2001, Sánches *et al.* 2003, Vaarandi 2002, Vincent *et al.* 2007, Zhu and Sethi 2001).

CHAPTER 3

Artificial intelligence in intrusion detection systems

This chapter offers a broad review of applications of Artificial Intelligence (AI) to intrusion detection. A general awareness of AI techniques is assumed. Section 3.1 provides references for texts on the techniques discussed, and considers briefly other methods applied to intrusion detection. The most widely applied AI techniques are then discussed, with consideration of hybridisation. Section 3.13 concludes with a summary of the field.

3.1 Background reading and related research

For further information on the AI techniques discussed in this chapter, readers are referred to the following texts: Russell and Norvig (1995) and Luger (2002) for a general treatment of AI; Mitchell (1997) for a general treatment of Machine Learning; Giarratano and Riley (1998) and Nikolopoulos (1997) on Expert Systems; Riesbeck and Schank (1989) on Case Based Reasoning; Haykin (1998) on Artificial Neural Networks; Burges (1998) and Cortes and Vapnik (1995) on Support Vector Machines; Dasgupta (1998) and de Castro and Timmis (2002) on Artificial Immune Systems; Quinlan (1993) on Decision Trees; Korb and Nicholson (2003) on Bayesian networks; Rabiner (1989) on Hidden Markov Models; Mirkin (1996), Everitt (1993), Kaufman and Rousseeuw (1990) and Jajuga *et al.* (2002) on clustering and classification; Cockayne and Zyda (1998) and Pierre and Glitho (2001) on Mobile Agents; Goldberg (1989) and Mitchell (1998) on Genetic Algorithms; Banks *et al.* (2008a, 2008b) on Particle Swarm Optimisation; and Dorigo and Stutzle (2004) on Ant Colony Optimisation.

Due to the scope of this review, there are techniques and approaches that are not considered here. However, that does not imply that they are not used in modern IDSs. Statistical techniques have been widely used (Biermann *et al.* 2001), particularly for anomaly detection, and still form parts of many hybrid IDSs. Applications of Bayesian networks are included here, but other model based approaches, and state based applications such as STATL and USTAT (Ilgun 1993), are not. Pattern matching, particularly string matching, has also been successfully applied to this domain, with proposed algorithms such as ExB (Markatos *et al.* 2002), E2xB (Anagnostakis *et al.* 2003), and Piranha (Antonatos *et al.* 2005).

The following texts are suggested as complementary reading: Kabiri and Ghorbani (2005) for a survey of intrusion detection and response, Sadoddin and Ghorbani (2006) for a survey of alert correlation, Zhou *et al.* (2010) for a survey of coordinated attacks and collaborative intrusion detection, Patcha and Park

(2007) for a survey of anomaly detection techniques, Gagnon and Esfandiari (2007) and Wu and Yen (2009) on AI applied to intrusion detection, in which the former focuses on knowledge representation and the latter on computational intelligence.

3.2 Rule based systems

One of the most common forms of Rule Based Systems (RBSs) that have been applied to intrusion detection are expert systems (Cannady 1998). The strength of this technique is in performing event correlation for misuse detection, as discussed in Section 2.3.3.1 on page 12. Existing event correlation tools are discussed in Section 3.2.1.

In the last decade, fuzzy logic and rule induction has increasingly been applied to intrusion detection, which is discussed in Section 3.2.2. Although most RBSs do not facilitate effective anomaly detection, there are examples of this in the literature, as considered in Section 3.2.3. Finally, hybrid systems incorporating RBSs are discussed in Section 3.2.4.

3.2.1 Event correlation tools

There is a range of event correlation tools created with rule based systems, all of which operate similarly. The different tools have been somewhat specialised for different environments, allowing different types of rules.

One tool that has been around for approximately two decades is the Production-Based Expert System Toolset (P-BEST), which has been integrated into several IDSs with a focus on handling SYN flooding and buffer overruns (Lindqvist and Porras 1999). Lindqvist and Porras (1999) briefly describe four IDSs that P-BEST have been used in: MIDAS, IDES, NIDES and EMERALD eXpert, and in (Lindqvist and Porras 2001), a fifth, eXpert-BSM; all systems being suitable for real-time misuse detection. The first three systems are host based, whilst the latter two have achieved support for distributed networks. Although eXpert-BSM was developed to analyse Sun Solaris audit trials on a host, it can be distributed by employing an alert collection application referred to as an *eftunnel*, which will produce a single event stream. Lindqvist and Porras (1999) highlight some drawbacks of P-BEST, such as being poor at dealing with uncertainty and missing data due to being strictly forward chaining.

Motivated by existing event correlation tools being platform dependent, complex to operate and specialised for specific event correlation tasks, Vaarandi (2002) presents a platform independent, open source, tool for rule based event correlation called the Simple Event Correlator (SEC). This tool is designed to be lightweight so that it is better able to deal with the complexity, size and cost issues typical of event correlators. Furthermore, SEC is intended to be usable in many domains, with existing applications such as: network fault management, intrusion detection, log file monitoring and fraud detection.

3.2.2 Fuzzy logic and rule induction

Tillapart *et al.* (2002) propose a fuzzy rule based system for network based intrusion detection. Fuzzy logic (Zadeh 1965) is one approach to obtaining more flexible rules compared with traditional expert systems (which operate with crisp threshold boundaries to classify network traffic or user behaviour). Fuzzy rules allow the IDS to quantify some degree of belonging to either intrusion or normal. Hence, this information may be incorporated in the alert reports, instead of only reporting intrusion alerts for traffic/behaviour that exceed a crisp threshold. Bridges and Vaughn (2000) contend that the very nature of intrusion detection is fuzzy; not always being clear whether events are a misuse or not. Furthermore, they also promote fuzzy logic because events may have membership in several categories, instead of just one or none.

Owens and Levary (2006) utilize fuzzy set theory to develop an adaptive expert system for network based intrusion detection. Fuzzy rules can be created to perform both misuse and anomaly detection. The system architecture includes a control mechanism, which may be independent of the system outputs (open loop) or dependent on it (closed loop). Owens and Levary have implemented a closed loop system, in which a system administrator is involved in the decision process. Events are mapped to fuzzy sets, which are then classified by an expert system that determines an alert with a suspicion level of either ‘low’, ‘medium’ or ‘high’. The system administrator is able to respond to the alerts and modify the reporting sensitivity.

Rule based systems such as expert systems typically include a comprehensive task of knowledge engineering to formulate the knowledge experts have of known intrusions into rules. As an alternative approach, there are rule induction techniques that can mine data sets to determine such rules automatically. Rules can be mined to represent both intrusive and normal behaviour, thus, allowing for rule based anomaly detection (discussed further in the following section). RIPPER (Cohen 1995) is a popular rule mining algorithm that can be used to create a classifier, which is considered in several studies discussed in this review, *e.g.*, (He *et al.* 2007, Lee and Stolfo 2000, Warrender *et al.* 1999).

Bridges and Vaughn (2000) examined a combination of a RBS and fuzzy association rule mining to monitor network traffic and system audit trails. The RBS facilitates misuse detection, and the latter anomaly detection. They find that fuzzy logic can help to extract more general patterns of intrusions. In their experiments, incorporating fuzzy logic into the rule mining reduced the number of false positives.

Florez *et al.* (2002) extended the research of Bridges and Vaughn (2000), making various improvements. In particular, they identified an issue with the previous approach in dealing with multiple occurrences of an item set (of events). The more occurrences of an item set, the less confidence the potential intrusion scenario would obtain. This contradicted their observations, where larger item sets gave better information. Consequently, they penalised item sets with few items, and improved the accuracy of the system. They also employed a faster data mining algorithm, adapting *prefix* trees to mine fuzzy association rules, which improved the general performance of the system.

In both studies above, a Genetic Algorithm (GA) has been used for feature selection and to optimise the fuzzy membership functions. GAs have also been applied to rule learning in other studies, *e.g.*, Jeya and Ramar (2007), Selvakani and Rajesh (2007), Shafi *et al.* (2009), Banković *et al.* (2007) and Orfila *et al.* (2009). For more information, see Section 3.12.2 on page 49.

3.2.3 Anomaly detection

Most RBSs do not facilitate effective anomaly detection. Commonly, researchers seek to compliment a RBS (for misuse detection) with other techniques for anomaly detection. However, there are examples of fuzzy rule based approaches for anomaly detection in the literature.

Dickerson *et al*. (2000, 2001) propose using a RBS for network based anomaly detection by creating a set of general fuzzy rules to deal with common intrusion scenarios. Their IDS architecture consists of three modules: first, network data is collected by a sniffer module, which is then organised and preprocessed by a second module to give fuzzy inputs to a fuzzy detector module. The second module includes data mining and feature selection to process the data before being input to the fuzzy detector. This was found to be very successful as the amount of input data is reduced and anomalies could be more easily detected. Dickerson *et al*. (2001) extend the approach to operate as a multi agent system, which is discussed in Section 3.9.

Tajbakhsh *et al*. (2009) propose an IDS based on fuzzy association rule mining, which is similar to that of Bridges and Vaughn (2000) and Florez *et al.* (2002). The main difference is that the approach of Tajbakhsh *et al*. allows for anomaly detection, in which fuzzy association rules are used to build stateless classifiers. They compare the performance of their IDS, used for misuse detection and anomaly detection on the KDD Cup ’99 data set (The UCI KDD Archive 1999). For anomaly detection, the IDS obtains

80.6% TPR and 2.95% FPR, compared with 91% TPR and 3.34% FPR for misuse detection. As a misuse detection system, the approach is not as successful as other machine learning techniques. However, as an anomaly detection module the approach is more successful. In comparison, an artificial neural network for anomaly detection, employed by Ghosh and Schwartzbard (1999), obtained best results of 77.3% TPR and 3.6% FPR.

3.2.4 Hybrid systems

Hybridisations that include RBSs appear in different forms in IDSs, which may be described as follows:

Algorithmic: techniques/algorithms are hybridised to perform a single task together. For example, using genetic algorithms to evolve rules for a RBS (Jeya and Ramar 2007, Selvakani and Rajesh 2007).

Cooperative: different techniques are employed to perform different, independent, tasks, which then are combined in a some manner to form a holistic system. For example, one technique for misuse detection and another for anomaly detection (Aydin *et al.* 2009, Goss *et al.* 2007, Yuan and Guanzhong 2007).

Hierarchical: there is a hierarchy in the architecture of the IDS, which includes different techniques performing different tasks at each level. For example, a RBS may be utilised at the top-level, correlating alerts from several detectors at a lower level (Depren *et al.* 2005).

Many systems include some combination of all three categories, and several examples of the two first categories have been discussed previously. For example, the IDS proposed by Bridges and Vaughn (2000) and Florez *et al.* (2002) incorporates hybridisation at two levels. The IDS comprises a RBS for misuse detection and fuzzy association rule mining for anomaly detection. Additionally, the anomaly detection is made possible by a hybridisation of association rule mining and a GA to optimise fuzzy membership functions. Furthermore, a GA is used to perform feature selection.

Another example of a cooperative hybrid system is a host based IDS developed by Yuan and Guanzhong (2007), which performs misuse detection and anomaly detection. An expert system is employed to perform misuse detection and anomaly detection is achieved by building statistical profiles of users described by rules of normal behaviour. A natural necessity of such hybrid systems is a ‘decision’ module, which can filter the alerts from the detectors and determine whether an alert should be raised to the user. Depren *et al.* (2005) propose a hybrid system comprising a Self Organising Map (SOM) for anomaly detection, a Decision Tree (DT) for misuse detection, and a RBS as a decision module. The RBS consists of three simple rules that are used to determine whether an intrusion alert should be raised and by which module (the SOM or the DT). For example, the first rule is:

> *“If anomaly module detects an attack and misuse module detects an attack then it is an attack and misuse module classifies this attack”* (Depren *et al.* 2005)

Hybrid systems are becoming more common, and consequently, so is alert correlation (Kruegel *et al.* 2004, pp. 28–33). Alert correlation may involve correlating alerts from different IDSs to give a higher level intrusion alert, or aggregate alert storms into a single intrusion alert. Zhou *et al.* (2007) have developed a comprehensive IDS prototype, which is an example of a system that is hybridized across all levels mentioned above. The IDS consists of three main components: a capability model (abstracted intrusions), inference rules and alert correlation algorithms.

Important to the IDS of Zhou *et al.* (2007) is the utilisation of a concept they refer to as ‘capability’, which is used to make abstract models of intrusions, allowing the system to correlate alerts belonging to the same intrusion. They focus on detecting multi-stage intrusions, adopting the ‘requires/provides’ model of

Templeton and Levitt (2000). Similarly to the intrusion model discussed in Section 2.2 on page 8, this takes into account that an attack may involve several phases, *e.g.*, first probing/scanning to identify vulnerable hosts and then exploiting those. Hence, the focus is more on detecting the attacker, rather than flagging an alert for every malicious event that is a part of an ongoing attack. Moreover, from a practical point of view, the abstraction groups events that are logically similar (have the same effect) but may have dissimilar event details. This reduces the amount of rules and details that the IDS needs to process.

From the capability model, Zhou *et al.* (2007) derive inference rules to describe the logical relationships between the different capabilities. One rule defines the relationship between two capabilities. Since the inference rules are defined independent of the capability model, different inference strategies can be employed without changing the capability model. The final part of the IDS is the alert correlation. Abstract alerts are correlated based on the capabilities and relationships determined in the two previous modules.

3.3 Instance based learning

Many researchers employ Instance Based Learning (IBL) techniques in intrusion detection and event correlation/fault management as a means of obtaining a more flexible system compared with most expert systems, particularly for dynamic networks. The general drawbacks of expert systems highlighted in the literature (Esmaili *et al.* 1996, Gürer *et al.* 1996, Jakobson *et al.* 2004, Lewis 1993, Owens and Levary 2006) include*:*

- Knowledge engineering is time consuming and difficult (extracting expert knowledge of intrusions and coding this in rules).
- Difficult to manage and update the rule base.
- Many specific rules, which generally (unless fuzzy) cannot detect slight variations of know attacks.

IBL operates by solving problems based on previously solved instances/cases, and, thus, unlike expert systems, does not require knowledge engineering to determine specific rules. Furthermore, the knowledge base of instances/cases can be updated automatically and the system may learn from its own experience during operation. However, IBL is not as efficient as expert systems in performing event correlation (Hanemann 2006), and has high memory requirements as it is necessary to store a large number of cases (Lane and Brodley 1999).

Esmali *et al.* (1996) were the first to consider IBL for intrusion detection (misuse detection). They propose using Case Based Reasoning (CBR) to alleviate some of the issues other IDSs at the time faced in acquiring and representing knowledge. The proposed system is designed to analyse command records from audit trails of the C2 BSM (Basic Security Module) on Sun computers. An ‘Action Class’ module is employed to group (abstract) commands that have a similar action, which helps alleviate issues with slight variations of attacks going by undetected.

Lane and Brodley (1999) have developed an IDS to perform anomaly detection by means of IBL. Their system builds up user profiles based on UNIX commands, which are used to catch long term, ‘unconventional’, misuse, such as data theft. Other types of ‘instant’ misuse are not handled as they consider these well covered by existing systems. Their research has a focus on data reduction techniques, addressing the general issue of high memory requirements of IBL. They examine two data reduction models: one based on instance selection and the other on clustering. Their results indicate that instance selection yields higher detection rates and and takes a shorter time to generate false alarms, as well as detecting anomalies. However, instance selection was not able to maintain the characteristics of the users, whilst clustering did. Hence, the authors consider clustering as the better alternative. A few years later, Lane and Brodley (2003) compared IBL with Hidden Markov Models, which is considered in Section 3.11 on page 45.

Similarly to Lane and Brodley, Micarelli and Sansonetti (2007) propose an approach to host based anomaly detection with CBR. Their system analyses audit trails from a C2 BSM of a Solaris computer, taking into account output parameters and arguments of system calls. However, the behaviour of users and the network configuration is represented graphically, which may be analysed by image processing algorithms. Initially, a case base is populated with images of normal interaction with different applications. During runtime of the system, input data is first subject to feature extraction and clustering to compare system call sequences. This gives a feature distribution that is represented graphically, which is compared to the case base using Earth Mover's Distance (Rubner *et al.* 2000). Micarelli and Sansonetti (2007) evaluated their system on a small subset of the DARPA99 data set (Lippmann *et al.* 2000b), on four *U2R* attacks, *eject*, *fdformat*, *ffbconfig* and *ps*. The case base was populated with 117 instances of normal application behaviour. The system was capable of detecting all attacks with no false positives.

CBR has also been used for alert correlation (Long 2006, Long and Schwartz 2008). Similarly to other applications of IBL to intrusion detection, Long and Scwartz (2008) apply CBR to alert correlation to avoid issues other techniques have in acquiring and representing knowledge. A case base is populated automatically from a training set with examples of correlated alerts. The IDS based on CBR was demonstrated to be successful, being able to operate with limited training data and was able to detect multi-stage intrusions (as well as single attacks).

Long and Scwartz (2008) consider alert correlation as a matching problem, *i.e.*, how to match current input scenarios (alerts) with cases in the case base. They compare the performance of a commonly used maximal matching approach, with a new approach they propose; order preserving matching. The former iterates through the all cases in the case base and locates a subset of the run-time alerts that closely match the respective case. The latter takes into account the temporal relationship between the alerts, considering the order of the alerts as important when matching. They found order preserving matching to be more efficient.

3.4 Bayesian reasoning

Bayesian reasoning is considered here a general phrase for a range of techniques that exploit Bayes theorem to deal with uncertainty. In short, Mitchell (1997, p. 154) provides the following definition:

> *"Bayesian reasoning provides a probabilistic approach to inference. It is based on the assumption that the quantities of interest are governed by probability distributions and that optimal decisions can be made by reasoning about these probabilities together with observed data."*

There is a diverse range of implementations of Bayesian reasoning in IDSs. The two main Bayesian approaches are discussed below in Sections 3.4.1 and 3.4.2. Applications related to event correlation are considered in Section 3.4.3, and other Bayesian applications are discussed in Section 3.4.4.

3.4.1 Bayesian networks

In recent years, Bayesian networks have been utilized in the decision process of hybrid systems (Kruegel *et al.* 2003, Mutz *et al.* 2006, Scott 2004). Bayesian networks offer a more sophisticated way of dealing with this compared with a RBS. Kruegel *et al.* (2003) argue that most hybrid systems obtain high false alarm rates due to simplistic approaches to combining the outputs of the techniques in the decision phase. They propose a hybrid host based anomaly detection system consisting of four detection techniques: analysing string length, character distribution, and structure, and identifying learned tokens, in which a Bayesian network is employed to decide the final output classification. The system was validated on the DARPA99 data set (Lippmann *et al.* 2000b), compared with a simple threshold based approach. Both approaches (Bayesian and threshold) were given the same outputs from the detection techniques. With 100% true positives, the threshold based approach lead to twice as many false positives as the Bayesian network.

Mutz *et al.* (2006) extended the work of Kruegel *et al.* (2003), proposing an application based IDS that also considers system call arguments when analysing user commands. Most IDSs exclude this information, which they argue is a reason for false negatives, as it is possible to execute intrusions with valid system calls. In their IDS, normal profiles are created for each application in the system that is being protected. The same detection techniques as in (Kruegel *et al.* 2003) are adopted, and they obtain improved detection rates. Mutz *et al.* also focus on CPU load, since the IDS should not take up too many resources because it might prevent the user from using the computer efficiently. In their experiments, the the CPU load remained relatively low. During stress tests, the increase in CPU load was within 20% on average.

3.4.2 Naïve Bayes

Naïve Bayes (NB) is a simplified version of Bayesian networks, which offer machine learning capabilities. According to Mitchell (1997, p. 155), there are two particular drawbacks of Bayesian networks, namely the requirement of *a priori* knowledge about the problem to determine probabilities, and that the method is computationally expensive. For the former, it is possible to extract probabilities from training data, if available (Korb and Nicholson 2003), which is achieved with NB. However, NB does assume that all the features in the data are independent of each other (Mitchell 1997, p. 177), which is the reason An *et al.* (2006) apply Bayesian networks to database intrusion detection instead of NB. Nevertheless, NB (utilized as a classifier) has been successfully applied to network based intrusion detection by several researchers.

Ben Amor *et al.* (2004) conducted an empirical investigation on the KDD Cup '99 data set, comparing the performance of NB and a Decision Tree (DT). The DT obtains a higher accuracy (92.28% compared with 91.47%), but NB obtains better detection rates on the three minor classes[1], namely *Probing*, *U2R* and *R2L* intrusions. Most significantly, the DT detects merely 0.52% *R2L* intrusions whilst NB detects 7.11%. Similar observations are made by Panda and Patra (2007), as they compare NB with an ANN. ANNs and DTs are biased towards the major class(es) (Chawla 2003, Jo and Japkowicz 2004), and, therefore, are prone to perform worse on the minor class(es). Therefore, this can be seen as a benefit of the NB, provided that the FPR does not become too high.

NB has also been found to be more robust than some other machine learning techniques. Gharibian and Ghorbani (2007) compare the performance of two probabilistic techniques, NB and a Gaussian classifier, and two predictive techniques, a DT and Random Forest (RF, an ensemble of DTs). They analyse the performance of the techniques on three different training sets of the 10% KDD Cup '99 data set (all tested on the original test set). Each training set consists of 90,000 instances, but with different proportions of normal and intrusive data. For each set, 10 randomly created versions were selected to examine the sensitivity of the techniques. In the best cases, NB and the Gaussian classifier performed significantly better on the minor classes, *U2R* and *R2L*, but NB performed worst on *DoS*. However, the DT and RF were very sensitive to the training data selected, and the mean performance was lower than the probabilistic classifiers.

Bosin *et al.* (2005) compare the performance of NB with an Adaptive Bayesian Network (ABN). They use a different subset of the KDD Cup '99 data set than Ben Amor *et al.* (2004), which led to significantly different results. Hence, direct comparisons are not made across these studies. However, the behaviour of the algorithms is similar, *i.e.*, NB obtains higher detection rates on the minor classes. The greatest difference is clear from the detection of *U2R* and *R2L* intrusions; the ABN detecting 0% *U2R* and merely 4.8% *R2L*,

[1] The minor classes is a reference to the classes with the fewest instances to learn from. The discussion here on classification of the major and minor classes strongly relates to Chapter 5, which focuses on learning from imbalanced data. For more information on learning from imbalanced data, refer to Section 5.1 on page 77.

whilst NB detects 52.4% *U2R* and 94% *R2L*. Due to the low proportion of instances of these two classes, the ABN does obtain the highest accuracy by correctly classifying more *Normal* and *DoS* instances, which are the major classes.

The findings of Liu *et al.* (2008a) suggests that NB with Kernel Estimation (NBKE) is advantageous. They compare the performance of NB with and without kernel estimation on data gathered at Wuhan University in June/July 2008, with a focus on detecting flooding attacks and port scans. NBKE obtains 98.80% accuracy compared with 94.40% for the basic NB algorithm. Furthermore, the authors propose using an additional feature, a Hurst exponential, which is a measure of the traffic rate and port dispersion (how many ports were used in a specific time window). Experiments on detecting UDP flooding gave a 6% higher accuracy with Hurst.

Valdes and Skinner (2000) have integrated NB as a module to EMERALD (Porras and Neumann 1997, Neumann and Porras 1999), which they refer to as eBayes TCP. This module analyzes TCP session data provided by EMERALD, which offers both anomaly and misuse detection. eBayes TCP is hypothesis driven, initially consisting of 5 normal user modes and 7 intrusive modes. The system is unsupervised and can learn new modes. In this context, observations on real data led to an additional mode being added, HTTP_F, which consists of long HTTP sessions being terminated abnormally, but not being intrusive. Valdes and Skinner validated eBayes TCP on the TCP data in the DARPA99 data set, which gave promising results. The system detected all but one visible *portsweeps* from week 4, and all *portsweeps* from week 5. Further, it detected all *mailbomb* and *process table* intrusions, though it detected some *satan* intrusions as *portsweeps*.

Observations such as those made by Ben Amor *et al.* (2004) and Panda and Patra (2007), as discussed above, motivate potential hybridisations of techniques. For example, Benferhat and Taiba (2005), similarly to Ben Amor *et al.* (2004), also observe that NB is better at detecting some intrusions than a DT. They emphasise an issue with false negative rates, which lead them to prose a hybrid system of anomaly detection and misuse detection, allowing the anomaly detection module to deal with normal traffic. Thames *et al.* (2006) implemented a similar hybrid model, training a Self Organising Map (SOM) on normal instances, which functions as a first level intrusion detector. At the second level NB appends its classification to that of the SOM. This hybrid system led to better classification rates than using NB alone.

3.4.3 Event and alert correlation

Burroughs *et al.* (2002) apply a Bayesian Multiple Hypothesis Tracking (BMHT) algorithm to correlate events from distributed network based IDSs. This allows event correlation on a higher level by collecting data from multiple isolated sources in the network, which they argue leads to a more complete view of the network. Data is gathered from the isolated IDSs, which is then refined by five steps: (1) preliminary analysis and noise removal, (2) normalising the data and presenting it in a common format, (3) using the BMHT algorithm to aggregate similar events, (4) analysing and predicting future behaviour based on the information gathered and processed thus far, and (5) refining the knowledge base according to the findings. Burroughs *et al.* validated this system on a network with 5 subnets monitored by Snort and SHADOW, which lead to 89% true positives and 20% false positives.

As an alternative to event correlation, Rish *et al.* (2005) propose using dynamic Bayesian networks for active probing in distributed networks. According to Rish *et al.*, *"probes are end-to-end test transactions that collect information about the performance of a distributed system"*. They utilise this approach to obtain real-time intrusion detection. Periodically, a set of probes are sent out in the system to gather information, which also establishes a current belief about the system state. If a potential problem is detected, more probes are sent to gather more information, which will update the system belief.

3.4.4 Other Bayesian applications

Previous sections discussed the main applications of Bayesian reasoning, but it has been used in a diverse range of other applications. For example, Barbara *et al.* (2001) use Bayesian estimation for smoothing of values in a system referred to as ADAM, to help improve its ability to detect new intrusions. With this approach, they take into account the frequencies impact on the estimation values, *i.e.*, *"it may be misleading to report both 0/5 and 0/500 as being equal to zero"*. Chebrolu *et al.* (2005) used a Bayesian belief network for feature selection in a hybrid system with a Classification and Regression Tree (CART). The feature set extracted with the Bayesian belief network led to better performance than a feature set extracted by the CART.

Xiang *et al.* (2008) propose a hierarchical hybridisation of Bayesian clustering (AutoClass) (Cheeseman and Stutz 1996) and a DT. AutoClass is an unsupervised machine learning algorithm, which they use in the second stage of the detection process to separate normal data from *U2R* and *R2L* intrusions, since the DT was unsuitable for detecting these classes of intrusion. Liu *et al.* (2006) use a Bayesian game approach to perform intrusion detection in *ad hoc* mobile networks. They aim to avoid installing an IDS on every node in the network and improve the monitoring efficiency. By using Bayesian game theory, they analyse the intrusion detection scenario as a game with pairs of attacking and defending nodes.

3.5 Decision trees

Decision trees (DTs) are popular in misuse detection systems, as they yield good performance and offers some benefits over other machine learning techniques. For example, they learn quickly compared with Artificial Neural Networks (ANNs), and DTs are not black boxes. Furthermore, as Bouzida and Cuppens (2006b) highlight, the tree structure that is automatically built from the training data can be used to produce rules for expert systems.

DTs do, however, suffer from a number of drawbacks. One, similar to that of RBSs, is that they cannot generalise to new attacks in the same manner as certain other machine learning approaches. They are not suitable for anomaly detection (for which there are no applications reported in the literature). Empirical findings also demonstrate that DTs are very sensitive to the training data and do not learn well from imbalanced data (Chawla 2003, Gharibian and Ghorbani 2007).

Section 3.5.1 discusses applications of DTs as classifiers to perform misuse detection. Hybridisation and classifier combinations incorporating DTs are discussed in Section 3.5.2, followed by other DT applications in Section 3.5.3.

3.5.1 Classifier performance

DTs have been successfully applied to intrusion detection both as a stand alone misuse detector (Bouzida and Cuppens 2006a;b, Chebrolu *et al.* 2005, Sabhnani and Serpen 2003) or as a part of hybrid systems (Chebrolu *et al.* 2005, Depren *et al.* 2005, Pan *et al.* 2003, Peddabachigari *et al.* 2007, Sinclair *et al.* 1999). A good example of the success of DTs is an application of a C5.0 DT by Pfahringer (2000), which won the KDD Cup '99 competition (although with bagging and boosting).

Sabhnani and Serpen (2003) have examined the performance of several machine learning techniques on the KDD Cup '99 data set, including a C4.5 DT. The DT obtained good accuracy, but does not perform as well as other techniques on some classes of intrusion, particularly *U2R* and *R2L* attacks, both of which are minor classes and include a large proportion of new attack types. An ANN and *k*-means clustering obtained higher detection rates on these classes, which are two techniques that are better able to generalise from learned data to new, unseen, data. Similar observations have been made by Gharibian and Ghorbani

(2007), as discussed in Section 3.4.2. Furthermore, Gharibian and Ghorbani found that DTs and Random Forests (ensemble of DTs) are very sensitive to the data selected for training, *i.e.*, the performance varied significantly on different folds (subsets) of the data. The probabilistic techniques they examined (NB and Gaussian) were more robust and obtained higher detection rates on the minor classes.

As stated above, DTs do suffer from the drawback of not being able to deal well with unseen data. New attacks may be classified as some default class, such as 'normal', as for the C4.5 DT employed in an investigation by Bouzida and Cuppens (2006b). Consequently this causes false negatives. Therefore, Bouzida and Cuppens developed a modified C4.5 DT, which classifies new/unseen data as a new 'unknown' class. By doing this, they avoid a significant amount of misclassifications of new attacks as normal connections, particularly *U2R* attacks.

Ohta *et al.* (2008) also propose a modification to the C4.5 DT classifier, aimed at reducing the false positive rate. They change the way in which the trees are built, by taking into account the type of errors that may be produced, choosing attributes that are less likely to produce false positives. The modified DT is evaluated on subsets of the KDD Cup '99 data set, and is compared with the original C4.5 DT. As an alternative approach to reducing the false positives, they also examine random oversampling and undersampling of the training data. The modified C4.5 DT outperformed the original DT and the sampling approach. Over/undersampling led to similar detection rates, but obtained a much higher false negative rate.

3.5.2 Hybrids and classifier combination

Several researchers have benchmarked a range of machine learning algorithms, observing that different techniques appear better at detecting different classes of intrusion (Anuar *et al.* 2008, Gharibian and Ghorbani 2007, Pan *et al.* 2003, Peddabachigari *et al.* 2007). However, as discussed in Chapter 4 on page 54, some of these observations are contradictory due to differences in methods. Nonetheless, creating classifier ensembles of different techniques has been shown to outperform the individual classifiers (Pan *et al.* 2003, Peddabachigari *et al.* 2007).

Although some researchers experience the instability of DTs as a drawback, that their performance is sensitive to the training data (Gharibian and Ghorbani 2007), others exploit this as a beneficial trait to construct successful ensembles of DTs (classifier combination) (Breiman 1996). One commonly used ensemble approach, Random Forest (RF) (Breiman 2001), was first applied to intrusion detection by Zhang and Zulkernine (2006) to perform network based misuse and anomaly detection. Their system makes use of a hierarchical hybridisation, in which the misuse detection module operates at the first level, employing a RF to classify attacks in real time. Anomaly detection is then employed at a second level, utilizing RF to perform outlier detection on data that is not classified as intrusive by the misuse detection module. On a small subset of the KDD Cup '99 data set, the hybrid systems obtains a 94.7% TPR and 2% FPR.

Panda and Patra (2009) examined other types of ensemble methods, including bagging (Breiman 1996), boosting (Schapire 1990, Freund 1995) and MultiBoosting (Webb 2000). The ensemble methods use REP trees as their base classifiers. REP (Reduced Error Pruning) utilises a validation set in the pruning process to help prevent overfitting. In comparison, they include NB, and two DTs: a J.48[2] and ID3. They evaluated the techniques on a subset of the KDD Cup '99 data set. The overall error rate is significantly lower for all ensemble methods compared with the other techniques. Bagging performed the best of the ensemble techniques. However, in concurrence with previous observations in the literature, the DT classifiers (including ensembles) perform worse on the minor classes compared with NB.

Depren *et al.* (2005) examined a hybrid of a J.48 DT and Self Organising Map (SOM), in which the former is applied to misuse detection and the latter to anomaly detection. A rule based system was also

[2] The J.48 algorithm is a Java implementation of the C4.5 algorithm (Quinlan 1993), which is included in Weka (Witten and Frank 2005).

employed as a decision support module, to decide which classification output of the two techniques should be chosen. Their experiments on the 10% version of the KDD Cup '99 data set shows the DT unable to detect an attack (*ftp_write*), which the SOM does detect, however, obtaining higher false positives overall. As with the results of ensemble approaches discussed above, this hybrid is also more successful than the DT and SOM alone. Other applications of SOMs to IDSs are discussed further in Section 3.6.3.

3.5.3 Other decision tree applications

Decision trees have not only been applied to intrusion detection as detectors. For example, they have been used as a means of feature selection (Chebrolu *et al.* 2005) and generating rules for expert systems (Sinclair *et al.* 1999).

Kruegel and Toth (2003) utilized DTs to speed up the matching process of rules in signature based systems. They argue that rule bases are becoming so large that common sequential matching approaches are inadequate, *i.e.*, do not scale well enough with the number of rules. Kruegel and Toth propose changing the matching approach from a rule-to-rule process to feature-to-feature, to avoid dealing with each rule individually. All the rules are considered as one set, on which a partitioning process is initiated that groups the rules into smaller sets (clusters) according to their feature specifications. They map these partitions to a DT, in which the complete (initial) set of rules represents the tree's root node, and the subsets become its children. The tree is built so that each leaf node represents a single rule, or *"a number of rules that can not be distinguished by any feature"* (Kruegel and Toth 2003).

Kruegel and Toth implement the proposed matching approach in Snort 1.8.7, and compare the performance with Snort 2.0, which includes a new matching algorithm that is more similar to theirs. They adopt the DARPA99 data set to evaluate the two systems. Their approach obtains a speedup of 40.3% on average, compared with Snort 2.0, with no loss of accuracy. Furthermore, the speedup becomes more significant with an increasing number of rules. Another significant benefit of the approach is that it can perform parallel matching of the features.

3.6 Artificial neural networks

Here, the term Artificial Neural Network (ANN) encompasses a range of models, including Multi Layer Perceptrons (MLPs) (Haykin 1998, pp. 156–255) and Self Organising Maps (SOMs) (Kohonen 1989), which are the main models applied to intrusion detection. First, some general pros and cons of ANNs are discussed in Section 3.6.1. The majority of the misuse detection applications of ANNs are implemented as feed forward MLPs (Cannady 1998, Endler 1998, Jing-Xin *et al.* 2004, Moradi and Zulkernine 2004, Mukkamala *et al.* 2002, Mukkamala and Sung 2003, Pan *et al.* 2003), which are considered in Section 3.6.2. Most of the misuse detection applications are network based, whilst host based applications are typically emplemented as anomaly detection systems. There are examples of MLPs being applied to anomaly detection, but SOMs have been more widely used (Section 3.6.3). Section 3.6.4 discusses other ANN models applied to intrusion detection.

3.6.1 Pros and cons

ANNs have several desirable properties for intrusion detection. Their ability to learn complex patterns in data sets and generalise from known patterns to new (Haykin 1998) makes them successful for both misuse and anomaly detection. Other benefits include their flexibility with respect to noisy/missing data (Cannady 1998) and that some networks are capable of continuously learning during run-time (Cannady 1998, Jing-Xin *et al.* 2004). Initially, statistical techniques were used to perform anomaly detection. However, ANNs now provide a good alternative (Cannady 1998). Debar *et al.* (1992) state that a drawback of a statistical component is that assumptions needs to be made on the distribution of the data, which is not the case with ANNs. Furthermore, Debar *et al.* also state that ANNs are less sensitive to the selected input data (features), *i.e.*, if a feature is irrelevant, the ANN is capable of learning to ignore it. However, Cannady (1998) highlights two drawbacks of using ANNs: (1) the training requirements (large amounts of training data are necessary, and the performance of the network is directly dependent on this), and (2) they are black boxes. Furthermore, a third drawback is determining the topology of the ANN, which is difficult and time consuming; mostly done *ad hoc* or optimised with an evolutionary algorithm (Debar *et al.* 1992, Lacerda *et al.* 2001, Öztürk 2003, Yao 1993).

3.6.2 Multi layer perceptrons

The topology of MLPs impacts significantly on the performance, which is therefore discussed in a separate section below. Thereafter, the applications to misuse and anomaly detection are discussed in their respective sections that follow.

3.6.2.1 Topology

The topology of the MLP is important, not only to the performance of the network, but also for how the intrusion detection is achieved. For example, there are many different ways in which an MLP can output its classification, which gives different properties to the intrusion detection. One approach is to use a single output neuron that gives a binary classification to signify whether an attack has been identified or not (Cannady 1998). Ghosh and Schwartzbard (1999) used one neuron to classify intrusive behaviour, but as a real valued output in the range [0,1] to signify the likelihood of an attack.

Bouzida and Cuppens (2006b) also consider the range of the output given, and only accepts the classification if the value of the output exceeds a certain threshold. This can be used to adjust the ability of the MLP to detect new attacks. Jing-Xin *et al.* (2004) employ a different strategy to classify new attacks, by adopting an MLP with three output neurons, each of which is responsible for one of three classifications: *normal, attack* or *unknown*. The two former are self explanatory, but the latter is assumed to be an intrusion (one that is unknown to the system). This 'unknown' category is used to extract new attacks that will be used for future training of the ANN (at which point it becomes a known attack).

MLPs are capable of detecting specific attacks, as explored by Endler (1998) and Moradi and Zulkernine (2004). Endler achieves this by making use of multiple networks, whilst Moradi and Zulkernine use a single network for which each attack type was represented by one neuron in the output layer. The latter approach has become more common in recent years, but instead of classifying each individual attack, the 5-class taxonomy presented in Section 2.2 on page 8 is adopted. See, for example, Bouzida and Cuppens (2006a;b), Faraoun and Boukelif (2007), Mukkamala *et al.* (2002), Peddabachigari *et al.* (2007) and Sabhnani and Serpen (2003).

Most studies adopt three layered MLPs (Cannady 1998, Jing-Xin *et al.* 2004, Moradi and Zulkernine 2004, Pan *et al.* 2003, Shum and Malki 2008), which have been mathematically proven to be capable of approximating any continuous function given an infinite number of neurons in the hidden layer (Hornik *et al.* 1989). However, four layers are necessary to approximate discontinuous functions (Sontag 1992), and are generally considered more promising for more complex problems (Pinkus 1999). Some studies have examined more than three layers, but the difference in results are insignificant (Bouzida and Cuppens 2006b, Moradi and Zulkernine 2004, Mukkamala *et al.* 2002, Mukkamala and Sung 2003).

3.6.2.2 Misuse detection

MLPs have been widely employed as classifiers to perform network based misuse detection, and most studies in the last decade have evaluated the performance on the KDD Cup '99 data set. However, as noted in Section 3.5.2, there are some discrepancies in the findings reported in the literature due to methodological differences in the studies; more specifically, different subsets of the data have made a significant impact on the results. Hence, direct comparisons across the studies discussed here cannot be made, as discussed further in Chapter 4 on page 54.

MLPs obtain very similar results to DTs, which is reflected in studies that compare the two (Bouzida and Cuppens 2006b, Sabhnani and Serpen 2003). Therefore, as with the DT, other techniques such as NB and clustering detect more *U2R* and *R2L* attacks. However, Sabhnani and Serpen (2003) found that the MLP obtained the best performance on *Probing* attacks compared with 8 other machine learning techniques. The MLP detected 88.70% *Probing* with a false alarm rate of 0.4%. In comparison, a C4.5 DT detected 80.80% with a false alarm rate of 0.7%, and *k*-means clustering detected 87.60% with a 2.6% false alarm rate. *Probing* and *DoS* intrusions were the focus of an investigation conducted by Moradi and Zulkernine (2004), examining learning of *Neptune* (*SYN flood*, *DoS*) and *Satan* (*Probing*) attacks (plus normal traffic). They obtained detection rates of approximately 90% with a four layered feed forward MLP.

Bouzida and Cuppens (2006a, 2006b) include a threshold parameter to the classification process of an MLP to determine whether predicted classifications should be accepted or not. This was implemented by checking if the value of the firing neuron exceeds this threshold. If not, the instance is classified as a 'new' class, and is considered an anomaly that requires further investigation. They observed that the accuracy increased when implementing this threshold, however, at the expense of more intrusions being detected as the 'new' class. This had an insignificant effect on *U2R* and *R2L*, however, which were still largely classified as normal traffic.

One of the first applications of MLPs to intrusion detection was to perform host based misuse detection. Instead of treating the task as a classification problem, Debar *et al.* (1992, 1992) employ a recurrent MLP to perform time series forecasting. The MLP is given sequences of commands and attempts to predict the next command. When the next command is obtained, this prediction is evaluated and the network is adjusted with a backpropagation algorithm if the prediction was wrong. Testing was conducted on sequences of UNIX commands gathered from an anonymous user. An interesting observation is that they obtain higher error rates on predicting some commands than on others, and there was a trend to confuse similar commands such as *sh* and *csh*. However, the approach showed promise, particularly since the system was able to adapt to new users, which Debar *et al.* emphasise as a benefit over statistical approaches that need to be configured for each new user.

Endler (1998) also employ a MLP to perform host based misuse detection, analysing the Solaris Basic Security Module (BSM) to monitor user behaviour as described by the system signals. One focus of this study was to explore the generalisation ability of the ANN. As Endler trained different networks for specific attack types, they were then tested on data from all types of attacks. The data was gathered from four simulated users during six weeks. Sequences of signals from this data were then grouped into events, which were labelled as 'normal' or 'intrusive'. Based on this data, all networks were first trained with the same normal data. Thereafter, each network was trained to detect specific attack types. No single network was able to successfully classify all other attacks, but Endler identified three combinations of attack types that provided sufficient coverage. The results indicate that the best network missed approximately 25% of the attacks during testing (the worst one nearly 60%), but with no false negatives. To obtain a broader coverage of intrusions, Endler hybridised the ANN with a histogram classifier[3] to perform anomaly detection. By doing so, all attacks were detected, but a the expense of a significant amount of false negatives.

[3] The histogram classifier is a statistical likelihood classifier, which Endler implemented according to Duda and Hart (1973).

3.6.2.3 Anomaly detection

Although MLPs are capable of performing anomaly detection, there are few examples of this in the literature. Ryan *et al.* (1998) propose using an MLP for host based anomaly detection, to analyse user behaviour based on UNIX commands. Their approach is to build daily profiles of each user based on the commands that are executed; more specifically, the number of times a user executes the respective commands. The 100 most commonly used commands were chosen, thus, giving a 100-dimensional vector of command frequency intervals for each user, which is used as input to the MLP.

Ryan *et al.* gathered normal data from 10 users on a NetBSD system during 12 days. They validated the MLP with 4-fold cross validation, using 65 randomly chosen vectors (from 8 days) to train the MLP, and testing on the remaining. To simulate anomalous behaviour, they randomly generated 100 vectors that were also used for testing. On average, the system correctly identified the legitimate users 93% of the time and 63% of the randomly generated vectors were classified as intrusive.

Ghosh and Schwartzbard (1999) report on an investigation of an MLP used for application based anomaly and misuse detection. In addition to the MLP, they employ a leaky bucket algorithm to facilitate some form of a memory mechanism since it is necessary to classify sequences of events. One network was trained for each process/program. They evaluated the MLPs on the DARPA98 data, on which the anomaly detection obtained best results of 77.3% true positives and 3.6% false positives. The misuse detection obtained nearly 20% false positives at approximately the same true positive rate. This, they argue, is due to limited intrusive data to learn from.

Although Ryan *et al.* (1998) and Ghosh and Schwartzbard (1999) demonstrate that the MLP is capable of performing anomaly detection, the approaches are obsolete. First, it is not sufficient to perform offline intrusion detection once a day, which was considered by Ryan *et al.* Second, other algorithms have been shown to be more successful in performing anomaly detection. As an alternative to using a MLP with a leaky bucket to facilitate a memory function, Ghosh *et al.* (1999) extend the work in (Ghosh and Schwartzbard 1999) by employing an Elman network (a recurrent ANN). This network gave significant performance gains, and in (Ghosh *et al.* 2000) was considered for real-time intrusion detection. Another ANN that lends itself better to anomaly detection is the self organising map, which is discussed further in Section 3.6.3.

3.6.3 Self organising maps

Differing from those ANN models discussed above, the SOM is an unsupervised model, *i.e.*, it does not need labelled training data to build its model. This is a desirable trait for intrusion detection, since labelling thousands, or even millions, of records is a laborious task and mislabelleing can occur. Furthermore, the SOM is more commonly applied to anomaly detection than any of the other ANN models.

The SOM can be referred to as a clustering technique, which has the particular benefit of being capable of producing lower dimensional representations of multi-dimensional data, in what is referred to as a map. Höglund and Hätönen (1998) implemented a SOM as a visualisation tool for user profiling to help detect intruders. The SOM is trained on normal user data, which forms clusters of common behaviour of the user(s). Deviations from the main cluster(s) signify possible intrusions. In their system, if an intrusion is suspected, further, detailed, data can be examined more clearly, such as CPU usage and which commands have been executed. In one example that Höglund and Hätönen provide, one deviation showed more heavy CPU usage and the use of the finger command, which was never used before in the 'normal' behaviour of that particular user. They also observe that some users have more than one cluster of normal behaviour, which they explain as the user working on different projects or possibly an observation of professional and private use of the computer.

A SOM was applied to network based anomaly detection by Depren (2005), as a component of a hybrid IDS in addition to a DT and a RBS. The DT performs misuse detection and the RBS determines the final output based on individual outputs from the two former techniques, as discussed previously in Section 3.2.4 on page 21. The system was tested on the 10% version of the KDD Cup '99 data set, in which normal data was extracted to train the SOM. Three different SOMs were trained for the three different types of network protocols: TCP, UDP and ICMP. If both the SOMs and DT detect an attack (or just the DT), the attack is classified by the RBS. If only the SOM detects an attack, the RBS still defines the output as an attack, but as an unclassified attack. In line with the findings of Pan (2003), the DT does not detect the *ftpwrite* (*R2L*) attack, but the SOM does. However, with more false positives overall.

As briefly discussed in Section 3.4.2 on page 24, Thames *et al.* (2006) propose a hierarchical hybrid IDS for network based intrusion detection, comprised of a SOM and NB. The two techniques are more integrated than typical hybridisations where each would give an independent classification. The SOM can be considered an anomaly module, as it is mainly trained on normal data. However, Thames *et al.* adopt a SOM algorithm for supervised learning, so that it can learn and classify a proportion of intrusive data, which they implement as follows: *"...an 'attribute' variable was assigned to each node in the SOM grid. This attribute was set as either 'normal' for data labeled as normal and 'abnormal' for all other data vectors regardless of the associated attack type"*. During testing, the 'winning' neuron will thus be able to give a classification of *normal* or *intrusion*.

Thames *et al.* (2006) first train the SOM on a selection of data consisting of approximately 90% normal data and a selection of instances from each type of attack. The SOM appends its classification output to each training instance, which is used to train the NB classifier. Thus, NB learns the classification behaviour of the SOM as well. After this training phase, the system is employed in the same manner, *i.e.*, the SOM first filters the data and appends its classification to the analysed network data, which is then passed on to the NB classifier. Experiments on a subset of the KDD Cup '99 data set demonstrated that the Hybrid model improved the accuracy by approximately 3% compared to only using NB.

3.6.4 Other ANN models and Hybridisations

Differing from the other ANN applications discussed in this section, Cannady (2000) proposes an IDS with an adaptive ANN, a Cerebellar Model Articulation Controller (CMAC) (Albus 1975), which is capable of learning new intrusive behaviour at run time. The CMAC was employed to recognise different types of *DoS* attacks, giving an output value in the range [0,1] to signify the likelihood of an attack. The experiments were conducted in several phases to examine *inter alia* learning specific attacks, learning to detect new attacks, and detecting previously learned (old) attacks. First, the CMAC learned a *Ping Flood* attack. After the first presentation of this attack, the network had a very high error rate. However, this decreased immensely after the second presentation, to ~2.2%. Thereafter, the CMACs generalisation ability was tested, in an attempt to recognise a *UDP packet storm* attack. After the initial weights, the error percentage was very high, ~93.3%, but, again, this decreased significantly after the second presentation of the attack, to ~2.2%. After the the CMAC had learned the *UDP packet storm* attack (in addition to the *Ping Flood* attack from the beginning), another *Ping Flood* attack was introduced to determine whether the previous knowledge of the *Ping Flood* attack remained. Indeed, the error rate was as low as 0.038% after the first presentation of the new *Ping Flood* attack.

Several researchers have examined the use of evolutionary algorithms to optimise ANNs for intrusion detection. Such algorithms have been applied in different ways, *e.g.*, as a training algorithm for an MLP (Michailidis *et al.* 2008), to train and perform feature selection for a radial basis function network (Hofmann and Sick 2003), or to optimise the entire structure of the ANN (Han and Cho 2006). These applications are discussed further in Section 3.12.4 on page 51.

3.7 Support vector machines

Similar to the MLP, Support Vector Machines (SVMs) (Burges 1998, Cortes and Vapnik 1995) are supervised learning algorithms, which have been applied increasingly to misuse detection in the last decade. One of the primary benefits of SVMs is that they learn very effectively from high dimensional data (Boser *et al.* 1992). Furthermore, they are trained very quickly compared with MLPs. For example, Mukkamala *et al.* (2002, 2003) conducted a comparative study of feed forward MLPs and SVMs for misuse detection. Almost identical detection rates were obtained, and the SVM was trained in 17.77 seconds compared with 18 minutes for the MLP (Mukkamala *et al.* 2002).

Applications of SVMs to misuse detection and anomaly detection are discussed in their respective sections below.

3.7.1 Misuse detection

Most SVM algorithms are binary classifiers, which is sufficient when only distinguishing between normal and intrusive data, as in (Mukkamala *et al.* 2002). Although more recent SVM algorithms have been proposed that directly support multi-class learning, *e.g.*, (Crammer and Singer 2002), a common approach is to combine several two-class SVMs (Duan and Keerthi 2005). For example, Mukkamala and Sung (2003) applied SVMs to network based intrusion detection, adopting the five class taxonomy presented in Section 2.2 on page 8, thus requiring five SVMs. For each SVM, the training data is partitioned into two classes so that one represents an original class and the other class represents the remaining, *e.g.*, *Normal* and *all intrusions*, or *Probing* and *Normal and the other attacks*, *etc*. The combination technique adopted is a winner-takes-all strategy (Duan and Keerthi 2005), in which the SVM with the highest output value is taken as the final output.

Peddabachigari *et al.* (2007) conducted an empirical investigation of SVMs and DTs, in which they analysed their performance as stand alone detectors and as hybrids. Two hybrid models were examined, a hierarchical model (DT-SVM), with the DT as the first layer to produce node information for the SVM in the second layer, and an ensemble model comprising the standalone techniques and the hierarchical hybrid. For the ensemble approach, each technique is given a weight according to detection rate of each particular attack type during training. Thereafter, when the system is tested, only the technique with the largest weight for the respective attack prediction is chosen to output the classification. The approaches were tested on the KDD Cup '99 data set. Their results indicate that the ensemble performs better on two attack classes, *Probing* and *R2L*, and equally as good as the other techniques on the the other attack types. The hybrid DT-SVM performs better, or equally as good as the SVM alone. However, the DT performs better on *Probing*, *U2R* and *R2L*. SVM and DT-SVM perform poorly on *U2R* and *R2L* compared with the DT and ensemble.

Due to the magnitude of data involved in network based intrusion detection, Khan *et al*. (Khan *et al.* 2007) propose a hybrid of SVM and clustering to shorten the training time. A hierarchical clustering (tree) algorithm is employed to locate boundary points in the data that best separate the two classes, which are then used to train the SVM. This is an iterative process, in which the SVM is trained on every new level of cluster nodes in the tree that is being built. In each iteration, support vectors are calculated and the SVM is tested against a stopping criterion to determine if a desirable threshold of accuracy has been exceeded. If not, the iterative process continues.

Khan *et al*. (Khan *et al.* 2007) evaluate their hybrid SVM / clustering algorithm on the DARPA98 data set. Their algorithm was trained in 13.18 hours, which is approximately 5 hours shorter than a basic SVM algorithm. They also improve the accuracy, mainly due to correctly classifying more *DoS* attacks. However, the FPR increased by approximately 3%.

Another modified SVM was proposed by Song *et al*. (2002), referred to as a *Robust* SVM (RSVM), developed to better deal with noise. The RSVM was applied to host based intrusion detection by Hu *et al.* (2003), analysing a subset of BSM audit data from the DARPA98 data set. They experiment with a clean training set, consisting of 300 normal processes and 28 intrusive, and a noisy training set, consisting of 316 normal processes (16 mislabelled) and 12 intrusive. The same test set was employed in both experiments, consisting of 5285 normal processes and 22 intrusive. The RSVM obtains the same true positive rate at 1% false positives as an SVM on the clean data. However, at 3% false positives, the RSVM obtains 100% true positives, whilst the SVM only obtains this at 14.2% false positives. Furthermore, on the noisy data, the SVM performs very poorly, obtaining merely 60% true positives at 10% false positives. The RSVM, however, obtains 100% true positives at 8% false positives. Another benefit of the RSVM is that it produces less support vectors, which makes it a quicker algorithm. For example, on the noisy data, RSVM required 15 support vectors, compared with 40 for the SVM. This resulted in a speedup of 58% on the test set.

3.7.2 Anomaly detection

Kim and Cha (2004) and Seo and Cha (2007) applied SVMs to host based anomaly detection of masquerades. Both studies analyse sequences of UNIX commands executed by users on a host. Kim and Cha applied a SVM with a Radial Basis Function (RBF) kernel, analysing commands over a sliding window. They adopt the data set used in (Schonlau *et al.* 2001), which gave a detection rate of 80.1%. This was over 10% higher than other techniques applied to this data (Seo and Cha 2007), however, with the highest FPR (9.7%). Seo and Chan examine two different kernels, K-gram and String kernel, which yield higher detection rates; 89.61% and 97.40%, respectively. The drawback is the same as with the RBF kernel employed by Kim and Cha, that the FPR is even higher; above 20% for the String kernel. Seo and Chan also examine a hybrid of the two kernel methods, which gave nearly identical results as Kim and Cha (2004) with a RBF kernel.

The various methods that Schonlau *et al.* (2001) apply to perform masquerade detection can be seen to obtain different performance trade-offs in detection rates and FPR. Of the results that Seo and Cha (2007) synthesise, Maxion and Townsend (2002) appear to obtain a better trade-off with NB, obtaining a 61.5% detection rate with 1.3% false positives. However, in practice, Maxion and Townsend argue that the false positive rate still needs to be lower.

An unsupervised one-class SVM was proposed by Schölkopf *et al.* (2001), which has been adopted in several studies, comparing its performance with clustering techniques. Details of these studies are discussed in Section 3.10.1 on page 42.

3.8 Artificial immune systems

Artificial Immune Systems (AISs) have been extensively researched in the last decade, mainly for anomaly detection. An overview of AIS models is provided in Section 3.8.1. Much research has been conducted on using negative selection, as that model lends itself conveniently to anomaly detection. However, within a decade of the proposition of negative selection, several researchers came to the conclusion that the model has problems with scalability, limiting its application to real problems. Consequently, some researchers considered alternative models, whilst others have, in recent years, proposed enhancements to negative selection to address scalability. This is an interesting aspect of applications of AIS, which are reviewed in Section 3.8.2. Another aspect of AIS is their capability of being distributed. This is a desirable trait in intrusion detection, which is discussed in Section 3.8.3. Finally, hybridisations incorporating AIS are discussed in Section 3.8.4.

For a complimentary review of AISs applied to intrusion detection, refer to Kim *et al.* (2007), and reviews by Dasgupta *et al.* (2003) and Timmis (2007) for a general treatment of AIS.

3.8.1 AIS models

Four main AIS models can be extracted from the literature (Dasgupta *et al.* 2003, Galeano *et al.* 2005, Kim *et al.* 2007, Timmis 2007):

- Immune/idiotypic networks,
- negative selection,
- clonal selection, and
- danger theory.

The majority of applications of AIS to intrusion detection employ negative selection (Kim *et al.* 2007), which therefore forms a natural focus of this review.

Kim *et al.* (2007) offer a list of benefits of AIS that are desirable for intrusion detection:

Distributed: The very nature of AIS is distributed, as it is comprised of several processes. Failure of one such process does not stop the AIS from conducting intrusion detection.

Self-organising: This is a trait that links AIS to machine learning. Not only does AISs have the ability to learn data by means of a supervised training process, they can also adapt and learn over time.

Lightweight: It is not as memory and CPU dependent as techniques such as CBR, since intrusive behaviour is not necessarily checked against a large data base of intrusion knowledge.

Multi-layered: An AIS can be designed with many layers, which can be assigned different tasks or run in different locations.

Diverse: A range of intrusion detectors can be spread out across a network, giving a broad coverage, whilst still remaining lightweight.

Disposable: The AIS is not dependent on a single process; if one is disposed of, it can automatically be replaced by another.

AISs approach anomaly detection in a different way to Artificial Neural Networks (ANNs), though both are trained to gain an understanding of what is considered normal. AISs typically obtain an understanding of what normal behaviour is and refers to this as *'self'*. Anything that deviates from this, *'non-self'*, is considered an intrusion. There are several alternatives ways to achieve this. Negative selection, as proposed by Forrest *et al.* (1994), has been popularly adopted and applied to anomaly detection. In short, as the AIS obtains an understanding of *self*, detectors (corresponding to T-cells in the human body) are generated in such a way that they do not match *self*. The assumption is that the detectors will match intrusions. These detectors are used to detect intrusions, in contrast to checking the entire model of normal (self), as would be the case with ANNs.

3.8.2 Using negative selection

Negative selection was proposed by Forrest *et al.* (1994), and was popularly used in AISs applied to anomaly detection until early 2000. Thereafter, researchers have started to analyse and question whether negative selection is appropriate for real applications of anomaly detection and consider other alternatives. However, negative selection has once again received more attention in recent years.

Dasgupta and Attoh-Okine (1997) surveyed applications of immune systems to computer security (virus detection), anomaly detection, pattern recognition and fault diagnosis. At the time (1997), only negative selection had been applied to anomaly detection. Later, Dasgupta and González (2002) consider positive and negative selection for network based anomaly detection. Positive selection creates an AIS system similar to ANNs, in which the network traffic is compared to the database of *self*. Due to the magnitude of normal data, positive selection is a memory intensive task that may be infeasible in real applications to large networks. Experiments on the DARPA99 data set indicate that positive selection obtains better results, but is slow and requires a large amount of memory. Negative selection obtains good results, detecting 4/5 of the intrusions detected with positive selection, but requires merely 10% of the resources required by positive selection.

Balthrop *et al.* (2002) apply an AIS with negative selection to network intrusion detection, introducing a new matching technique for detectors, *r*-chunks, as an alternative to full length detectors. With *r*-chunks, detectors do not contain a full string of information, but only specific chunks of it. The remaining, not defined, information is considered as wild cards. This saves memory and Balthrop *et al.* (2002) claim it is an easier model to analyse. Results on a small data set show promise, but there were uncertainties concerning how well this scales.

Related to the uncertainties of Balthrop *et al.* (2002), recent research has uncovered issues with negative selection with respect to scalability and coverage of the problem space (Kim *et al.* 2007, Stibor *et al.* 2005, Timmis 2007). Stibor *et al.* (2005) question the feasibility of employing negative selection in a real scenario for anomaly detection. They investigate empirically the performance of an AIS with negative selection, which includes an analysis of the use of hamming distance (binary representation) and Euclidean distance (real valued parameter representation) as affinity metrics. They found that negative selection, using a binary representation and the Hamming distance as an affinity metric, is infeasible for real intrusion detection applications because of scalability issues related to generating detectors. However, as Timmis (2007) state, it is unclear whether the scalability issues are related to the negative selection, as they see it being directly linked to the representation and affinity metric used. Furthermore, although the AIS is capable of detecting all *non-self*, Stibor *et al.* (2005) argue that the AIS requires both positive and negative data to achieve high classification rates. They compare their approach with a SVM, which obtains good results with just knowledge of either positive or negative data.

Kim and Bentley (2001) also considered negative selection when applying an AIS to network anomaly detection, drawing the same conclusions as Stibor *et al.* (2005) regarding scalability. They state that too much computational effort is required to generate enough detectors for a practical scenario, but do suggest that it can be used to filter invalid detectors. Ma *et al.* (2008) argue that the scalability issue of negative selection can be alleviated, as they propose a new, quicker, method of generating detectors using an antigen feedback mechanism. This approach obtains good detection rates, however, they do not present theoretical proof or empirical findings to support their argument concerning improved scalability.

Due to the drawbacks of negative selection, more research efforts have been directed at using danger theory (Kim *et al.* 2007). Whilst most AISs adopt the notions of the *adaptive immune system* in the human body, which can be compared with anomaly detection, danger theory adopts this as well as the *innate* part of human immune systems, which can be considered as misuse detection. Zhang and Liang (2008) stress the importance of this since it is difficult to properly define *self* in practice, and that *self* can change over time, leading to increased error rates. Creating *non-self* from *self* does not imply that it is an intrusion, and, as Kim *et al.* (2007) state, such a system is prone to raising false positives. However, this is a general drawback of anomaly detection.

There have been several successful applications of danger theory to intrusion detection, *e.g.*, (Aickelin *et al.* 2003, Aickelin and Cayzer 2002, Burgess 1998; 2000), as reviewed in (Kim *et al.* 2007), and, more recently, (Amer 2008, Ou and Ou 2009, Xu *et al.* 2007). As another alternative to negative selection, idiotypic networks have been demonstrated to be successful in detecting *DoS* attacks (Ostaszewski *et al.*

2008). Although researchers started exploring alternatives to negative selection, recent research offers alternative ways of generating detectors to address the scalability issues. For example, Ma *et al.* (2008), as discussed above, and several applications of Genetic Algorithms, discussed in Section 3.8.4.

3.8.3 Distributed AIS

The distributed nature of AISs is one of the benefits emphasised by Kim *et al.* (2007). Hofmeyr and Forrest (1999) propose a distributed model for network based intrusion detection based on TCP/IP traffic. Normal traffic is treated as *self*, and *immature non-self* detectors are randomly generated asynchronously. These detectors are grouped into sets, which can exist on a host or in a Local Area Network (LAN). As the immature detectors are created, they are allowed to match any network traffic. If they match a connection, they are deleted according to negative selection, with the assumption that they have connected to normal traffic (*self*). It is assumed that only normal traffic occurs during this immature phase, which can be considered as an additional test for the detector. If it does not match any TCP/IP connections during this immature period, it becomes mature, and is considered a match for *non-self* (an intrusion). These detectors can be distributed across the LAN, one set of detectors per host, which comes with the benefit of obtaining near linear speedups (Hofmeyr and Forrest 1999). However, Hofmeyr and Forrest distribute the detectors according to the behaviour of a protein that, in the body, has the role of transporting protein fragments (peptides) from within a cell to its surface (without breaching the cell membrane). These proteins can bind to a set of peptides, and in a similar fashion, Hofmeyr and Forrest incorporate this into their model by introducing randomly generated *permutation mask*s to the detectors.

The model of Hofmeyr and Forrest (1999) has shown some success and has since been adopted by several authors. For example, Hou *et al.* (2002) only propose a change from binary string detectors to numerical detectors, and Kim and Bentley (2002) introduce dynamic clonal selection instead of negative selection. Dynamic clonal selection is proposed as a means of enhancing the adaptability of the AIS. Kim and Bentley (2002) demonstrate that this new selection technique is capable of performing incremental learning and adapting to novel data during run-time. They also examined some properties of the algorithm, namely the tolerisation period (the period for allowing immature detectors to remain if they do not match any data), the activation threshold and the life span of detectors. Their results indicate that a large toleristaion period is necessary to lower false positives. The two other parameters were shown to be epistatic and dependent on the environment.

Another distributed solution was proposed by Hall and Frincke (2003), applying an AIS to a multi-enterprise misuse management system. Their AIS model consists of two layers, one that is central and creates detectors, and another that is responsible for data gathering, reduction, detection and response, and to forward successful detections to the first layer. Other distributed applications have been achieved by hybridising AIS with Mobile Agents, *e.g.*, (Dasgupta 1999, Machado *et al.* 2005, Ou and Ou 2009, Ping *et al.* 2004, Wang *et al.* 2008a, Yang *et al.* 2009), which are discussed further in Section 3.9.4 on page 40.

3.8.4 Hybrids

Powers and He (2008) propose a network based hybrid IDS comprising an AIS and a SOM. The AIS is employed to perform anomaly detection, whilst the SOM is utilised to extract more information about the attacks that the AIS detects. This they achieve by first training the SOM on intrusive data (unsupervised). After this phase, the neurons (cluster centres) of the SOM are labelled according to their respective attack classes. Clustering has also been incorporated in an AIS, by Sifei and Jiayi (2007), who propose using an unsupervised graph clustering algorithm to generate detectors.

The AIS implementation of Powers and He (2008) includes some modifications of the negative selection approach. First, they altered the representation from binary strings to real valued detectors, as in (Dasgupta and González 2002, González and Dasgupta 2002). Second, they adopt a Genetic Algorithm (GA) to generate detectors instead of random generation. The GA attempts to generate detectors with two objectives: (1) *"to maximise the generality of the detector"*, and (2) *"to minimise the number of self samples in the training data matched by the detector*" (negative selection). They do not, however, perform multi-objective optimisation. Instead, they calculate the fitness as the sum of weighted ratios of each objective, for which the weights need to be set *ad hoc*. Other researchers have also investigated using GAs to generate detectors, *e.g.*, (Haag *et al.* 2007, Ostaszewski *et al.* 2006).

Dal *et al.* (2008) propose a different system architecture comprising an AIS and a GA, which includes a *secondary immune response*. The secondary immune response is achieved by incorporating a memory cell mechanism into their system, which allows it to more quickly detect previously classified attacks. They adopt negative selection and randomly generate detectors using the r-Contiguous bits algorithm. In addition to each detector, they evolve sister detectors, which are utilised in the detection process to determine whether the system should classify a suspected anomaly or not. At least three sister detectors need to be activated for this to occur, and if they do, they are all considered in a process to create a *memory cell*. A number of sister detectors that exceed a fitness threshold are selected to contribute towards a new evolutionary process. At the end of the process, a detector with a higher fitness than the previously selected sister detectors is selected as a memory cell. Dal *et al.* (2008) claim that the approach yields superior performance, but only present an analysis of the memory cell mechanism, as they report on an empirical investigation of the DARPA98 data set.

3.9 Mobile agents

Mobile Agents (MAs) are distributed by nature. They are typically written in a scripting language and roam around a system to perform designated tasks. Several agents may be present in the same system but perform different tasks, which are generally correlated by a higher level monitor.

3.9.1 Pros and cons

In 1994, Chess *et al.* (1994) published a report for IBM about the use of MAs, to determine whether there are any benefits of using them compared with other techniques that can perform the same tasks. They conclude that the tasks that MAs can perform are (generally) possible with other techniques as well, however, there are many technical benefits of using MAs, including *"high bandwidth remote interaction, ease of distributing individual service clients, scalability, lower overhead for secure transactions and robust remote interaction*". Although these points are good as independent advantages, Chess *et al.* (1994) emphasise the combined advantages that MAs offer, which cannot be achieved with other techniques. Jansen *et al*. (1999, 2000) state other benefits of MAs, such as autonomous execution, overcoming network latency, offering robustness and fault tolerance.

There are drawbacks of MAs, however. Chess *et al.* (1994) state that they need particular consideration with respect to security. Jansen *et al*. (1999, 2000) also emphasise this point, but from another perspective, that the can MAs become security threats themselves as they may need to execute with admin rights to perform their tasks. Furthermore, as Chess *et al.* (1994) discuss, the behaviour of the MAs can be similar to that of a virus, and it is not straight forward to deal with this in order to authenticate a MA. Other drawbacks that Jansen *et al*. (1999, 2000) highlight are the volume of code that is required to implement a MA, the speed at which they detect intrusions (due to using scripting languages), and limited methods and tools.

3.9.2 Early work and hierarchical structures

Many applications of MAs adopt a hierarchical architecture. Helmer *et al.* (1998) propose an architecture consisting of three levels of agents, listed here from the lowest:

Data gathering agents: extract information from system calls, the network, authentication events and other functions.

Low-level agents: responsible for classifying recent activities, sharing this information amongst other low level agents and high-level agents. To perform classification, different machine learning techniques can be used, depending on the task of the agents.

High-level agents: maintain a data warehouse of all the information and apply data mining.

The same model has been adopted and improved in a later study by Helmer *et al*. (2003), specifically aimed at making the agents lightweight. This addresses one of the drawbacks that Jansen *et al.* (1999, 2000) highlight, concerning the volume of code that agents often require. The agents in (Helmer *et al.* 2003) implement minimal functionality, to be as lightweight as possible, but the model includes a new feature that allows communication and collaboration between the low level agents. This extension did not compromise the intrusion detection capabilities. Instead there are additional benefits such as reducing the system load when the system is not experiencing any intrusions.

Another hierarchical model has been developed by Zhicai *et al.* (2004), which also consists of three levels of agents. In their architecture, the layers and responsibilities of agents are closely connected to the topology of the network environment, which is somewhat different to Helmer *et al.* (1998, 2003). Starting from the top, agents reside on network level, subnet level and node level. The agents on each level perform nearly identical tasks; the main difference is which other agents they communicate with and report to. The agents on each level will have an awareness of other agents on the same level and agents on the level below and/or above. That is, a network agent communicates only with subnet level agents, whilst subnet level agents can communicate with one another, network agents and detectors at the node level.

All agents in the model proposed by Zhicai *et al.* (2004) have an information and knowledge base, a communication module, an analysis module and a response module. The agents at node level have two additional modules, a sensor and a collector, to allow them to detect and gather data, process this information and forward it to the analysis module. Due to the sharing of information between the agents in the different layers, autonomy is readily achieved with this model. This is possible because higher level agents can detect anomalies in agents in the layer below. If an agent is identified as anomalous, it is replaced by a clone of a normal agent on the same level.

3.9.3 Recent research

The majority of applications of MAs utilise a hierarchical structure of agents, as discussed above, although different researchers propose variations tailored to their application. For example, Krmíček *et al.* (2007) propose an IDS that considers a three level architecture comprising: (1) data collection and preprocessing, (2) intrusion analysis, and (3) administrator interface. To achieve real time intrusion detection, the detection agents do not gather data; instead, they receive preprocessed data from probes in the first layer. These agents collaboratively perform anomaly detection, from which alerts, along with DNS (Domain Name Server) records and other history leading to the alerts, are passed on to user interface agents in the top layer.

Another hierarchical model was proposed for wireless sensor networks (Kachirski and Guba 2003), which employs three main types of agent, namely action agents, decision agents and monitoring agents. The latter can spawn three different monitoring agents, which can exist at different levels: packet, user and system. Data gathered from all of these agents are merged to analyse the entire network for potential intrusions.

Rehak *et al.* (2008) applied MAs to perform real time intrusion detection in high speed (gigabit) backbone networks. They employ collective trust modelling within groups of collaborative detection agents, in which each of the agents contribute to the model of aggregated anomalies (potential intrusions). The result of this process is obtaining ranked clusters of data flows, which helps to focus analysis necessary by a human operator, and reduces the number of alerts.

Deeter *et al.* (2004) propose a different application of MAs, in which the agents function as middleware between different IDSs. The purpose is to be able to combine network based and host based systems, as well as anomaly and misuse detection. This is achieved by distributing agents to network monitors, web servers (and other service providers), and hosts that execute other IDSs. Liu and Li (2008) apply MAs to intrusion detection in a similar manner, to enable decentralised data collection from host and network based IDSs.

Motivated primarily by enhancing the security of the IDS itself, Ramachandran and Hart (2004) were the first to propose a peer-to-peer (P2P) architecture for mobile agents. This decreases the risk of the IDS becoming targeted for intrusion, which is more likely with a central coordinator/manager module. Furthermore, such centralised modules may become overloaded with client requests (Ye *et al.* 2008). In the model proposed by Ramachandran and Hart (2004), the agents monitor virtual neighbourhoods. There is regular, direct communication amongst all agents within each neighbourhood, which allows them to monitor each other. When one agent detects a potential intrusion, it issues a voting process between all neighbours to determine appropriate action. A similar model was proposed by Wang *et al.* (2005), although they exclude the regular communication between neighbours to decrease the computational overhead.

Ye *et al.* (2008) contend that the regular communication between the neighbours is a drawback of the original model of Ramachandran and Hart (2004). Furthermore, they argue that the model is unable to facilitate detection of multiple hosts in a network. Thus, the constraints on information gathering only within neighbourhoods may lead to false negatives. Hence, they propose a migration scheme, in which agents are allowed to collect data from other hosts in the network. Ye *et al.* (2008) compare results from their model with that of Wang *et al.* (2005), and not only does the detection rate improve, the network latency and network loads are lower.

3.9.4 Hybrid systems

There are many examples of hybridisations of AISs and MAs in the literature, which form effective network based IDSs. One of the first models was proposed by Dasgupta (1999), and allows immunity based agents to roam around nodes and routers in a system to monitor the network status. These agents are not restricted in the network and interact dynamically with each other and the network. This freedom does not imply that the agents are considered as independent entities; there is a hierarchy and the agents communicate with each other to collectively detect intrusions. The hierarchy consists of three main types of agents, namely, monitoring agents, communication agents, and decision agents. The monitoring agents reside in the network nodes to gather information on several levels: user, system, process, and packet. Thus, such a system is not restricted to intrusion detection, but can perform general network management and fault localisation. It is suggested that these agents may implement negative selection, as discussed in Section 3.8 on page 34.

The communicator agents are responsible for communication between the other agents, and the decision agents are responsible for deciding which action should be taken depending on the information gathered from the other agents. Decision agents can spawn three different types of agents via communicator agents, depending on the situation: helper agents, killer agents or suppressor agents. Helper agents report information to an administrator, who should then take action accordingly. Killer agents are spawned to take aggressive action against potential intruders, *e.g.*, shutting down a machine or disconnecting a node, killing processes or disabling sessions. The role of suppressor agents is to suppress further action from the action agents to help prevent false positives.

Ping *et al.* (2004) propose a similar MA architecture to that of Dasgupta (1999), consisting of three types of agents that simulate functions of the human immune system. These are monitor agents, decision agents and killer agents. Monitor agents reside on nodes and collect information, which is then filtered and coded for decision agents to reason about. The decision agents are distributed to zones of monitor agents, and collect this information from them to determine whether an intrusion is being executed or not. This is done according to security policies and immune memory. To ensure robustness, an intrusion is only alerted if anomalous behaviour exceeds a certain threshold. Anomalous behaviour can be triggered by failures in the network, so this mechanism helps prevent false negatives. When a potential intrusion is identified, an aggressive response is initiated by producing a killer agent that should stop the intruder by isolating it. This is done by cutting off routing requests and dropping packets from the intruder.

The two models discussed above were not implemented and evaluated empirically, but recent prototype systems have been implemented that incorporate many of the proposed concepts, *e.g.*, Machado *et al.* (2005) and Boukerche *et al.* (2007). In the latter study, the authors strive to obtain real-time intrusion detection by monitoring distributed host based systems with various types of agent. They adopt four main types: monitor agents, delivery agents, reacting agents, and persistence agents. There are three different types of monitor agents, each responsible for monitoring different files of the *Logcheck* tool (available for Linux and Solaris (Anon 2010)), namely attack, event, and security violations. The persistence agent is a new asset to this model, which is employed to ensure data persistence to guarantee data distribution, and to be able to deal with network instability.

Similar hierarchical models have also been adopted in other studies, to allow distributed monitoring of hosts (Wang *et al.* 2008a, Yang *et al.* 2009). Wang *et al.* (2008a) employ immune agents on the hosts to detect intrusions. These communicate with local monitor agents, which analyse the state of the LAN. At the highest level, a central monitor agent supervises the entire network. An empirical investigation at Sichuan University indicates that this system obtains significantly better true and false positive rates compared to an older AIS based IDS called LISYS (Hofmeyr and Forrest 2000). However, they do not provide any details about their experiments. Similarly, although Yang *et al.* (2009) include a section on empirical findings, there are no details about the experiments they have conducted. Therefore, it is not possible to evaluate the significance of their findings. However, an interesting observation is that the recent applications have excluded killer agents, which both Dasgupta (1999) and Ping *et al.* (2004) proposed in their models. Instead, the applications make use of passive responses, by alerting system administrators of potential intrusions.

There are several other hybrid systems that incorporate other techniques or even entire detection systems into a multi-agent framework. For example, Herrero *et al.* (2008) incorporate ANNs to perform intrusion detection in dynamic networks, Wasniowski (2005) utilize Fuzzy agents, Wang *et al.* (2009) propose a hybridisation of MA and clustering for intrusion detection in wireless sensor networks, and Mosqueira-Rey *et al.* (2007; 2009) employ rule-based agents (from SNORT) for network based misuse detection.

3.10 Clustering

One of the main benefits of clustering is unsupervised learning. Labelling of data is not necessary and natural patterns in the data are extracted. Most clustering approaches are unsupervised and are commonly applied to anomaly detection, which is reviewed in Section 3.10.1. It is possible, however, to perform supervised clustering if labelled data is available. This generally obtains better results on benchmark data as they can be employed for misuse detection, as seen in Section 3.10.2. Hybrid approaches involving clustering are discussed in Section 3.10.3.

3.10.1 Unsupervised clustering

There are several applications of clustering techniques to network based anomaly detection, *e.g.*, Portnoy *et al.* (2001), Eskin *et al.* (2002), Guan *et al.* (2003), Leung and Leckie (2005), and Song *et al.* (2008). These studies make two common assumptions about the data: (1) that the vast majority of the data is normal (Portnoy *et al.* 2001) and (2) that the intrusions are statistically different from normal data (Denning 1987). More concretely, Portnoy *et al.* (2001) assume that the normal data makes up more than 98% of the total amount of data. However, there is no data set publicly available that fulfils this assumption, although it may be realistic in a practical scenario. Therefore, these researchers have adopted altered/filtered version of the KDD Cup '99 data set to fit their assumptions. It should be noted, however, that they have done this filtering process differently, and, consequently, their results are not directly comparable.

Portnoy *et al.* (2001) propose a version of single linkage clustering[4], which only takes one pass of the data to create the clusters. The algorithms starts with no clusters. For each instance in the training set, the Eculidean distance to existing clusters is calculated to determine the closest cluster (if any). If this distance is within a predefined threshold, the instance is assigned to that cluster. Otherwise, a new cluster is created with the instance as its centroid. Thereafter, according to assumption one, above, the largest clusters are labelled as 'normal'. The remaining, small, clusters are labelled as 'intrusion'. This obtained true positive rates of approximately 50% with 2% false positives. A modified version of this algorithm was used in a comparative study by Eskin *et al.* (2002), comparing this clustering technique with *k*-nearest neighbour (*k*-NN) and a one class SVM modified to handle unlabelled data (Schölkopf *et al.* 2001). The modification of the clustering algorithm allows instances to belong to more than one cluster. For this comparative investigation, only the training set of the KDD Cup '99 data set was used, filtered so that only 1–1.5% intrusions remained[5]. All three techniques obtained above 90% true positive rates at less than or equal to 10% false positives. The SVM obtains the highest true positive rates with 10% false positives, however, it is the technique that has the greatest decrease in true positive rates with lower false positives. The clustering technique proposed by Eskin *et al.* (2002) obtains a consistently higher true positive rate as the false positive rate is decreased below 4%. At 2% false positives, their technique obtains 66% true positives, whilst *k*-NN obtains 5%, and SVM obtains 5% true positives at 3% false positives (results are not provided for the SVM below 3%).

A more comprehensive approach than those above has been proposed by Leung and Leckie (2005), to perform network based anomaly detection, although they also follow the same underlying assumptions regarding data. Somewhat differently, clusters for intrusions are not produced or labelled during training, as their aim is to build clusters to cover 95% of the data, which, according to their assumptions, should only cover the normal data. During testing, any instance that does not fall within any of the clusters is considered an intrusion. Their approach is a grid and density based clustering technique, similar to pMAFIA (Nagesh *et al.* 2000), which includes a Frequency Pattern Tree (FP-Tree), which they refer to as fpMAFIA. Their algorithm has several phases to build and refine the clusters:

1. Identify frequent instances using the grid and density based clustering algorithm.
2. Build a FP-Tree based on the transformed data from phase 1.
3. Use a *count back* method to extract candidate clusters.

[4] Single linkage clustering is also known as nearest neighbour clustering.

[5] Neither Portnoy *et al.* (2001) nor Eskin *et al.* (2002) offer any details on how this filtering process was conducted, what intrusive data remains, nor how the training and test partitions were selected.

4. Remove duplicate clusters.

Leung and Leckie (2005) tested this approach on the KDD Cup '99 data set, using filtered versions of the full training set and test set. To fulfil assumption one, the training set was filtered so that only a quarter of the instances remained. Furthermore, *Smurf* and *Neptune* intrusions were removed from the test set, which leaves normal with 85% of the total number of instances. They obtain significantly worse results than Portnoy *et al*. (2001) and Eskin *et al*. (2002), only achieving near 100% true positive rate with approximately 40% false positives. With false positive rates below 10%, the true positive rate is below 30%, whilst all the clustering techniques that Eskin *et al*. (2002) implemented obtained more than 90% true positives. However, it is important to stress that lower detection rates are to be expected when the test set is used (Sabhnani and Serpen 2004), which Portnoy *et al*. (2001) and Eskin *et al*. (2002) did not do.

Guan *et al.* (2003) propose a clustering algorithm based on *k*-means, which they call *Y*-means. One of their main contributions is to make the algorithm less sensitive to the chosen value of *k* (the number of clusters). The algorithm does, however, include an additional parameter to determine the threshold for labelling. They report positive results on a subset of the KDD Cup '99 data set, obtaining a 82.32% detection rate with merely 2.5% false positives. However, the performance is strongly dependent on the particular data used. Detection rates with a different subset, as reported by Song *et al.* (2008), are lower, at approximately 60% with 2.5% false positives.

Similar to Guan *et al.* (2003), Song *et al.* (2008) propose a new clustering algorithm based on *k*-means. They have been able to exclude the need to determine the number of clusters, but introduce two new parameters. Based on a selected range of parameter values, Song *et al.* (2008) demonstrate that their approach is more robust than the *Y*-means algorithm. Furthermore, it obtained higher detection rates than *Y*-means, as well as the clustering algorithm proposed by Portnoy *et al.* (2001). They also compared their results with a one class SVM (Schölkopf *et al.* 2001), which gave nearly identical detection rates above a 2.37% false positive rate.

The applications of clustering discussed thus far have performed *stateless* anomaly detection of network connections. Lee *et al.* (2008b) apply clustering to detect distributed *DoS* (*DDoS*) attacks, aiming to identify the different phases of such attacks. After an initial phase of parameter selection, they perform a hierarchical cluster analysis to generate groups of traffic corresponding to normal data and the different phases of the *DDoS* attacks. They evaluate the approach on the DARPA2000 data set, and are able to identify 6 clusters, which were manually labelled based on the data they represented. Two of the clusters represented groups of normal traffic, one cluster for phase one and one for phase two of the *DDoS* attack[6], one cluster representing the actual attack, and one for post attack. Phases three and four of a *DDoS* attack were not detectable because they are local intrusions that do not generate network traffic.

Al-Mamory and Zhang (2009) applied clustering to perform alert aggregation/reduction. They proposed a semi-automated system based on nearest neighbour clustering, aimed at reducing the workload on the system administrators caused by multiple alerts related to the same intrusion(s). To validate the approach, they conducted experiments on the DARPA98, DARPA99 and a 'real' data set. Alerts were gathered from Snort for two months. The data from the first month was used to generate filtering rules, which then were used to classify the alarms from the second month. On average, the clustering reduced the number of alerts by 74%.

3.10.2 Supervised misuse detection

A supervised clustering and classification technique has been proposed by Ye and Li (2001), which aims to learn both normal and intrusive behaviour. Initially, two clusters are created, one for normal data and one

[6] Phase one involves the attacker performing an *IPsweep* of the hosts, and phase two is to *Probe* valid IPs to locate hosts running the *sadmind* daemon which is to be exploited in consecutive phases.

for intrusive data. The centroids of these clusters are calculated as the mean of all the normal and intrusive data respectively. Thereafter, an incremental learning process commences, repeating the following for each instance: find the nearest cluster, which is calculated using correlation coefficients for each dimension; if the label of this instance coincides with that cluster, assign it to the cluster; otherwise, create a new cluster with that instance as the centroid. Ye and Li consider two approaches to classifying data: (1) the nearest of *k* clusters, or (2) the weighted sum of the distances to the *k* nearest clusters, giving an output in the range [0,1].

Ye and Li (2001) tested their clustering algorithm on a data set composed of a small subset of normal instances from the DARPA98 data set and intrusions gathered from private simulations. On this data set, their clustering algorithm outperforms two decision trees (CHAID[7] and CART). Li and Ye (2005) present a new, more robust, version of their clustering algorithm, which includes support for incremental updating of the clusters. They evaluate the new clustering algorithm on the DARPA2000 and KDD Cup '99 data sets. All seven attack sessions of the DARPA2000 data set were detected with no false positives, however, they do not provide details on how they analyse the classification of the individual events to determine the classification rates for the attack sessions.

The results Li and Ye (2005) present on the KDD Cup '99 data set are not as positive. At approximately 1% false positives, they are able to detect 72.8% *Probing*, 97% *DoS*, 9.2% *U2R* and 6.3% *R2L* attacks. These results are not competitive with the results obtained with other supervised machine learning algorithms. However, the particular benefit of this approach is that incremental and continuous learning is possible.

Another study presents some more positive findings for clustering applied to the KDD Cup '99 data set. Sabhnani and Serpen (2003) benchmark 9 machine learning techniques, including *k*-means clustering. Different techniques were found to excel at detecting different types of attacks, and *k*-means performed best on two out of the four attack classes: *DoS* and *U2R*.

3.10.3 Hybrid approaches

Spinosa *et al.* (2008) propose utilising both supervised and unsupervised clustering for network based intrusion detection. First, they perform supervised learning of normal traffic. Thereafter, the system enters a second phase comprising unsupervised continuous learning, employed to detect novel intrusions. For this, they adopt a *k*-means algorithm, but include a mechanism to dynamically set the value of *k*, as this may change with time. They validate the approach on the 10% version of the KDD Cup '99 data set, which, at 19% false positives, detected 56% *Probing*, 99% *DoS*, 92.1% *R2L* and 71% *U2R* attacks.

Leon *et al.* (2004a) employ a Genetic Algorithm (GA) to perform clustering, which they achieve by incorporating a niching mechanism. After an initial run of the GA, the clusters are refined using a Maximal Density Estimator (MDE) (Nasraoui and Krishnapuram 1999) to optimise the centres of the clusters. A final step involves fuzzy logic to transform the crisp cluster boundaries to degrees of membership. That is, every sample will belong to every cluster to a certain degree. This approach is used to perform anomaly detection, in which all clusters are made up of normal data. Hence, it is not important which cluster a normal instance is assigned to, as long as it is within a certain distance. If an instance's distance does not fall within a defined threshold of a cluster, it is assumed to be an intrusion. Leon *et al.* tested their approach on a subset of the KDD Cup '99 data set, using only the 10% training set. Training was performed on 5000 normal samples, and 40% of the data set was used for testing. They only use the numerical features in the data set, which gives 33 features. Furthermore, they investigate the effects of applying Principal Component Analysis (PCA) to the data set, which reduced the features to 21. The true positive and false positive rates are 99.20% and 2.20% with PCA, respectively, and 94.09% and 7.84% without.

[7] CHAID is an abbreviation for Chi-squared Automatic Interaction Detector (Kass 1980).

The true positive rates Leon *et al*. (2004a) obtain are higher than those of Portnoy *et al*. (2001) and Eskin *et al*. (2002). However, neither of these three studies give sufficient information about the data that they use for training and testing. Furthermore, even though the definition of the detection rate metric that Leon *et al*. (2004a) adopt is the same as true positives, their figures do not add up, as they do give individual detection rates for each of the intrusion classes. According to those figures, the true positive rate would be 94.83% with PCA, not 99.20%.

Other population based search/optimisation techniques have been used for clustering, see *e.g.*, (Tsang and Kwong 2005a;b, Ramos and Abraham 2004, Feng *et al.* 2006), which are further discussed in Section 3.12.3 on page 50.

As previously discussed in this review, several researchers have incorporated Fuzzy logic into their algorithms. This is also the case for clustering. For example, Shah *et al.* (2003) consider fuzzy clustering for network based detection, taking an outlier detection approach similar to the unsupervised clustering applications discussed above. Chimphlee *et al.* (2006) apply Fuzzy Rough Clustering to unsupervised anomaly detection, which utilises fuzzy logic and rough set logic. This algorithm is based on a *C*-means algorithm, which assigns crisp, binary, values to cluster memberships. The extension to Fuzzy *C*-means allows memberships across ranges, in which each data instance belongs to each cluster to a certain (fuzzy) degree. The final extension to a rough *C*-means, uses a three part classification to determine membership: a lower approximation, boundary, and a negative region. Fuzzy clustering has also been employed to build DTs Makkithaya *et al.* (2008). In principle, this is done in a similar manner to the non-Fuzzy approach to building DTs based on clustering, as per Kruegel and Toth (2003), which is described in Section 3.5.3 on page 28. Makkithaya *et al.* (2008) argues the benefits of using clustering to build DTs as: *"In contrast to C4.5-like trees, all features are used once at a time, and such a development approach promotes more compact trees and a versatile geometry of the partition of the feature space."*

Other hybridisations include distributed systems including AIS (Sifei and Jiayi 2007) and MAs (Zhang *et al.* 2005, Wang *et al.* 2009). The two latter studies focus on intrusion detection in wireless *ad hoc* networks. Other studies using clustering for *ad hoc* networks include Ahmed *et al.* (2006) and Elhdhili *et al.* (2008), as discussed in Section 2.4 on page 14.

3.11 Hidden markov models

The Hidden Markov Model (HMM) is the only machine learning technique that explicitly learns state based classification (sequential modelling), and, thus, is not limited to conducting stateless intrusion detection. This enables HMMs to perform more comprehensive intrusion detection and detect multi-stage intrusions (Lee *et al.* 2008a, Ourston *et al.* 2003). Despite this advantage, HMMs have not been as extensively applied to intrusion detection as the other machine learning techniques reviewed here.

Section 3.11.1 highlights aspects of the implementation of HMMs, followed by a discussion of the performance results reported in the literature, in Section 3.11.2. A diverse range of applications and recent research are discussed in Section 3.11.3.

3.11.1 Implementation considerations

Researchers typically train HMMs on sequences of events recorded over a 'sliding window' of typically 3-10 events (Al-Subaie and Zulkernine 2006, Hoang *et al.* 2009, Wang *et al.* 2008b). Consequently, HMMs cannot offer comprehensive, temporal, event correlation. Furthermore, Rabiner (1989) mention some limitations of HMM in the context of speech recognition, which are important for intrusion detection as well: HMM assumes that *"successive observations are independent* [and that] *the probability of being in a given state at time t only depends on the state at time t-1*".

As with most other machine learning algorithms, there are parameters of HMMs that need to be determined empirically. HMMs cannot derive the number of states from the learning process; it needs to be defined *a priori* and remain static, and, in this domain, it is not straightforward to determine the number of states (Lane and Brodley 2003). Lane and Brodley (2003) found that the performance of HMM is strongly affected by the number of states, being very poor with few states (they examined as few as 1 state) as well as many (50+). Furthermore, in the context of user profiling for host based intrusion detection, Lane and Brodley found that the ideal number of states varied depending on the user. However, states in the mid-range generally led to good detection rates.

3.11.2 Performance

Although HMMs can yield high classification rates, they are found to be very computationally heavy, both for training and classification (Lane and Brodley 2003, Wang *et al.* 2008b, Warrender *et al.* 1999). Warrender *et al.* (1999) studied the performance of a HMM compared with a selection of more simple techniques, namely an enumeration based algorithm, a frequency based algorithm and a rule induction technique (RIPPER (Cohen 1995)). These techniques were tested on a selection of system call data sets, on which the HMM gave the best overall performance. However, the HMM was found to be significantly more computationally heavy, whilst the simple enumeration based algorithm performed sufficiently well in comparison. Wang *et al.* (2008b) compare the performance of HMM with PCA, finding that the HMM provides higher detection rates, but is infeasible for real-time intrusion detection.

Lane and Brodley (2003) compare HMMs with IBL, also finding that HMM yields better accuracy on detecting intrusions. However, there was no gain in classifying normal behaviour. Therefore, they put their results in the context of a hierarchical combination of classifiers, in which a ‘light’ classifier (such as IBL in this case) can serve as a filter of more obvious data (normal behaviour in this case) at a lower level, then at a higher level, a more comprehensive technique (such as HMM in this case) can be employed to classify the remaining data. Despite the longer training time necessary, Ourston *et al.* (2003) found that HMM was able to obtain higher classification rates than an ANN and DT with few training instances. However, once the number of training examples were increased, the DT started to outperform the HMM.

Similarly to recent research efforts that address the scalability issues of AISs, researchers have proposed training algorithms for HMMs that are significantly quicker. For example, Hoang *et al.* (2009) propose a training algorithm that reduced the training time four times, and also reduced the memory requirements of the HMM. This algorithm was a modified version of the algorithm proposed by Davis *et al.* (2002) for learning from different observation spaces, which Hoang *et al.* (2009) modified for incremental learning. Although they obtain significant speedups, they conclude that HMMs are still not fast enough for real-time intrusion detection.

3.11.3 Applications and recent research

HMMs have very diverse applications in intrusion detection compared with other machine learning techniques that only offer stateless intrusion detection. There are straightforward applications, such as that of Joshi and Phoha (2005), applying HMM to network based anomaly detection of TCP traffic. In contrast, Khanna and Liu (2006) employ a HMM to correlate events from several sources, including CPU activity, system call and process activity, and network and session activity. As mentioned previously, HMMs have been used to perform multi-stage intrusion detection (Ourston *et al.* 2003, Lee *et al.* 2008a), and Lee *et al.* (2008a) propose an agent based architecture with detection agents that employ HMMs.

Wright *et al.* (2004) demonstrate that HMMs can perform intrusion detection of encrypted traffic. They base the detection on packet size and arrival time of packets. The exact packet size is unavailable in encrypted traffic, but, as they state, "*payload sizes rounded to the next whole number multiple of the cipher's block size will be observed*". They simulate encrypted data based on the DARPA98 data set and data gathered at George Mason University (GMU). The results based on the DARPA98 data indicate that packet based classification is better than time based, leading to ~95% accuracy on *ftp*, *smtp* and *http* data. The detection rates of *smtp* was even 10% higher than a comparative study using a DT on the original data. However, the detection of *telnet* was poor; 21.7% based on packets and 14.2% based on time. Similar observations were made on the GMU data, however, obtaining lower classification rates. It is unclear whether this is due to an increased number of protocols in that data set.

As discussed above, one aspect of the IDS proposed by Hoang *et al.* (2009) is the significant speedup obtained with their new training algorithm. Additionally, they obtain significant performance improvements by incorporating HMMs into a hybrid IDS to perform application based anomaly detection. The HMM is an additional detection module to a database of unique normal sequences, extracted from a training phase. The outputs from this module are taken as inputs to a fuzzy inference module, describing frequency, probability of intrusion (from the HMM), and distance to known sequences (from the normal database).

Hoang *et al.* (2009) evaluate the hybrid approach on a synthetic *sendmail* data set (University of New Mexico 1996), and compare the results with using the normal database only, and a two layer scheme from their previous work (Hoang *et al.* 2003). In the two layer scheme, HMM is utilised in a second step to provide further analysis of *sendmail* traces if they do not exist in the database of normal traces, or if the frequency of occurrence of the traces is low. All approaches are reported with less than 0.3% false positives, and the hybrid system consistently obtains the lowest false positive rates of the three. More significantly, the hybrid system obtains approximately 10–20% higher detection rates than the two layer scheme on all test sets.

3.12 Population based search and optimisation techniques

The techniques referred to here are methods typically used to solve search and optimisation problems, such as Genetic Algorithms (GAs) (Holland 1992, Goldberg 1989), Genetic Programming (GP) (Koza 1992; 1994), Particle Swarm Optimisation (PSO) (Kennedy and Eberhart 1995, Banks *et al.* 2008a;b), and Ant Colony Optimisation (ACO) (Dorigo *et al.* 1999, Dorigo and Stutzle 2004). One of the strengths of these techniques is their parallel nature, and that their application is very diverse, provided that the problem can be quantified into some form of fitness measure, as discussed in Section 3.12.1. All the techniques discussed here naturally form part of hybrid systems, such as rule induction, as discussed in Section 3.12.2. However, they have also been used successfully as stand alone detectors, as discussed in Section 3.12.3. Section 3.12.4 considers Evolutionary Neural Network (ENN) applications, in which GAs and PSO have been used to optimise the weights and topology of ANNs. Section 3.12.5 discusses other applications.

3.12.1 Background and problem representation

GAs and PSO are commonly associated with the optimisation of continuous numerical functions, and ACO with combinatorial optimisation. Some of the benefits of adopting such techniques are flexibility in retraining, online/continuous learning and the potential for parallelism in the algorithms, which can be exploited both in the training and detection process. However, the challenge is to represent the intrusion detection problem in a form that can be processed and evaluated by these algorithms. For example, for evolving the weights of ANNs, each weight becomes a gene value of a chromosome or individual[8] (solution) in a population that is evolved (optimised) by a GA. A set of weights can be applied to an ANN and evaluated

[8] In GAs, a potential solution to a problem is referred to as a *chromosome* or *individual*. In PSO, it is referred to as a *particle*.

on a training or estimation data set, from which the mean of squared errors (MSE) is typically used as a *fitness measure*. The lower this value, the fitter the solution, which is what drives the evolutionary process. Refer to Section 5.2 on page 81 for more information on GAs and ENNs.

As mentioned above, the techniques considered here have also been applied to intrusion detection as detectors. For example, Balajinath and Raghavan (2001) have applied a GA to perform intrusion detection based on UNIX commands. First, they encode the commands with numeric values. This is done in an ascending order according to the frequency of use. They then use user behaviour entropy indices as a measure of the randomness of the command history of each user, represented as a 3-tuple $(match\,index, entropy\,index, newness\,index)$. The match index is *"a measure of regularity in user behaviour*", the entropy index *is "a measure of the distribution of commands in the command sample"*, and the newness index *is "a measure of the number of new commands which have not occurred earlier"*. A fitness function is defined that measures the user behaviour entrophy compared to average user behaviour entrophy. This is also used in the classification process, in which a difference between these entropy values would signify an intrusion if it exceeds a defined threshold.

GP was employed in this domain only a few years after the approach was first proposed. GP is an evolutionary algorithm with similar operators to GAs, but is different in that it evolves programs (generally as a tree structure) from a higher level description language. Crosbie and Spafford (1995) used GP to evolve autonomous agents for network based intrusion detection. One focus of their paper is on dealing with fitness evaluation and different feature information when evolving the trees. That is, network data contains numeric, binary and nominal information. When evolving the trees, it is important to prohibit certain combinations of operators, such as assigning a logic operator to an integer type (Crosbie and Spafford 1995). Their solution is to grow several trees, evolved with their own sets of primitives. Similarly to using MSE as a fitness measure when evolving ENNs, as discussed above, Crosbie and Spafford calculate the fitness as a measure of error in the prediction, but also penalise misclassifications.

As mentioned above, ACO is set apart from the other approaches, as it is primarily applied to combinatorial optimisation. There are recent ACO algorithms proposed for continuous numerical optimisation, such as Dréo and Siarry (2004) and Socha and Dorigo (2006), however, they have not been applied to this domain.

All the techniques discussed here have been applied to rule induction, as discussed in Section 3.12.2. ACO represents the problem as a graph and treats it as combinatorial optimisation. For example, He *et al.* (2007) proposes the following approach: "*In the problem graph... each node represents a condition that may be selected as part of the crisp rule antecedent being built by an ant. An ant goes round the graph selecting nodes according to a constraint satisfaction method, building its rule antecedent. The rule conclusion is assigned afterwards by a deterministic method*". They take an iterative approach to determine the final rule set. Once one run of the ACO algorithm is completed, the 'best' rule is added to a rule base. The instances in the training set covered by this rule are deleted, and the process is repeated until the number of uncovered instances in the training set is below a predefined threshold.

The fitness function used in these population based search/optimisation techniques is very important. General error (such as MSE) or performance based measures may suffice to obtain solutions with a acceptable quality. However, such measures may not be ideal. Moreover, one of the benefits of using these techniques is to be able to incorporate domain knowledge into a measure of what constitutes a good solution. For example, He *et al.* (2007) initially consider a general performance-based fitness function to evaluate the accuracy of a rule:

$$Fitness = \frac{TP}{TP+FN} \cdot \frac{TN}{FP+TN} \tag{3.1}$$

This function, however, gave poor performance in preliminary experiments. Hence, they propose a new fitness function that takes into account the coverage of a rule as well as its accuracy. Furthermore, the accuracy and coverage are managed by two additional parameters, *a* and *b*, as below. These parameters function as weights to adjust the balance between the two objectives.

$$Fitness = \left(\frac{TP}{TP+TN}\right)^{a} \cdot \left(\frac{TP}{TP+FP+TN+FN}\right)^{b} \tag{3.2}$$

3.12.2 Rule induction

It is over a decade ago that Sinclair *et al.* (1999) proposed using GAs and DTs to generate rules for an expert system. Although they did not validate their approach empirically, many researchers have since proposed several successful rule induction approaches. For example, Banković *et al.* (2007) use a GA to evolve rules for detecting network based intrusions, demonstrating success on *Probing* and *DoS* intrusions from the KDD Cup '99 data set. Lu and Traore (2003) adopt GP to evolve existing rules to be able to detect novel intrusions. They performed experiments with the DARPA99 data set, using data from the first day (10,000 instances) for evolving the rules, then testing with data from the second day. The test data consisted of 10,000 instances, containing two new types of intrusion, which they were able to detect.

In a recent application of GP for rule induction, Orfila *et al.* (2009) demonstrate two benefits of such an approach: small and simple rules may be evolved. The GP algorithm was constrained to trees of maximum 20 nodes and a depth of 6 levels. They adopt the first 5 traces from the LBNL/ICSI Enterprise Tracing data set from Lawrence Berkeley National Laboratory and ICSI (2005) to evaluate empirically their approach, and compare the results with a C4.5 DT. In one experiment, using trace #5 for training and testing on the remaining traces, they demonstrate the significant difference in tree size obtained with GP compared to a C4.5 DT. The DT produced a tree with 63 nodes, compared to 7 nodes with GP. The GP tree in this case obtained a 53.22% true positive rate (TPR) and a 4.1% false positive rate (FPR), compared with 42.31% and 2.1% with the DT. On average, based on four sets of experiments[9], GP obtains a TPR of 62.15% with 5.22% FPR, whilst the DT obtains a TPR of 48.82% with 2.18% FPR. Since there is a different trade-off in the performance of the two approaches, it is difficult to conclude which is better. However, from a data mining perspective, GP produces more comprehendible trees of rules.

Tsang *et al.* (2007) apply a multi-objective GA (MOGA) to evolve fuzzy rules, taking into account two objectives: accuracy and interpretability. They implement this rule generation in a Mobile Agent system, which uses a Fuzzy Set Agent (FSA) to generate fuzzy sets and offspring solutions based on this, and an Arbitrary Agent (AA) that adopts a MOGA to evaluate the FSA and its offspring. The AA also controls the population of FSAs, *i.e.*, FSAs with poor fitness values do not prosper. Although their focus is on obtaining interpretable rules, the performance results reported on the KDD Cup '99 data set are very good. They compare their findings with a C4.5 DT, NB, *k*-NN, and a SVM. Their approach outperformed all other techniques, although the difference in performance compared with the SVM is negligible.

As introduced in the previous section, He *et al.* (2007) applied ACO to rule induction for intrusion detection. Compared with a more established rule induction technique, RIPPER (Cohen 1995), ACO obtained nearly identical TPR. As reviewed in Section 3.2.2 researchers have given more attention to fuzzy rules, as they provide more flexible systems compared with crisp rules. Abadeh *et al.* (2007) demonstrate that an ACO based algorithm for fuzzy rule induction obtained significantly better results on the KDD Cup '99 data set compared with RIPPER. RIPPER obtained a 94.25% TPR with a 2.2% FPR, whilst the ACO based system obtained a 95.5% TPR with less than 0.01% FPR.

Other researchers have also focused on fuzzy rule induction. For example, Bridges and Vaughn (2000) and Florez *et al.* (2002) employ a GA to optimise fuzzy membership functions, as discussed in Section

[9] They excluded trace #3 from training due to its size (approximately 2.2 million instances).

3.2 on page 19. Another example is González *et al.* (2003) who expand on previous research of Dasgupta and González (2002), as discussed in Section 3.8.2 on page 35, by evolving *fuzzy* anomaly signatures. As an additional step to improving on the detection rates, Abadeh *et al*. (2006) incorporate a PSO in the evolutionary process of generating fuzzy rules. The PSO performs a local search around each individual in the population, which includes a new operator for modifying the rules. By including the PSO, the detection rates of *DoS* and *Probing* intrusions increased, although the false positive rate also increased. They randomly select a small set of instances for training (650) and testing (9600), thus, direct comparisons with other studies are not possible. Fries (2008) also report significantly improved results with fuzzy rule induction compared to other GA based approaches to intrusion detection. However, Fries provides few details concerning the experiments, and it is unclear what impact the feature selection that is performed has on the results.

3.12.3 Clustering and stand alone detection applications

In an early study applying GAs to intrusion detection, Balajinath and Raghavan (2001) emphasise the benefit of being able to continuously learn user behaviour, to keep track of user drift. Similarly to Balajinath and Raghavan, Neri (2000) adopts a distributed GA, REGAL (Giordana and Neri 1995), to determine patterns of normal network behaviour. As previously discussed in Section 3.10.3 on page 44, Leon *et al*. (2004a, 2004b) demonstrate the potential of GAs to perform network based anomaly detection by means of clustering, which they achieve by incorporating a niching mechanism.

Banković *et al.* (2008) have chosen a GA over other clustering algorithms, to obtain more robustness, reduce the problem of 'getting stuck' in local optima and to exploit the parallel nature of the algorithm. They utilize the clustering potential of the GA to perform unsupervised, network based, anomaly detection. The approach does not require a predefined number of clusters, such as the popular *k*-means algorithm, and new data that is introduced to the cluster model does not need to be assigned to existing clusters; instead, new clusters may be created, thus, giving more flexibility. In a different application, by Lin and Wang (2008), a GA is hybridised with *k*-means clustering, which allows for the value of *k* to be optimised.

There are several applications of ACO based clustering to intrusion detection. Ramos and Abraham (2004) apply an unsupervised ant clustering model, referred to as ACLUSTER, to network based intrusion detection. They argue that it is a desirable approach in this domain as the parallel and distributed nature of the ant model offers real time online training, and there is no need for complete retraining. The same benefits are argued by Feng *et al.* (2006), who propose ACO clustering as a part of an agent system. Other benefits of their system include that it facilitates unsupervised and supervised learning, and that it is self organising. Feng *et al.* (2007) later propose a new ACO based clustering system, hybridised with a SOM for network based anomaly detection.

Tsang and Kwong (2005b) improve on an existing ant clustering model by Lumer and Faieta (1994), to better deal with high dimensional data. They adopt the KDD Cup '99 data to evaluate the performance of their ant clustering model, and compare with *k*-means clustering, SOM, another ant based clustering technique, and a multiple classifier (from other studies). Their algorithm obtained the highest detection rates on *R2L* and *DoS*, and second best for *U2R*, *Probing* and *Normal*. The ACO clustering system proposed by Feng *et al.* (2006), as mentioned above, was also validated on a small subset of the KDD Cup '99 data set, and outperformed a DT, SVM, LGP (Linear Genetic Programming), and *k*-NN.

There are several studies that demonstrate the success of GP for intrusion detection. Abraham *et al*. (2007) and Hansen *et al*. (2007) have both obtained high detection rates on the KDD Cup '99 data set. However, both studies used small subsets of the data, which prevents direct comparisons with other studies adopting the full data set. Abraham *et al*. (2007) examined three types of GP algorithms: Linear Genetic Programming (LGP), Multi-Expression Programming (MEP), and Gene Expression Programming

(GEP). They found that the different algorithms obtained better detection rates on certain classes. For example, MEP obtained the highest detection rates on *U2R* and *R2L*, whilst LGP detected *Probing* and *DoS* intrusions with higher accuracy. More interestingly, as few as 2–7 features (of the original 41 features) were sufficient to detect certain intrusions with MEP.

3.12.4 Evolutionary neural networks

Michailidis *et al.* (2008) employ PSO to optimise the weights of MLPs. The principle is similar to that of using GAs to evolve the weights, as discussed in Section 5.2.3 on page 83. Michailidis *et al.* evaluate the approach on a small (undersampled) subset of the 10% KDD Cup '99 data set. Their findings are comparable to other applications of MLPs, as discussed in Section 3.6.2.2 on page 30. Their approach detects more *Probing* and *U2R* attacks, but at a higher FPR (approximately 3%). However, it is unclear whether this can be attributed to the use of PSO to train the MLPs, or due to the undersampling.

Tian and Gao (2009) propose a hybrid training algorithm comprised of a self adaptive GA[10] and backpropagation for anomaly detection. Their main motivations are to speed up the training process and prevent the backpropagation training algorithm from 'getting stuck' in local optima. The GA continuously evolves the initial weights of MLPs, which are then trained with backpropagation, using the MSE of these MLPs as a fitness measure. If none of these MLPs have obtained a sufficiently low MSE, the process is repeated. The authors do not, however, report on intrusion detection performance, the only MSE, which reaches lower values than with backpropagation alone.

Similar to studies discussed in Section 3.6.2.3 on page 31, Han and Cho (2006) consider the application of ANNs to application based anomaly detection; more specifically, learning system call sequences. Instead of adopting a fixed network structure, which has been common for applications with MLPs (Ryan *et al.* 1998, Ghosh and Schwartzbard 1999, Ghosh *et al.* 1999) and Elman networks (Ghosh *et al.* 2000), they optimise this with a GA. The GA evolves the number of neurons (maximum 15) and how they are connected. Since the structure is more flexible, neurons in the input layer may be directly connected to neurons in the output layer. They also adopt local search algorithms and the backpropagation algorithm to train the networks during the continuous evolutionary process of the GA.

The ANNs of Han and Cho (2006) were applied to analyse sequences of 10 commands. Therefore, the ANNs have 10 input neurons. One ANN is evolved for each program, which can process 45 common BSM system call events (out of 280). They employ the DARPA99 data set to evaluate their approach, in which training is conducted on normal data from weeks 1 and 3, and testing on data from weeks 4 and 5. Compared with the previous studies with MLPs and Elman networks (Ryan *et al.* 1998, Ghosh and Schwartzbard 1999, Ghosh *et al.* 1999; 2000), they obtain better detection rates. Moreover, they conclude that the training process is quicker compared to the systematic approaches adopted in the other studies to determine the ideal network structure.

3.12.5 Other applications

Most applications of the population based techniques have been covered in the previous sections, however, there are a few hybrid systems that are mentioned here. For example, Tsang and Kwong (2005a) incorporated ACO into an agent based system, which consists of six different agents. First, a monitor agent gathers data and performs feature extraction. The data is then passed on to a decision agent, which performs unsupervised anomaly detection based on ant clustering. The remaining agents deal with action, coordination of higher level information, communication between the user and the coordination agents, and, finally, an agent monitors the entire agent system. Banerjee *et al.* (2005) propose using 'emotional' ants as sensors

[10] The self adaptive GA evolves parameter values of the genetic operators during the optimisation process.

to detect intrusions in the early stages. These ant agents monitor each other, as well as the network, for intrusion. To avoid ants visiting repeatedly nodes in the network, they propose using a tabu list[11] which prohibits ants from revisiting nodes for a certain time. However, they do not report any empirical findings to demonstrate that this works in practice.

There are several applications of GAs to feature selection, *e.g.*, Fries (2008), Helmer *et al.* (2002), Stein *et al.* (2005), Tsang *et al.* (2007). In addition to the general benefits of feature selection, as discussed previously in Section 2.5 on page 16, Tsang *et al.* (2007) use feature selection to help their rule induction algorithm obtain more comprehendible rules for human operators to analyse.

Finally, as previously discussed in Section 3.8.4 on page 37, GAs have been adopted in AIS to generate detectors (Dasgupta and González 2002, González and Dasgupta 2002, Haag *et al.* 2007, Ostaszewski *et al.* 2006, Powers and He 2008).

3.13 Summary

Rule Based Systems (RBSs) are commonly used in commercial Intrusion Detection Systems (IDSs), and are better established than the other Artificial Intelligence (AI) techniques reviewed here. RBSs are well suited for event correlation to perform misuse detection. However, other techniques are better suited for anomaly detection, such as statistical methods and clustering.

The ability to facilitate anomaly detection is one of the benefits that has motivated much research on machine learning for intrusion detection. In the last decade, an increasing amount of research on machine learning for misuse detection can also be observed in this review. The application of techniques such as Artificial Neural Networks (ANNs) to misuse detection offers some desirable flexibility in the detection process compared with conventional RBSs, *i.e.*, variations of learned attacks can be detected. The inflexibility of RBSs, due to operating with 'crisp' rules, has been considered one of their main drawbacks (Esmaili *et al.* 1996, Gürer *et al.* 1996, Jakobson *et al.* 2004, Lewis 1993, Owens and Levary 2006). However, this observation is no longer entirely accurate, since researchers have proposed several applications of fuzzy RBSs, which have also been shown to be capable of performing anomaly detection (Dickerson and Dickerson 2000, Dickerson *et al.* 2001, Owens and Levary 2006, Tillapart *et al.* 2002).

A general benefit of machine learning is that knowledge engineering/extraction/acquisition is avoidable. Furthermore, for techniques that are not 'black boxes', such as Decision Trees (DTs) and rule mining/induction (*e.g.*, RIPPER (Cohen 1995)), knowledge of novel intrusions may be extracted. These techniques can be employed as detectors in an IDS, or the extracted rules may be incorporated in a RBS. Rule induction approaches are typically used for misuse detection, mining rules of intrusions. Fuzzy association rule mining has been found to successfully determine patterns for anomaly detection (Bridges and Vaughn 2000, Florez *et al.* 2002, Tajbakhsh *et al.* 2009). Optimisation techniques such as Genetic Algorithms (GAs), Genetic Programming (GP) and Particle Swarm Optimisation (PSO) have also been applied successfully to rule induction. Recent studies using such techniques focus on more pragmatic problems of obtaining rules that are small and easy to interpret (Orfila *et al.* 2009, Tsang *et al.* 2007).

Hybridisation of techniques has become commonplace in IDSs, which allows researchers and practitioners to exploit the benefits of individual techniques and approaches. For example, IDSs that employ one technique to perform misuse detection, such as a RBS, and another to perform anomaly detection, such as a statistical method, ANN or clustering (Aydin *et al.* 2009, Goss *et al.* 2007, Yuan and Guanzhong 2007). In such hybrid systems, it is necessary to adopt a decision module that is able to correlate the alerts from the individual detectors and determine whether an alert should be issued. For this, RBSs and Bayesian

[11] Although Banerjee *et al.* (2005) do not discuss the similarity of their tabu list with tabu search, these are related. For information on tabu search, refer to Glover (1989; 1990).

networks have been successfully applied (Depren *et al.* 2005, Kruegel *et al.* 2003).

Much research on machine learning treats the intrusion detection problem as a classification task, adopting techniques such as DTs, ANNs and naïve Bayes (NB). Combining classifiers is a popular approach to improving the accuracy of an individual classifier (Bauer and Kohavi 1999, Brown *et al.* 2005, Dietterich 2000a, Tan and Gilbert 2003). Furthermore, classifier ensembles can provide additional security from an adversary (Biggio *et al.* 2008; 2009). Popular classifier combination approaches such as AdaBoost and Random Forests (RFs) have been applied to intrusion detection (Panda and Patra 2009, Zhang and Zulkernine 2006), as well as more specialised combinations based on observations in the literature that different classifiers perform well on different classes of intrusion (Anuar *et al.* 2008, Gharibian and Ghorbani 2007, Pan *et al.* 2003, Peddabachigari *et al.* 2007, Sabhnani and Serpen 2003). However, as identified in this review, there are discrepancies in the results reported in the literature, which makes it difficult to determine which classifiers are indeed best suited for detecting different classes of intrusion. These discrepancies are investigated further in the following chapter.

CHAPTER 4

Discrepancies in findings reported in the literature

The KDD Cup '99 data set (The UCI KDD Archive 1999) was created for the Knowledge Discovery and Data Mining Tools competition and associated conference in 1999 (Elkan 2000), and has since been used extensively to validate prototypes of network based intrusion detection systems. However, as observed in the previous chapter, there are discrepancies in the findings reported in the literature with this data set, which are investigated in this chapter.

The investigation into the discrepancies uncovered another issue related to the KDD Cup '99 data set, and the data from which is has been derived. It was created by transforming the raw *tcpdump* of the DARPA98/99 data (Lippmann *et al.* 2000a;b) to a set of features considered suitable for machine learning techniques (Lee and Stolfo 2000). Shortly after the DARPA98 data set was created, with the purpose of facilitating the evaluation of existing IDSs, McHugh (2000) published a critique of the evaluation project. Other researchers have published further criticisms of both the DARPA and KDD Cup '99 data sets (Bouzida and Cuppens 2006a;b, Brugger 2007a, Mahoney and Chan 2003, Sabhnani and Serpen 2004), which has led Brugger (2007b) to claim that the data sets are fundamentally flawed and that findings based solely on this data are invalid.

Most of the existing criticisms are directed at the DARPA data (Brugger 2007a;b, Mahoney and Chan 2003, McHugh 2000), but Brugger (2007b) generalises to the KDD Cup '99 data set as well, which is not entirely appropriate. However, according to some studies (Bouzida and Cuppens 2006a;b, Sabhnani and Serpen 2004), the KDD Cup '99 data set exhibits 'issues' that are not present in the DARPA data. The word 'issues' is used carefully here since the existing studies have not provided sufficient analysis and discussion to conclude that these *methodological factors* are indeed issues of the data set. Several methodological factors have been identified in this thesis to significalty affect the results. Furthermore, this thesis considers whether these are indeed problems with the data set, or whether they are simply challenges of intrusion detection, for which common machine learning approaches may be unfruitful.

McHugh (2000) hoped that by criticising the current efforts and discussing issues openly, this would aid similar initiatives in the future. However, a decade later, no such initiative has been undertaken. Although Brugger (2007b) discourages researchers from using the KDD Cup '99 data set, he does acknowledge that researchers continue to use it due to a lack of better publicly available alternatives. The review in the previous chapter demonstrates this clearly. Therefore, it is important to address the criticisms and methodological issues to determine whether the KDD Cup '99 data set can be used in future work to provide useful findings.

Section 4.1 includes a more specific discussion of the discrepancies in the findings reported in the literature. Section 4.2 discusses the main criticisms from the literature, directed at both the DARPA and KDD Cup '99 data sets. The focus of this work, however, is on the KDD Cup '99 data set. Details of this data set are provided in Appendix A on page 196. Section 4.3 details the aims of the empirical investigation and the research method. Results are presented in Section 4.4, followed by a discussion in Section 4.5. Section 4.6 concludes with a discussion of the implications of the findings obtained here, addressing future use of the KDD Cup '99 data set.

4.1 Contradictory results

There are three publicly available subsets of the KDD Cup '99 data set, a (full) training set, a 10% version of this training set, and a test set. Some researchers adopt the full training set and test set, as in the competition, whilst others adopt the 10% training set and the test set. However, some adopt only the training set, or even smaller subsets.

One can expect that using different subsets of the data will lead to different results, particularly since the original test set includes 17 new attacks. However, not only are the results different, they are in some cases contradictory. For example, Pan *et al.* (2003) claim that Artificial Neural Networks (ANNs) are unable to detect *U2R* and *R2L* intrusions, whilst Mukkamala and Sung (2003) report comparatively high detection rates (48% *U2R* and 95% *R2L*). Similarly, for Decision Trees (DTs), Sabhnani and Serpen (2004) report high detection of *U2R* (99.18%) and *R2L* (99.18%), whilst Benferhat and Tabia (2005) report very poor detection, 10.09% and 0.56% respectively. Such discrepancies are an issue since recent research on machine learning for intrusion detection attempts to combine classifiers based on observations that different classifiers perform well on different classes of intrusion (Anuar *et al.* 2008, Gharibian and Ghorbani 2007, Pan *et al.* 2003, Peddabachigari *et al.* 2007, Sabhnani and Serpen 2003, Xiang *et al.* 2008). Due to these observations, it is not possible to conclude that one technique is better suited for a particular class of intrusion. However, there are great methodological differences between these studies, which are uncovered when delving deeper into the results they present. For example, Pan *et al.* only included one type of intrusion for *U2R* and *R2L* each, whilst Mukkamala and Sung employed a very small set of data (5092 instances for training and 6890 for testing).

An analysis of the existing findings indicates that the choice of data subset is the main cause of the discrepancies. The subsets used can be classified as one of the following:

1. Selecting only a few types of intrusions.
2. Compiling a new, small, version of the data set.
3. Using the original training set only.
4. Using the original training set and test set.
5. Merging the training and test sets to create a new data set.
6. Filtering data to fit assumptions about the distribution of normal and intrusive data.

The classification above is largely self explanatory, but to avoid ambiguity, it should be noted that studies using the training set only (#3) do employ validation methods such as holdout or cross validation. Approach #6 is not within the scope of this paper, as it has only been observed in studies using clustering techniques to perform unsupervised anomaly detection (Eskin *et al.* 2002, Leung and Leckie 2005, Portnoy *et al.* 2001).

Table 4.1: Overview of the detection rates of *U2R* and *R2L* intrusions for ANN, DT and NB classifiers, as reported in the literature.

Study	Technique	U2R %	R2L %	Data subset
Pan *et al.* (2003)	ANN	0	0	#1
Bosin *et al.* (2005)	NB	52.40	94	#2
Chebrolu *et al.* (2005)	DT	48	90.58	#2
Mukkamala and Sung (2003)	ANN	48	95	#2
Peddabachigari *et al.* (2007)	DT	68	84.19	#2
Depren *et al.* (2005)	Hybrid	80	98.02	#3
Ben Amor *et al.* (2004)	DT	7.89	0.52	#4
	NB	11.84	7.11	
Benferhat and Tabia (2005)	DT	10.09	0.56	#4
	NB	11.40	8.66	
Bouzida and Cuppens (2006a;b)	DT	7.02	2.58	#4
	ANN	0	~27	
Sabhnani and Serpen (2003)	DT	1.80	4.60	#4
	ANN	13.20	5.60	
Shafi *et al.* (2009)	DT	33.33	0.31	#4
	ANN	44.00	35.34	
	NB	60.00	8.89	
Sabhnani and Serpen (2004)	DT	87.50	99.18	#5
	ANN	89.28	99.44	

However, it can be noted that these studies adopted the training set only, which was then filtered to meet their assumptions regarding the proportion of intrusion compared to normal data.

Much research has been undertaken in this domain with ANNs (Haykin 1998), DTs and naïve Bayes (NB), which allows for a comparison of results across most of the subsets used in the literature. Although the methods in the literature are not identical, clear trends in results can be derived from the results, as seen in Table 4.1. This table presents results on *U2R* and *R2L* intrusions, which best demonstrates the problem. The detection rates for *Normal*, *Probing* and *DoS* intrusions are generally similar on all subsets.

With reference to Table 4.1, the following discussion will exclude the first study since no class performance can be compared, although the authors make such a generalisation themselves. The small data subset (#2) has led to higher true positive rates compared with most of the other subsets; 48–68% *U2R* and 84.19–95% *R2L*. Unfortunately, the selection process used to create this data set is not transparent, which prevents further analysis. Depren *et al.* (2005) do not offer classification rates for the DT used in their study, but results of their hybrid system are included to demonstrate the classification rates possible on the training set alone (#3). The results obtained on the official test set (#4) are similar for *U2R* (0–13.20%), but significantly lower for *R2L* (0.52–27%). The reasons for this performance decrease are discussed in the next section. Finally, merging the training and test sets (#5) to produce a new data set leads to the highest detection rates of all subsets; 87.50–89.28% *U2R* and 99.18–99.44% *R2L*. Merging the data sets changes the intrusion detection challenge significantly since there are no longer new attacks in the test set, making it similar to adopting only the training set.

4.2 Criticisms

Both the KDD Cup '99 and DARPA data sets have been criticised by several researchers. Although the KDD Cup '99 data stems from the DARPA data, not all criticisms apply to both data sets. However, the transformation of the DARPA data has introduced further issues. Key points from the literature are discussed below.

4.2.1 Not representative of real network traffic

McHugh (2000) argues that the data in the DARPA data sets is not representative of data in a real network; the structure of the simulated network would not allow for particular intrusions to be executed properly and the distribution of intrusions are unrealistic. First, the proportion of intrusions is very large compared to the normal traffic. Second, McHugh argues that one would observe a significantly higher proportion of Probing instances compared with other intrusions. This applies to both the DARPA and KDD Cup '99 data sets. Consequently, McHugh argues that if systems are tested and tuned according to the DARPA (or KDD Cup '99), they may perform poorly in a 'real' intrusion detection scenario.

With respect to the proportion of intrusion compared with normal traffic, some researchers have found it necessary filter out data to make it fit their definition of what 'real' network data looks like. Examples of this can be seen in applications of clustering to anomaly detection (Eskin *et al.* 2002, Leung and Leckie 2005, Portnoy *et al.* 2001), which assume that intrusions are in the minority, so that they can be detected as outliers.

4.2.2 TTL values

Mahoney and Chan (2003) discovered that only intrusions had 'Time To Live' (TTL) values of 126 and 253, whilst the majority of normal data had TTL values of 127 and 254. Consequently, it would be possible to perform intrusion detection based on one feature. Based on those findings, Brugger (2007b) encourages that papers based solely on the DARPA data set should not be accepted for publication. However, Brugger extends this statement to the KDD Cup '99 data set, which is inappropriate since it does not contain this information in its feature set.

4.2.3 Training and test sets

Sabhnani and Serpen (2004) investigated why machine learning algorithms achieve poor detection of *U2R* and *R2L* intrusions in the KDD Cup '99 data set. Their conclusion is that the training and test sets are simply too different for machine learning algorithms to be successful. The underlying reason(s) for this is unclear, although their findings indicate that it is not due to the new attacks in the test set. They suggest merging the training and test sets to produce a new data set, which gives significantly better results. However, they do not discuss whether this is appropriate.

4.2.4 R2L intrusions

Another issue related only to the KDD Cup '99 data, as pointed out by Bouzida and Cuppens (2006b), is that many *snmpgetattack* (*R2L* intrusion) instances are identical to normal instances. Consequently, this causes misclassifications between these two classes. Bouzida (2006) argues that this is caused by poor transformation of the *tcpdump* data from the DARPA data set. However, it is unclear whether it is possible to separate these attack instances from normal traffic with alternative transformation functions. Irreducible error due to overlapping classes is a feature of many practical machine learning problems (Gadaras *et al.*

2007, Liu 2008, Raudys 2002), so although resulting in misclassification, it is not necessarily indicative of a defective data set, but of an unsolvable aspect of the problem given the information available to the machine learning algorithms.

4.3 Method

The aims of this empirical investigation are presented in Section 4.3.1. One of the aims is to determine methodological factors that may affect the results, which are discussed in Section 4.3.2. Specifications of the classifiers adopted for the experiments are provided in Section 4.3.3. Details of the data subsets and metrics that are adopted are provided in Sections 4.3.4 and 4.3.5 respectively. Section 4.3.6 provides an overview of the empirical investigation.

4.3.1 Aims of the investigation

There are three mains of this empirical investigation, as specified below.

Aim 1.1: Determine causes of discrepancies in the literature

The 'review' of the published work, in Section 4.1, suggest that the discrepancies may be primarily caused by researchers adopting different subsets of the KDD Cup '99 data set. It is not surprising that researchers that adopt small subsets obtain different results from those that adopt the original training and test sets, since the latter includes 17 new attacks. However, as indicated by Sabhnani and Serpen (2004), the new attacks are not the main cause of the poor performance on *U2R* and *R2L* when the test set is adopted. Therefore, the underlying factors that affect the results are not known.

There are two facets to this aim. The first is to uncover potential methodological factors that affect the results, which is considered in the subsequent section, dealing with the following issues observed in the literature:

1. Data subsets.
2. Validation method.
3. New attacks in test set.
4. *R2L* instances:
    (a) *snmpgetattack* instances identical to *Normal* instances.
    (b) attacks not present in the test set.
5. Class imbalance.
6. Duplicates.

The second facet of this aim is to empirically demonstrate how these methodological factors affect the results.

Aim 1.2: Provide a benchmark to assist in interpreting the findings reported in the literature

The results reported in Table 4.1 on page 56 do not yield a comprehensive insight into the findings. Since there are methodological differences in the studies, such as data preprocessing (scaling, normalisation, removal of duplicates, feature selection, *etc.*), classifier configurations, and choice of classification approach (binary, multi-class, *etc.*), a comprehensive and objective analysis of the results is not possible. Therefore, benchmark results, produced using well understood machine learning classifiers and following a repeatable method, are needed, in order to facilitate a better interpretation of the current body of research.

Aim 1.3: Determine whether the KDD Cup '99 data set is useful to current and future research

The data set is outdated, in terms of the intrusions that are present, and it has been subject to criticism, as discussed in Section 4.2 on page 57. Nevertheless, researchers continue to use it. Therefore, it is important to determine if and how it can still be used to provide valid and useful findings in future research. In addressing this aim, the usefulness of the data to both the intrusion detection and machine learning communities must be considered.

4.3.2 Methodological factors

This section outlines methodological factors that are considered in this study. Additional factors that are not included here are highlighted in Section 4.3.2.5.

4.3.2.1 Validation and taxonomy

The studies that do not adopt the original training and test sets may take a different approach to validation, such as cross validation. This has been done, for example by Sabhnani and Serpen (2004), as they merge the training and test sets to form a new data set. Whilst cross validation allows for more extensive use of the data for training and testing, the results obtained with holdout validation may be significantly different as it will be more sensitive to the selection of training and test data. Two particular factors in this case may affect the results: (1) the magnitude of duplicates, which is discussed further, below, and (2) the taxonomy that is adopted, which may affect the minor classes in particular.

Six attacks in the data set have less than 10 instances each, being as low as 2 in the training set. Even when grouping the attacks, there are only 52 instances of *U2R* attacks in the official training set (whether the full training set or the 10% version) and 70 in the test set. In this context, it is important to bear in mind that the attacks in each of the intrusion classes (in the taxonomy of Kendall (1999)) may not be similar in terms of the feature values, as discussed by McHugh (2000). Hence, if data is sampled randomly, the majority of instances in the training set may be of significantly different attacks than those in the test set.

4.3.2.2 Difference between the training and test sets

The empirical investigation that Sabhnani and Serpen (2004) have conducted, which claims that the training and test sets are unacceptably different, is constrained. First, they only perform binary classification; *U2R* versus *non-U2R* instances, and *R2L* versus *non-R2L* instances. It is not clear whether their findings can be generalised to all classes, and how these would compare with other studies conducting multi-class classification. Therefore, further analysis as a multi-class classification problem is desirable. Second, there is no investigation of the data set itself, to indicate why the training and test sets are too different for more successful classification. Their experiments indicate that the difference is not due to the new intrusions in the test set, though they do not examine the effects of removing them. Third, they do not discuss whether merging the data sets is actually appropriate, despite conducting such a significant modification to the classification problem. Although significantly better detection of *U2R* and *R2L* is achieved, this may not be a realistic reflection of the *intrusion detection* challenge.

4.3.2.3 Classification of the R2L class

The classification of *R2L* intrusions is likely to be poor for two reasons, when the original training and test sets are used:

1. There are *Normal* instances that are identical to *R2L* intrusions.
2. Some *R2L* attacks are not represented in the test set, namely *spy* and *warezclient*.

As pointed out by Bouzida and Cuppens (2006b), many *snmpgetattack* instances are identical to *Normal* instances. Statistics gathered in this study reveal that there are 8,054 *Normal* instances identical to *snmpgetattack* instances; 7,273 of which are in the test set, the remaining in the training set (10% version). There are also 21 *Normal* instances that are identical to *DoS* attacks; 12 *ping of death* and 9 *teardrop*. Misclassifications are therefore inevitable, which still remains an issue when the training and test sets are merged.

There are only 2 instances of *spy* attacks, but, more significantly, there are 1,020 instances of *warezclient*, which is 90.59% of all *R2L* instances in the training set. Furthermore, there is a large increase in *R2L* intrusions in the test set; from 1126 to 16347, and nearly doubling the number of attack types from 8 to 14. Therefore, regardless of the introduction of new attacks in the test set, detection of *R2L* is likely to be poor.

4.3.2.4 Duplicates and class imbalance

Due to the lack of temporal information in the KDD Cup '99 data set, there are many duplicate instances in the data set. Duplicates may have a negative effect on on the training process of machine learning techniques, because they affect the quality of the data. Haykin (1998, p. 179) and LeCun (1998) discuss the importance of high quality training data (in the context of neural networks trained with backpropagation), and argue that the training samples should be diverse. Duplicates compromise this diversity, but the effects of this are not indicated by Haykin or LeCun. They are, however, examined by Kolcz *et al.* (2003), who investigate empirically the effects of duplicates on naïve Bayes and a Perceptron with Margins, in an application to spam detection. They observe that the amount of duplication has a negative effect on the accuracy of the classifiers, and argue that duplicates should be removed.

The *DoS* and *Probing* classes exhibit the most duplicates, due to the nature of the intrusions. Table 4.2 provides an overview of the number of instances of each class before and after removing duplicates, which illustrates the significant changes to the class balance.

Table 4.2: Number of instances of each class in the 10% training set and test set before and after removing duplicates. Class proportion is given in brackets.

	Training		Test	
	Before	After	Before	After
Normal	97,278 (19.69%)	87,832 (60.33%)	60,593 (19.48%)	47,913 (62.00%)
Probing	4,107 (0.83%)	2,131 (1.46%)	4,166 (1.34%)	2,682 (3.47%)
DoS	391,458 (79.24%)	54,572 (37.48%)	229,853 (73.90%)	23,568 (30.49%)
U2R	52 (0.01%)	52 (0.03%)	70 (0.02%)	70 (0.09%)
R2L	1,126 (0.23%)	999 (0.69%)	16,347 (5.26%)	3,058 (3.96%)

Class imbalance has not been considered previously in studies on intrusion detection. However, according to the general literature on learning from imbalanced data, common machine learning techniques such as ANNs and DTs are known to be biased towards the major class(es) (Chawla 2003, Jo and Japkowicz 2004). As a consequence, if the imbalance is extreme, the minor class(es) may simply be 'ignored'. Therefore, if the duplicates are removed, which causes the class balance to change significantly, the results are likely to be different.

Another factor to consider is that duplicates may also lead to somewhat deceptive results during testing, since the ability to detect one instance will be multiplied according to the number of duplicates. If an instance with many duplicates in the test set is not classified correctly, this can cause a misleadingly poor classification rate on that particular class. The converse is also true. It follows that two classifiers that obtain a small difference in the number of *unique* instances correctly classified, may obtain significantly different results.

4.3.2.5 Issues not considered here

There are methodological factors that are not considered here due to the scope of this work. For example, although Bouzida (2006) argues that the transformation of the *tcpdump* data in the DARPA data set is flawed, alternative approaches are not examined here. The focus is on the use of the existing KDD Cup '99 data set.

Different studies have adopted different classification approaches, such as binary classification (normal/intrusion), the five class classification used in the DARPA evaluation and KDD Cup '99 competition, or individual attacks. Only the five class classification is used here, as it is the most widely used. However, in some experiments, an analysis of individual attacks is conducted to confirm the underlying causes of particular observations. For example, the classification of new attacks in the test set.

There are different ways of preprocessing the data, such as how to deal with nominal features, scaling/normalisation of feature values, and feature selection. Some classifiers can deal directly with nominal features, whilst others can not. Such preprocessing is not required for the classifiers adopted here, however, and feature selection is not performed.

4.3.3 Classifiers

Much of the previous research that has motivated this study include Decision Trees (DTs), naïve Bayes (NB) and Artificial Neural Networks (ANNs). However, ANNs are not adopted here since the training process is too time consuming for the extensive experiments conducted for this investigation, given the constraints on time and computing resources. NB is a probabilistic, frequency based technique, and the literature suggest that NB does not have as strong a bias towards the major classes as DTs and ANNs (Ben Amor *et al.* 2004, Kolcz *et al.* 2003, Panda and Patra 2007). Therefore, NB is likely to produce different classification behaviour/performance. Both the DT and NB algorithms are employed from Weka v. 3.4 (Witten and Frank 2005, Anon 2006), which supports replication of these experiments.

The aim of this study is not to develop new techniques, which might outperform those adopted here; they were chosen because they are widely used and are well understood, making them ideal for the purpose of this investigation. The specifications of the classifiers are provided in their respective sections below.

4.3.3.1 Decision tree

A J.48 DT is used, which is a Java implementation of the C4.5 algorithm (Quinlan 1993). This is the most commonly used DT in this domain. Both pruning and reduced error pruning (REP) have been examined, investigating confidence factors of {0.05, 0.25, 0.50} for the former. Reduced error pruning can prevent over fitting of the data (Mitchell 1997, pp. 69–71), and is employed here with 5 folds, which leaves 80% of the training set for building the tree, and 20% for pruning.

Other settings include: minimum of 2 instances per leaf node, using sub tree raising, but not employing binary splits or LaPlace for smoothing counts at leaf nodes. These settings were determined during preliminary experiments.

4.3.3.2 Naïve Bayes

The NB algorithm in Weka can use one of two configuration options for dealing with numeric parameters, namely kernel estimation or supervised discretisation. Both options were investigated and it was observed that no processing of numeric parameters and kernel estimation led to significantly worse performance than supervised discretisation (in terms of true positive rates). Kernel estimation led to similar classification rates as the DT, performing well on the major classes, but not the minor classes. No processing of numeric parameters simply led to an overall decrease in classification rates. Therefore, results with supervised discretisation are reported here.

4.3.4 Data set and validation

All subsets of the KDD Cup '99 data set are adopted here, which are publicly available at the UCI KDD Archive (1999). However, due to the memory requirements of the full training set (requires more than 3GB RAM when Weka is used), only a version of the data set with no duplicates could be used. The 10% training set and test set were merged to produce a new version of the data set, as in (Sabhnani and Serpen 2004). The small subset used in some studies (Bosin *et al.* 2005, Chebrolu *et al.* 2005, Mukkamala and Sung 2003, Peddabachigari *et al.* 2007, Sung and Mukkamala 2003) is not adopted since there is not enough information in the literature to recreate it.

At the time this investigation started, there was one bugged line in the training set (a Normal instance with too many feature values), which was deleted. This instance has been removed in the data set that is currently available to download. The detection is performed by the five classes presented in Section 2.2 on page 8, with more details on specific attacks in Appendix A on page 196.

Two validation methods are adopted in this study: holdout and cross validation. Holdout validation was used in the KDD competition, in which the data was split into a training and test set. In the experiments conducted on the training set alone, or on the merged data, both holdout and cross validation is used. According to the findings of Kearns (1996), for holdout validation, the data is partitioned so that 80% of the instances are used for training and the remaining for testing. The selection of data for each partition is performed chronologically[1] based on individual attacks whilst ensuring that attacks with few instances, even as low as two, will receive at least one instance in the test partition. Ten folds is chosen for cross validation, which is supported by an empirical investigation with DTs and NB by Kohavi (1995), and is considered common by Depren *et al*. (2005).

4.3.5 Metrics

Several metrics are adopted in this study to provide multiple perspectives on performance. Confusion matrices present information about predicted and actual classifications. An example of a confusion matrix can be seen in Table 4.3. Rows (predicted) show what the instances of each respective class have been classified as. Take the first row, for example, 60,262 *Normal* instances were classified as *Normal*, whilst 243 *Normal* instances were classified as *Probing*, *etc*. The final column, %correct, indicates that 99.50% of *Normal* instances were correctly classified (as *Normal*). These classification rates are used in performance overviews presented throughout the thesis. Columns (actual) show everything classified as that respective

[1] The selection is performed chronologically so that these experiments can be repeated in other, independent, studies.

class. For example, 14,527 instances of *R2L* attacks were classified as *Normal*. The last row, %correct, indicates that 74.60% of all *Normal* classifications were actually *Normal*.

Table 4.3: Confusion matrix for the winning entry of the KDD Cup '99 competition (Pfahringer 2000).

Actual\Predicted	Normal	Probing	DoS	U2R	R2L	%correct
Normal	60,262	243	78	4	6	99.50
Probing	511	3,471	184	0	0	83.30
DoS	5,299	1,328	223,262	0	0	97.10
U2R	168	20	0	30	10	13.20
R2L	14,527	294	0	8	1,360	8.40
%correct	74.60	64.80	99.9	71.40	98.8	

From the confusion matrices, accuracy, true positive and false positive rates are calculated, as defined in Section 2.1 on page 7. These provide a numerical measure of how well the classifiers perform overall, whilst the confusion matrices allow the behaviour to be analysed. The different metrics can be biased and can give very different impressions of the performance. For example, accuracy does not give any information about behaviour of the classifiers and they can give the impression of excellent detection rates provided the major class(es) are detected well. This is an issue here due to the significant class imbalance in the data set. For example, if *U2R* intrusions are not detected at all, it will have an insignificant impact on the accuracy (and the TPR).

Although Receiver Operator Curves (ROCs) are popularly used in the literature, to give a measure of performance of IDSs in terms of trade-off between TPR and FPR, they are not used here as they do not add any means of better understanding the results. For the same reason, the cost matrix used in the KDD Cup '99 competition, to give a single performance metric, is not adopted here.

4.3.6 Outline of the empirical investigation

Many experiments are conducted in this study to demonstrate empirically the impact of the methodological factors, as discussed above in Section 4.3.2 on page 59. The experiments can be summarised as follows:

General observations: exploring classifiers performance and validation methods.

Performance on original data sets: providing a benchmark that is used to compare the results obtained from the subsequent experiments.

Removing new attacks from the test set: this is undertaken to determine whether these are the sole reason for the performance difference when adopting the training and test sets, compared with only adopting the training set or merging the training and test sets. There is a particular focus on the *R2L* class for this set of experiments.

Removing normal instances identical to intrusions: this is conducted on the training and test sets, as well as the merged data set. As above, there is a focus on the *R2L* class.

Removing duplicates: this is conducted on all data sets, with an emphasis on how it affects the class balance.

The purpose of this study is to investigate empirically the factors that may affect the results, but this does not imply that they are considered flaws of the data at this point. Some of the factors have indeed been addressed as flaws of the data set in other studies, as previously discussed in Section 4.2 on page 57. However, some factors may be considered practical challenges of intrusion detection, rather than flaws, which is discussed further in Section 4.6.

4.4 Results

General observations are discussed in Section 4.4.1, followed by results on the original data subsets (no preprocessing) in Section 4.4.2. Section 4.4.3 discusses results obtained when removing new attacks from the test set. Sections 4.4.4 and 4.4.5 discuss results obtained on the different data subsets when removing duplicates and *Normal* instances identical to intrusions.

The following abbreviations are used in the tables of results:

- Data set: Tr = Training set only; Tr&T = training and test sets; M = merged data set.
- Data processing: [o] refers to original data; [d] no duplicate instances; [n] no *Normal* instances identical to intrusions.

4.4.1 General observations

This section discusses observations of algorithm behaviour and the results obtained with the two validation methods considered in this study. According to these observations, the results with reduced error pruning and holdout validation are excluded from consecutive sections as they detract from the focus of the investigation.

4.4.1.1 Algorithms

Of the two classifiers adopted in this study, NB obtains higher classification rates on the minor classes than the DT, which was expected. The trade-off is a lower classification rate on the major classes, *Normal* and *DoS*, which coincides with other observations in the literature (Ben Amor *et al.* 2004, Panda and Patra 2007). Despite these differences, both techniques obtain similar changes in classification rates when different data subsets are employed.

The results obtained with the DT using reduced error pruning (REP) differ from those using confidence factors (CF). Since REP prunes according to a fold (partition) of the training set, the success depends on how accurate an estimate this fold is of the data in the test set. Considering this in the context of the findings of Sabhnani and Serpen (2004), demonstrating how machine learning techniques fail to detect *U2R* and *R2L* intrusions due to significant differences between the training and test sets, it is not unexpected that the performance is worse with REP.

For all experiments, the accuracy and true positives are worse with REP. Generally, the classification rates on *U2R* and *R2L* are lower with REP, whilst the differences in the detection of the other classes are negligible. The one exception to this observation is an increased classification rate with REP on *Probing* and *U2R* when the original training and test sets are used. Further analysis of the data would be necessary to shed more light on these observations. Nevertheless, it can be concluded that REP has an overall detrimental effect on the results. An overview of the differences in classification rates obtained with REP compared with CF is given in Table 4.4.

Table 4.4: Difference in classification rates of DTs when REP is used compared with CF. A positive value signifies improved performance with REP compared with CF.

Data set	Accuracy%	TPR%	FPR%	Normal%	Probing%	DoS%	U2R%	R2L%
Tr [o]	−0.45	−0.01	+0.04	−0.08	−0.22	0	−5.77	−1.16
Tr [d]	−0.04	−0.06	+0.02	−0.02	−0.33	−0.02	−19.23	−1.81
Tr&T [o]	−0.06	−0.02	+0.07	−0.07	+9.22	−0.06	+5.71	−2.46
Tr&T [d]	−1.85	−2.63	−0.22	+0.22	−0.79	−2.51	−5.72	−0.60
Tr&T [n]	−0.37	−0.42	+0.05	−0.05	+0.48	−0.09	−4.29	−5.66
M [o]	−0.02	−0.06	−0.05	+0.05	−0.45	0	−0.82	−11.85
M [d]	−0.03	−0.11	−0.01	+0.01	−0.23	0	−4.92	−1.53
M [n]	−0.03	−0.02	+0.03	−0.03	−0.40	0	−11.48	−0.41

4.4.1.2 Validation methods

The results obtained with the two validation methods, holdout and cross validation, are generally significantly different. There are negligible differences in the true positive rates (TPR), though holdout validation appears to give a more negative representation of the detection rates on the minor classes; *U2R* in particular. With holdout validation, there are merely 10 *U2R* instances in the test partition when the training set is used, which is a two fold issue. On the one hand, poor detection of this class will have an insignificant impact on the accuracy and TPR. On the other hand, the actual detection rate of *U2R* is very sensitive to number of instances correctly classified. Thus, the DT, detecting one *U2R* intrusion leads to 10% *U2R* detection, which is merely 0.20% less TPR than when cross validation is used (even though cross validation leads to 46.15% *U2R* intrusions detected). Holdout validation on the training set leads to unpredictable results in some cases too; for example, when duplicates are removed. This has no impact on the FPR for the DT (compared with when duplicates are present), but this leads to an increase of ~15% in the FPR for NB. In addition, *DoS* detection decreases to ~10%, but is mainly misclassified as *Probing*. Similarly, the majority of false positives are detected as *Probing*. An overview of the classification rates obtained on the training set with holdout and cross validation is provided in Table 4.5.

Table 4.5: Comparison of classification rates obtained with holdout and cross validation (CV) on the training set.

Tech., val. method	Accuracy%	TPR%	FPR%	Normal%	Probing%	DoS%	U2R%	R2L%
DT, holdout	99.61	99.78	0.24	99.76	99.51	99.63	10	80.97
DT, CV	99.97	99.98	0.04	99.96	99.37	99.99	46.15	96.36
NB, holdout	77.83	99.78	3.19	96.09	99.76	73	80	98.23
NB, CV	98.71	99.83	1.78	98.22	98.47	98.84	80.77	96.71

As will be discussed further in subsequent sections, the experiments on the merged data set are more stable and yield more predictable and understandable results. Similarly to the results on the training set, the TPR remains nearly identical regardless of validation approach. However, the FPR varies significantly. On the original data, the FPR is higher for both the DT and NB with holdout validation, ~5% and ~11% respectively. However, this difference is reduced when duplicates and *Normal* instances identical to intrusions are removed. In this case, for NB, the FPR is even ~2.5% lower than when cross validation is used.

Depren *et al.* (2005) state that cross validation is preferred when there is limited data available. From the range of experiments conducted, the results obtained with holdout validation appear deceptive in some cases compared with the results obtained with cross validation seeming to be very sensitive to the class balance and duplicates. This also needs to be considered in context of the discussion in Section 4.3.2.1 on page 59, that the individual attacks within the four classes of intrusion may be significantly different in terms of their feature values. For example, when one split is selected, the training partition may contain many duplicates of certain attacks, whilst the test partition may have a high proportion of other attacks that are significantly different to those in the training partition. Then, in experiments where duplicates are removed, the results will be biased by another training / test split. The effects of this are particularly prominent for the minor classes, especially *U2R*, as observed in this study.

Generally, then, it is considered here that cross validation provides a more reliable evaluation, and, where possible, only cross validation results are reported in the following sections. Results obtained with holdout validation are, however, provided in Appendix B.1 on page 199. Note that, due to memory requirements, the run-time for cross validation on the full training set is infeasible. Therefore, for the full training set, holdout validation is used.

4.4.1.3 Full training set

Due to the magnitude of data in the full training set, it was only possible to adopt a version with no duplicate instances. Moreover, due to the memory requirements, the run time for cross validation became infeasible. Therefore, results are reported only for holdout validation.

In most aspects, the results obtained with the full and 10% versions of the training set are similar. However, whereas the 10% training set led to a significant decrease in *DoS* classification rates when duplicates are removed, they remain high when the full training set is used (holdout validation). The FPR is also lower. Interestingly, the DT is able to detect 60% more *U2R* attacks (70% compared with 10%) with the full training set, compared with the 10% version, achieving the same detection rate as NB. An overview of the results is provided in Table 4.6.

Table 4.6: Overview of results obtained with the full and 10% training sets, using holdout validation.

Tech. - data	Accuracy%	TPR%	FPR%	Normal%	Probing%	DoS%	U2R%	R2L%
NB - 10% [o]	77.83	99.78	3.91	96.09	99.76	73.00	80.00	98.23
NB - 10% [d]	52.89	99.43	19.13	80.87	100	5.16	80.00	98.01
NB - Full [d]	95.29	99.91	5.60	94.40	99.82	97.93	70.00	97.01
DT - 10% [o]	99.61	99.78	0.24	99.76	99.51	99.63	10.00	80.97
DT - 10% [d]	65.96	99.70	0.31	99.69	99.53	9.85	10.00	97.01
DT - Full [d]	99.94	99.93	0.04	99.96	98.34	99.98	70.00	96.02

An overview of results when the original test set is employed is provided in Table 4.7. Contrary to the results obtained with holdout validation on the training set, the results obtained with the DT deteriorate when the full training set is used (compared with the 10% version); approximately 10% less *U2R* and 5% less *R2L* intrusions are detected. The choice of training set has an insignificant effect on NB except for the TPR, which is nearly 10% higher with the full training set when duplicates are removed. This is due to more *DoS* intrusions being misclassified as *Probing* instead of being classified as *Normal*.

Table 4.7: Overview of results obtained with the full and 10% training sets, using the original test set.

Tech. - data	Accuracy%	TPR%	FPR%	Normal%	Probing%	DoS%	U2R%	R2L%
NB - 10% [o]	91.14	91.76	1.10	98.90	83.61	94.98	64.29	10.44
NB - 10% [d]	39.48	78.59	1.02	98.98	84.13	25.04	64.29	10.42
NB - Full [d]	39.35	87.95	1.21	98.79	86.68	24.90	60.00	9.93
DT - 10% [o]	92.58	91.16	0.51	99.49	75.61	97.27	12.86	5.79
DT - 10% [d]	92.48	91.24	0.77	99.23	79.16	97.03	21.43	7.30
DT - Full [d]	92.18	91.03	0.55	99.45	80.44	96.87	10.00	2.71

4.4.2 Performance on original data

This section presents the results obtained on the data subsets without any preprocessing, such as removing duplicates or *Normal* instances identical to intrusions. An overview of the classification rates is provided in Table 4.8.

Table 4.8: Overview of results of the DT and NB on the different data subsets.

Data - Tech.	Accuracy%	TPR%	FPR%	Normal%	Probing%	DoS%	U2R%	R2L%
Tr - NB	98.71	99.83	1.78	98.22	98.47	98.84	80.77	96.71
Tr - DT	99.97	99.98	0.04	99.96	99.37	99.99	46.15	96.36
Tr&T - NB	91.14	91.76	1.10	98.90	83.61	94.98	64.29	10.44
Tr&T - DT	92.58	91.16	0.51	99.49	75.61	97.27	12.86	5.79
M -NB	95.37	99.03	14.17	85.83	98.79	97.72	83.61	96.11
M - DT	99.16	99.53	2.19	97.81	98.96	99.99	53.28	93.05

The TPR and individual classification rates on the intrusion classes are very similar for the training and merged data sets. Furthermore, the classification rates on the intrusion classes are significantly higher on these data sets compared to the original test set; particularly on *Probing*, *U2R* and *R2L*. However, there is a significant difference in the FPR obtained on the training and merged data sets. The FPR is higher on the merged data set, and, consequently, the TNR (correct classification of *Normal*) is lower. This has been found to be related to the *R2L* class.

An analysis of confusion matrices reveal that the higher FPR on the merged data set is caused by misclassifications of *R2L* and *Normal* instances. As discussed previously in Section 4.3.2.3 on page 60, this is expected due to *snmpgetattack* instances identical to *Normal* instances. On the test set, *R2L* detection is poor, and the majority of the misclassifications are as *Normal*, which is shown in Table 4.9. However, on the merged data set, classification behaviour is changed, giving high detection of *R2L*, but with a significant proportion of *Normal* instances misclassified as *R2L*. See Table 4.10 for a confusion matrix for NB on the merged data set, illustrating this.

Table 4.9: Confusion matrix for the DT on the training and test set.

Actual\Predicted	Normal	Probing	DoS	U2R	R2L	%correct
Normal	60,285	218	75	2	13	99.49
Probing	888	3,112	166	0	0	74.7
DoS	6,253	11	223,588	0	1	97.27
U2R	55	0	0	9	6	12.86
R2L	**14,940**	344	3	114	946	5.79
%correct	73.14	84.45	99.89	7.2	97.93	

Table 4.10: Confusion matrix for NB on the merged data set.

Actual\Predicted	Normal	Probing	DoS	U2R	R2L	%correct
Normal	135,507	2,376	19	824	**19,145**	85.83
Probing	65	8,173	13	17	5	98.79
DoS	5,889	4,706	607,163	3237	316	97.72
U2R	18	0	0	102	2	83.61
R2L	303	148	0	228	16,794	96.11
%correct	95.57	53.06	99.99	2.31	46.31	

The decreased detection rates on the original test set cannot be explained by the *snmpgetattack* instances, as with the merged data set. The performance on *Probing* and *U2R* is also significantly worse. For both the DT and NB, the majority of misclassifications are false negatives, as illustrated in Table 4.9, above. This is influenced by the additional attacks in the test set. However, as will be demonstrated in Section 4.4.3, removing the new attacks from the test set still leads to poor classification rates on *U2R* and *R2L*. Another factor is that *R2L* validation is arguably very poor when using the original training and test sets, in which the test set does not contain the intrusion (*warezclient*) that makes up 90.59% of the *R2L* intrusions in the training set. In the merged data set, this is not an issue, since the data has been sampled proportionally for each attack. The following sections discuss further these factors.

4.4.3 Removing new attacks from the test set

It is clear that the new attacks pose a greater challenge for the machine learning techniques, but are not the sole reason for the low detection rates of *U2R* and *R2L* when the original training and test sets are used. As seen in Table 4.11, removing the new attacks leads to an overall increase in detection rates except for *U2R*. *Probing* and *DoS* are now detected with nearly 100% accuracy and there is a significant increase in *R2L* detection. However, *U2R* and *R2L* detection remains very low compared with the results obtained with the merged data set (which was more than 50% *U2R* and 90% *R2L*).

Table 4.11: Overview of results on the training and test set, before and after removing new intrusions from the test set.

Tech. [data]	Accuracy%	TPR%	FPR%	Normal%	Probing%	DoS%	U2R%	R2L%
NB [o]	91.14	91.76	1.10	98.90	83.61	94.98	64.29	10.44
NB [n]	96.57	98.13	1.10	98.90	99.66	97.74	64.10	28.40
DT [o]	92.58	91.16	0.51	99.49	75.61	97.27	12.86	5.79
DT [n]	98.13	97.98	0.51	99.49	98.65	99.98	10.26	15.77

The high detection rates of *Probing* obtained here suggest that the decreased performance on the original training and test sets is because the new attacks are too different to the existing *Probing* intrusions in the training set. An additional experiment was conducted to verify this, by performing classification according to the individual attacks in order to determine which attacks are being misclassified. Some differences in the results are to be expected, since the classification approach is changed, but they are similar, as seen in Table 4.12.

Table 4.12: Overview of predicted *Probing* classifications for the DT on the original training and test sets.

Classification approach	Normal	Probing	DoS	U2R	R2L	%correct
Five class	888	3,112	166	0	0	74.70
Individual attacks	656	3,283	227	0	0	78.80

Of the two new *Probing* intrusions, *mscan* and *saint*, it is the former that is responsible for the majority of misclassifications (822 out of 883). The *saint* intrusions are generally classified as *satan* and *ipsweep* (other *Probing* intrusions), whilst *mscan* is predominantly misclassified as *Normal* and *neptune* (*DoS*). An analysis of the feature values for the *Probing* intrusions suggests some underlying causes of these misclassifications. Most strikingly, of the services that *mscan* targets, no other *Probing* attacks target *imap4*, but a large proportion of *neptune* attacks do. Further, of the other services *mscan* targets, very few of the other *Probing* attacks target the same; only one instance of *satan* targets *pm dump*, five *telnet* and *pop3*, and three *sunrpc*. Compared with the poor classification rates of *mscan*, all other *Probing* attacks contributed with high detection rates, 98.04% *ipsweep*, 100% *nmap*, 91.24% *portsweep*, 99.45% *satan*, and 97.96% *saint*.

These observations support the criticism and potential dangers of adopting the taxonomy employed in the DARPA evaluation (McHugh 2000), as discussed in Section 4.3.2.1 on page 59. The *saint* intrusion is evidently similar to the existing *Probing* attacks in the training set (particularly *satan*), but *mscan* is not.

4.4.4 Removing normal instances identical to intrusions

Removing the *Normal* instances that are identical to intrusions does not have a significant impact on the classification rates on the test set. However, the difference is more clear on the merged data. The detection rate of *Normal* instances has increased and the FPR has decreased due to not misclassifying as many *Normal* instances as *R2L*, which coincides with expectations. Refer to Table 4.13 for an overview of these results.

Table 4.13: Results obtained on the merged data set with [o] and without [n] *Normal* instances identical to intrusions.

Tech. [data]	Accuracy%	TPR%	FPR%	Normal%	Probing%	DoS%	U2R%	R2L%
NB [o]	95.37	99.03	14.17	85.83	98.79	97.72	83.61	96.11
NB [n]	96.33	99.04	9.59	90.41	98.83	97.72	81.97	95.75
DT [o]	99.16	99.53	2.19	97.81	98.96	99.99	53.28	93.05
DT [n]	99.93	99.96	0.17	99.83	99.03	99.99	61.48	98.99

4.4.5 Removing duplicates

There is generally a negligible difference in the results obtained when duplicates are removed from the training set. The most significant difference is the detection of *DoS* intrusions for NB, decreasing from 98.84% to 95.50%. As discussed in the previous section, the results differ significantly depending on the validation method used. It is worth mentioning that the detection of *DoS* intrusions decreased from 99.63% to 9.85% for the DT using holdout validation. However, the majority of *DoS* intrusions were detected as *Probing*, and, thus, the true positive and false negative rates remained similar. Similarly, the most significant difference when the test set is adopted is the detection rate on *DoS* for NB, detecting only 25.04%. The confusion matrix in Table 4.14 shows that the majority of misclassifications of *DoS* intrusions are as *Probing*, which is why the TPR remains high. However, it does also lead to a large number of false negatives.

Table 4.14: Confusion matrix of NB on the training and test sets, when duplicates were removed from the training set.

Actual\Predicted	Normal	Probing	DoS	U2R	R2L	%correct
Normal	59,972	363	7	67	184	98.98
Probing	396	3,505	125	112	28	84.13
DoS	**38,817**	**132,078**	57,561	1,288	109	25.04
U2R	16	0	0	45	9	64.29
R2L	14,391	141	0	112	1,703	10.42
%correct	52.8	2.58	99.77	2.77	83.77	

The results obtained with the DT are more positive, leading to increased detection of the minor classes, *Probing*, *U2R* and *R2L*, as seen in the overview in Table 4.15. The class imbalance is not as great when removing duplicates, which is arguably why the DT is now better able to detect the minor classes, whilst NB, being less biased towards the major classes, maintains an almost identical detection rate on these classes.

Table 4.15: Classification rates on the test set when training on the original training set [o] compared to removing duplicates [d] (from the training set only).

Tech. [data]	Accuracy%	TPR%	FPR%	Normal%	Probing%	DoS%	U2R%	R2L%
NB [o]	91.14	91.76	1.10	98.90	83.61	94.98	64.29	10.44
NB [d]	39.48	78.59	1.02	98.98	84.13	25.04	64.29	10.42
DT [o]	92.58	91.16	0.51	99.49	75.61	97.27	12.86	5.79
DT [d]	92.48	91.24	0.77	99.23	79.16	97.03	21.43	7.30

Overall, the effects of removing duplicates from the merged data set are more predictable and positive for both techniques. The greatest change in the results is a reduction in misclassifications of *Normal* and *R2L*, leading to significantly lower FPRs and increased detection of *Normal* instances. The main contributor to this improvement is that most of the *snmpgetattack* instances identical to *Normal* instances are removed since they are duplicates; only 86 instances remain. The class imbalance is also reduced, which, as discussed above, is a likely cause of better detection of the minor class *U2R* for the DT. An overview of the results on the merged data set is presented in Table 4.16. The results with NB are positive in the sense that true negative rate (correct classification of *Normal*) is higher and the FPR is significantly lower. However, the detection rates on *DoS*, *U2R* and *R2L* have decreased.

Table 4.16: Overview of classification rates obtained with [o] and without [d] duplicates in the merged data set.

Tech. [data]	Accuracy%	TPR%	FPR%	Normal%	Probing%	DoS%	U2R%	R2L%
NB [o]	95.37	99.03	14.17	85.83	98.79	97.72	83.61	96.11
NB [d]	94.45	97.47	4.59	95.41	98.52	92.67	76.23	89.50
DT [o]	99.16	99.53	2.19	97.81	98.96	99.99	53.28	93.05
DT [d]	99.68	99.56	0.18	99.82	97.97	99.94	61.48	93.86

Experiments on the merged data set has the advantage of having duplicates removed from both the training and test data. Further experiments on the training and test sets, removing duplicates from the test partition also, led to significantly higher classification rates compared with removing duplicates only from the training set. For example, NB, which achieved 25.04% correct detection of *DoS* intrusions, now detects 84.41%. Further, *R2L* detection increases by approximately 20% for both algorithms.

4.5 Summary and discussion

The first and primary aim of this investigation was to establish the underlying causes of the discrepancies in the findings reported in the literature. The empirical findings obtained here demonstrate clearly that there is a significant impact on results depending on the choice of data subset/partition, processing such as removing duplicate instances, and validation method. This goes some way towards explaining the often substantial differences and contradictions observed across the current body of empirical work. The importance of these findings is two fold: first, the perils of comparing results and findings across different studies are evident; second, it is clear that there is a need for deeper consideration of methodological factors in empirical studies.

The second aim of this study was to provide an empirical benchmark that can be used to help interpret the findings of studies that adopt different data subsets. The findings here have demonstrated that it is imperative to know the details of the data and processing adopted. Otherwise, the classifiers may be attributed credit for something that is actually due to manipulating the data. An overview of classification rates on the three subsets examined here is presented in Table 4.17. The results presented here on the training and merged data sets are with cross validation.

Table 4.17: Overview of results of the Decision Tree (DT) and naïve Bayes (NB) on the three subsets adopted in this study.

Tech. - Data	Accuracy%	TPR%	FPR%	Normal%	Probing%	DoS%	U2R%	R2L%
DT - Tr	99.97	99.98	0.04	99.96	99.37	99.99	46.15	96.36
DT - Tr&T	92.58	91.16	0.51	99.49	75.61	97.27	12.86	5.79
DT - M	99.16	99.53	2.19	97.81	98.96	99.99	53.28	93.05
NB - Tr	98.71	99.83	1.78	98.22	98.47	98.84	80.77	96.71
NB - Tr&T	91.14	91.76	1.10	98.90	83.61	94.98	64.29	10.44
NB - M	95.37	99.03	14.17	85.83	98.79	97.72	83.61	96.11

Results obtained on the original test set are bound to be worse for machine learning techniques applied to *misuse* detection, compared with results that can be obtained on the training set alone, or by merging the training and test sets. This is due to the new attacks in the test set and the selected training data for *R2L*; *i.e.*, not testing the *R2L* intrusion that makes up 90.59% of the training set (*warezclient*), whilst also doubling the number of *R2L* attacks in the test set. Therefore, the performance on this class remains poor even when the new attacks are removed from the test set. Detection of *Probing* is approximately 20% lower on the test set, compared with the other subsets, which is due to very poor detection of one of the new attacks, *mscan*. This attack is accountable for the majority of misclassifications (822 out of 883 for the DT). The feature values of these attacks are not similar to the other *Probing* attacks, with particular reference to the services that are targeted.

It is inevitable that misclassifications will occur between *Normal* and *R2L* when the original test set is adopted, due to *snmpgetattack* instances being identical to *Normal* instances. This remains a factor that affects misclassifications on the merged data set. According to the findings obtained here, the misclassifications are mainly in the form of false positives on the merged data set, giving higher detection of *R2L*. On the original test set, the misclassifications appear primarily as false negatives.

One form of preprocessing the data that was found to have a significant impact on the findings (generally) is removing duplicate instances. This has been done by, for example, Sabhnani and Serpen (2004) on the merged data set. First, removing duplicates improves the general data quality, since the instances become more diverse (LeCun *et al.* 1998, Haykin 1998, p. 179). Second, removing duplicates also removes most of the *snmpgetattack* instances that are identical to *Normal* instances. Consequently, misclassifications between *Normal* and *R2L* are reduced significantly. Third, the class balance changes significantly. As the classes in the data set become more balanced, the DT is better able to detect the minor classes. This observation is in accordance with existing literature on class imbalance (Chawla 2003, Jo and Japkowicz 2004). The probabilistic, frequency based, NB technique, however, is not as affected by the class imbalance. This observation is also in accordance with existing literature indicating that NB is more robust in this respect (Ben Amor *et al.* 2004, Kolcz *et al.* 2003, Panda and Patra 2007). An overview of the classification rates obtained with the DT on the merged data set is provided in Table 4.18, which shows how the classification rates improve when duplicates and *Normal* instances identical to intrusions are removed.

Table 4.18: Classification rates obtained with the DT on the merged data set when duplicates [d] and *Normal* instances identical to intrusions [n] are removed, compared with original data [o].

Data	Accuracy%	TPR%	FPR%	Normal%	Probing%	DoS%	U2R%	R2L%
[o]	99.16	99.53	2.19	97.81	98.96	99.99	53.28	93.05
[d]	99.68	99.56	0.18	99.82	97.97	99.94	61.48	93.86
[n]	99.93	99.96	0.17	99.83	99.03	99.99	61.48	98.99

From the discussion and results presented above, it is more clear why Sabhnani and Serpen (2004) are able to report significantly better findings than any other study in the literature. Furthermore, they perform cross validation on the merged data set, whilst most other studies have performed holdout validation. According to the empirical findings obtained here, the results with cross validation appear more positive than those obtained with holdout validation. This was found to be due to holdout validation being sensitive to the training/test split, whilst cross validation is more robust since it provides an average from partitions (folds). This split is influenced by duplicates and, consequently, also the class balance, which impact on the minor classes; in particular *U2R* due to the relatively few instances of this class.

Depren *et al.* (2005) also perform cross validation, stating that this is preferred when there is limited data available, which the empirical findings obtained here support. However, cross validation may not be an appropriate validation method for intrusion detection since it is then not possible to explicitly evaluate the detection of new attacks. This is a factor that needs to be considered for each study, according to the aims of the investigation.

4.6 Implications

The implications of the empirical findings are considered in Section 4.6.1, followed by a discussion of future use of the KDD Cup '99 data set in Section 4.6.2.

4.6.1 Considerations of methodological factors

Although it was not possible to recreate the small subset employed in several studies (Bosin *et al.* 2005, Chebrolu *et al.* 2005, Mukkamala and Sung 2003, Peddabachigari *et al.* 2007), due to a lack of available information, the findings in this study provide some indication as to why the reported *U2R* and *R2L* performance on this subset is higher than on the original training and test sets (and the training set alone). First, the magnitude of data is reduced significantly, whilst number of *U2R* instances remains (Peddabachigari *et al.* 2007), which implies that the class imbalance is not as extreme. Hence, according to the findings in this thesis and elsewhere in the literature (Chawla 2003, Jo and Japkowicz 2004), detection of the minor classes (*Probing*, *U2R* and *R2L*) is likely to be higher for classifiers such as DTs and ANNs. Second, with particular reference to *R2L* detection, the high detection rates may be influenced by avoiding the issues with *R2L* evaluation in the original training and test sets, *i.e.*, not testing the *warezclient* attack, and *snmpgetattack* instances being identical to *Normal* instances. Third, duplicates may have been removed, which would increase the quality of the data.

Merging the data set may be desirable for evaluating machine learning techniques for *misuse detection*, however, it is not a straightforward solution. First, one element of testing is lost: detecting new intrusions (discussed further below). Second, due to the negative effects of duplicates and *Normal* instances identical to intrusions, it is beneficial for the performance of machine learning techniques to remove these. However, to do this, it is necessary to establish that certain underlying assumptions hold.

With respect to *Normal* instances identical to intrusions, these were removed in some experiments in this study to demonstrate empirically the impact this has on the results. The occurrence of identical instances could be a result of several factors, such as flaws in the data collection for the DARPA evaluation, limitations in *tcpdump* (not enough audit information to separate certain intrusions from normal behaviour), mislabelling, limitations in the transformation of the DARPA *tcpdump* to the KDD Cup '99 data, or some *snmpgetattacks* may simply be identical to normal traffic regardless of the other potential factors. Bouzida and Cuppens (2006a, 2006) assert that it is due to limitations in the transformation of the DARPA data. According to this, one can justify the removal of these instances, as it would be avoidable when gathering and preprocessing data for applied misuse detection. However, they do not demonstrate alternative transformation functions that are capable of avoiding this problem.

Duplicates are intrinsic to the intrusion detection problem, due to the nature of some intrusions (mainly *DoS* and *Probing*). Because of the negative impact of duplicates on the training process, these would arguably be removed from the training data. However, it is not straightforward to determine whether duplicates should be removed from the test set or not. On the one hand, the existence of duplicates is more representative of real traffic. On the other hand, duplicates in the test set can skew the results, and complicate their interpretation. For example, if an attack with many duplicates is detected correctly, this leads to high detection rates. If an attack with many duplicates is misclassified, this leads to a poor detection rate. Such classification behaviour can result in misleading findings. Therefore, if duplicates remain in the test set, it is important that this is a realistic representation of the data in the network environment the IDS is deployed in. This is, however, one of the main criticisms of McHugh (2000), as discussed in Section 4.2 on page 57.

The decision to merge the data sets should not be motivated by simply obtaining higher classification rates. As with the issues of removing duplicates, the purpose of the investigation should ultimately determine which subset of the data to use. For certain studies, using the original training and test sets may be appropriate. The results in this study (and in the cited literature) indicate that current misuse detection approaches will not be able to detect a significant proportion of intrusions in the test set. It is, therefore, inappropriate to rely on the generalisation ability of a classifier (trained for misuse detection) to detect new attacks, particularly when adopting the popular five class taxonomy[2]. Anomaly detection or hybrid approaches may, on the other hand, detect new attacks, and when exploring such techniques, it is essential that novel attacks are present in the test data.

Compared with using single machine learning classifiers, Xiang *et al.* (2008) propose a hierarchical hybrid system that appears more successful, as seen in Table 4.19. At the first level, a DT will classify an instance as *DoS*, *Probing* or 'others'. At the next level, 'others' are further refined to either 'normal' or 'intrusion' by an AutoClass (AC) classifier (Bayesian clustering) (Cheeseman and Stutz 1996). Finally, the DT will refine 'intrusion' further to either *U2R* or *R2L*. When a classification has been made, according to the five-class taxonomy of Kendall (1999), each class is further refined into the specific attacks.

Table 4.19: Results of the hierarchical hybrid approach of Xiang *et al.* (2008) compared with the DT in this study (on the original test set).

	FPR%	Normal%	Probing%	DoS%	U2R%	R2L%
Hybrid	3.2	96.80	93.40	98.66	71.43	46.97
DT	0.77	99.23	79.16	97.03	21.43	7.30

Xiang *et al.* (2008) also sampled the training data to reduce the class imbalance; 10,000 random *Normal* and all *U2R* and *R2L* instances for AC, and an undersampled subset for the DT, with only 3,000 *Normal* instances, less than 10,000 *Probing* and *DoS* intrusions, and all *U2R* and *R2L* instances. The classification rates on the original test set are significantly higher than those reported here, however, at the expense of a higher FPR, as seen in Table 4.19. Xiang *et al.* (2008) notes that all *snmpgetattack* instances are misclassified as *Normal*, which is the main reason that *R2L* detection is not higher. This implies that the increase in FPR is not caused by this attack.

4.6.2 Using the KDD Cup '99 data set in future research

With respect to the criticisms discussed in Section 4.2 on page 57, it is important to question whether the KDD Cup '99 data is still useful for research into machine learning and/or intrusion detection. Ultimately, as Brugger (2007b) acknowledges, researchers continue to use this data due to a lack of better publicly available alternatives.

With respect to the DARPA data, the issue with TTL values (Mahoney and Chan 2003) is indeed a serious flaw, as it is possible to perform effective intrusion detection based on one feature, which would not be possible in practice. This is one of the main reasons why the DARPA data set (and by association the KDD Cup '99 data set) are now not valued for publication of research findings by some researchers in the intrusion detection community. However, the issue of TTL values does not apply to the KDD Cup

[2] One of the criticisms of McHugh (2000) is that the taxonomy is not appropriate, since attacks are not grouped into the four intrusion classes according to similarity between their feature values. Some of the findings here support this, demonstrating how a new *Probing* attack in the test set, *mscan*, is misclassified because it has different feature values to the other *Probing* attacks.

’99 data, and, therefore, this generalisation is inappropriate. Nevertheless, the criticism of McHugh (2000) and Brugger (2007b), regarding the data not being representative of ‘real’ network traffic, is important to address.

Many papers discuss network based intrusion detection in a generic way, whilst the traffic in different types of networks is arguably significantly different. Hence, satisfying a requirement for a general network based data set to be representative of real environments from a machine learning / data mining point of view is not possible. Research on machine learning applied to intrusion detection implies that the techniques will be trained on data gathered from the specific environment they are deployed in. Therefore, if the data set poses some generic, real, challenges for machine learning approaches when applied to intrusion detection, there is still value in it. It is simply in the interpretation of the findings of such research that caution must be observed, in evaluating the similarity of these challenges to those posed by any particular environment in which the techniques may be considered for deployment.

Similarly to other authors (Brugger 2007b, McHugh 2000), it is not recommended here that the KDD Cup ’99 data set is simply used as an evaluation benchmark. However, it can be used to address some generic challenges for machine learning applied to intrusion detection, such as:

- Related to the learning process:
    - Dealing with high dimensional data (curse of dimensionality (Bellman 1961, Duda 2001); memory requirements).
    - Learning from a large data set (learning speed).
    - Learning from imbalanced data.
- Feature selection (data reduction).
- Incremental/continuous learning.
- Detecting new intrusions.

Most of the challenges above are self explanatory, and have been discussed to some degree in this thesis. For example, learning from imbalanced data is a challenge that has been shown here to be linked to poor detection on some classes of intrusion. Learning from imbalanced data is a challenge of equal interest to many other domains, such as medicine (Cohen *et al.* 2006, Mazurowski *et al.* 2008, Mena and Gonzalez 2006), credit scoring (Huang *et al.* 2006), customer churn (Burez and van den Poel 2009, Xie *et al.* 2009), natural language processing (Kobyliński and Przepiórkowski 2008), lexical acquisition (Kermanidis *et al.* 2004) and text recognition (Stamatatos 2008). The findings obtained here provide strong indications of the effect that class balance has on classifiers such as the DT. This is further investigated in Chapter 5, which concretely demonstrates that the intrinsic class imbalance in intrusion detection causes poor true positive rates for some machine learning techniques that are biased towards the major class(es). Therefore, it is necessary to develop and apply appropriate training techniques and/or investigate methods of sampling the training data.

Feature selection provides benefits for both the training and classification processes; it effectively reduces the amount of data required to process, reduces the dimensionality of the problem (which can be beneficial to some classifiers), and saves memory. Several machine learning techniques have been applied to this task, see, for example, Chebrolu *et al.* (2005), Mukkamala and Sung (2002), Ramos and Abraham (2004) and Sung and Mukkamala (2003).

One particular challenge of interest in this domain is adapting over time. Some techniques require complete retraining as an offline process once more data is gathered. Therefore, incremental/continuous learning is a desirable trait as the data changes over time due to new applications and new types of intrusion.

Several researchers propose methods that allow for this, for example, Cannady (2000) proposes an IDS with an adaptive ANN, a Cerebellar Model Articulation Controller (CMAC) (Albus 1975), which is capable of learning new intrusive behaviour during run time. Other studies include, for example, Kim and Bentley (2002), using Artificial Immune Systems, and Ramos and Abraham (2004), using Ant Colony Optimisation.

Evaluating whether new attacks can be detected is of great interest to intrusion detection. If the merged data set is used for this purpose, several approaches for misuse detection may be taken to test the detection of new intrusions, such as: (1) the approach taken in the KDD Cup '99 competition, splitting the data into a training and test partition, in which the test partition should contain several new intrusions of each class of intrusion; (2) train on all but one type of intrusion and test only the detection of the intrusion type excluded from the training. The latter approach is more laborious, but avoids some complications. If the taxonomy of Kendall (1999) is adopted for classification, the selection of new attacks for the test partition should not be performed arbitrarily according to these four class of intrusion. This is because the selected attacks for each partition may not be similar in terms of the feature values (McHugh 2000), which has been demonstrated here. Instead, new attacks for the test set should be selected based on some degree of similarity with the intrusions in the training set to allow for a fair test.

Whichever data subset/partition is adopted, there is a need for more clear documentation of the data and experiment method, to support independent validation of the findings and comparison across studies. There is also a need for a more thorough understanding of the nature of the data, in order to be able to interpret effectively the results, and assess their implications in practical application.

CHAPTER 5

Learning from imbalanced data

Researchers have applied machine learning to intrusion detection for several decades, however, with varying degrees of success. This was the focus in the previous chapter, investigating discrepancies in the findings obtained with the KDD Cup '99 data set. That investigation indicated that the results are significantly affected by the class balance in the various subsets adopted in the literature. More specifically, the findings indicate that this is the reason *U2R* and *R2L* intrusions have been detected poorly, which has not been considered previously in the literature.

Although applied popularly to intrusion detection, ANNs and DTs have been found to learn poorly from imbalanced data as they are biased towards the major class(es) (Chawla 2003, Jo and Japkowicz 2004). The reason for this is that the training methods used are based on a general measure of error of the classifier (Weiss and Provost 2003, Weiss 2004). Therefore, if there is a significant imbalance, the major class(es) will have more 'weight' in the calculation of error, to the extent that the classifier may simply 'ignore' the minor class(es) (Kotsiantis *et al.* 2006, Weiss and Provost 2003). An alternative training method is examined here, using a Genetic Algorithm (GA) to evolve the weights of Multi Layer Perceptrons (MLPs). The essential benefit of this is the ability to perform arbitrary transformations of individual classification rates into a fitness measure, instead of using a general error based metric. However, such a metric can still be adopted as an evaluation function, which allows demonstration of how the this affects the performance.

A review of research on class imbalance is provided in Section 5.1. Details of GAs and how they can be used to evolve MLPs are discussed further in Section 5.2, followed by the experiment method in Section 5.3. The results from experiments are presented and discussed in Sections 5.4 and 5.5.

5.1 Class imbalance

Learning from imbalanced data is a significant challenge that has received much attention in the data mining community in recent years. Class imbalance refers to unequal representation of classes in a data set, *i.e.*, there are more instances of one/some class(es) than others. Japkowicz (2003) distinguishes between two types of class imbalance: *between classes* and *within class*. The latter refers to imbalances between natural clusters of data within one class. This is discussed further in Section 5.1.1, which reviews the background to this research area and the challenges that class imbalance poses. Throughout this section, unless stated otherwise, class imbalance refers to *between classes*.

Sections 5.1.2 and 5.1.3 consider the main levels on which class imbalance can be addressed (Chawla *et al.* 2004): (1) data level (sampling training data) and (2) algorithm level. It has become popular in recent years to apply classifier combination techniques to learn from imbalanced data, which fall under both data and algorithm levels. However, since classifier combination is a different concept, it is treated separately in Section 5.1.4. There are also methods that consider feature selection, which fall under the data level. However, this is not considered here; instead, refer to Kotsiantis *et al.* (2006).

5.1.1 The class imbalance problem

There are few real world problems for which it is possible to obtain an equal amount of data for each of the classes. Furthermore, for some problems it is the minor class that it is imperative to classify correctly, such as medical diagnosis (Mena and Gonzalez 2006), which typically operates with few malignant instances compared with benign. This is a critical issue because some machine learning algorithms are biased towards the major class(es) (Chawla 2003, Jo and Japkowicz 2004, Weiss and Provost 2003), and, therefore, obtain poor classification rates on the minor class(es). The classifier may even simply ignore the minor class and predict everything as the major class (Kotsiantis *et al.* 2006, Weiss and Provost 2003), whilst still achieving a (misleadingly) high overall accuracy. Consequently, if the minor class represents cancer, this would never be detected.

It is only in recent years that class imbalance has received widespread attention from the research community (Weiss 2004). Early work can be traced back to DeRouin *et al.* (1991), focusing on training ANNs on imbalanced data. Approximately a decade later, two workshops were arranged for this topic (Holte *et al.* 2000, Chawla *et al.* 2003). Class imbalance is now considered a critical issue that has been addressed in several domains, including *inter alia* medicine (Cohen *et al.* 2006, Mazurowski *et al.* 2008, Mena and Gonzalez 2006), chemistry and biology (Al-Shahib *et al.* 2005, Tan *et al.* 2003), natural language processing (Kobyliński and Przepiórkowski 2008), lexical acquisition (Kermanidis *et al.* 2004), text recognition (Stamatatos 2008), customer churn (Burez and van den Poel 2009, Xie *et al.* 2009), credit scoring (Huang *et al.* 2006), intrusion and fraud detection (Cieslak *et al.* 2006, Phua *et al.* 2004).

As mentioned previously, Japkowicz (2003) refers to two types of class imbalance, *between classes* and *within a class*. The class imbalance referred to above is between classes, which is the main focus of the research on class imbalance. Class imbalance within a class is related to *small disjuncts* in data (Holte *et al.* 1989), which can be described as clusters of data within a class. Disjuncts are typically used in techniques such as decision trees to perform their classification. However, the small disjuncts in data have been shown to be more prone to producing errors (Weiss 1995) and may simply be ignored by the classifier, which may build a more *accurate* model based on the larger disjuncts (Japkowicz 2003, Weiss 1995). Furthermore, Japkowicz (2003) argues that small disjuncts may be the underlying reason for unsatisfactory classifier performance. Therefore, applying methods for dealing with *inbetween* class imbalance may not have a positive effect on the performance (Japkowicz 2003, Jo and Japkowicz 2004). Some research suggests that studies should consider both class imbalance between classes and the problem of small disjuncts within classes (Japkowicz 2003, Jo and Japkowicz 2004, Weiss 2004). A benefit of addressing the within class imbalance is that it also reduces the imbalance between classes (Weiss 2004).

Most of the methods that have been proposed for dealing with class imbalance, which are discussed in the following sections, address only class imbalance between classes, and generally do not consider the issue with small disjuncts.

5.1.2 Data level

One of the most common methods of dealing with class imbalance is to apply some form of sampling to the training data. This may be in the form of *undersampling* the major class(es) or *oversampling* the minor class(es), which can be conducted *randomly*, *directed*, or according to methods such as *clustering* and *evolutionary algorithms*.

Undersampling techniques aim at removing instances from the major class(es) to obtain a balanced data set. Oversampling aims at producing more instances of the minor class(es) to balance the data set. Some studies suggest that one type of sampling is superior to the other (Drummond and Holte 2003), whilst other researchers contend that it is problem dependent (van Hulse *et al.* 2007, Weiss 2004). Moreover, Weiss (2004) suggests that both sampling techniques should be used in some cases for data sets with class imbalance.

For both sampling approaches, random sampling can be conducted, which may produce satisfactory results. The danger of random undersampling is that important data may be excluded since the data distribution is not considered Kotsiantis *et al.* (2006). Similarly, not considering the data distribution when performing oversampling may induce unfruitful biases in the data. Furthermore, in this case, it is important to consider the noise that may be present in the minor class (Weiss 1995). That is, random oversampling may duplicate a significant amount of noise instead of the important data. Nevertheless, both sampling approaches have been shown to help (Japkowicz 2000b;a, van Hulse *et al.* 2007).

There are more sophisticated sampling methods that take into account the class distribution. For example, generation of synthetic training samples (An 1996, Chawla *et al.* 2002). However, as Weiss (2004) contends, the true distribution of a ‘real’ problem is generally unknown. Therefore, this is found to be both method and problem dependent (Burez and van den Poel 2009, Weiss 2004).

A sampling technique based on clustering has been proposed (Japkowicz 2001, Nickerson *et al.* 2001), which aims at explicitly oversampling the small disjuncts of the minor classes. Empirical findings reported in (Jo and Japkowicz 2004), with a C4.5 DT and a MLP trained with backpropagation, are encouraging. The clustering approach outperformed four other methods that do not consider the small disjuncts: random oversampling, random undersampling and generation of synthetic training samples. The approach also outperformed a pruning method for DTs, which does consider the small disjuncts. However, it is interesting to note that on synthetic data, random oversampling achieved the same performance as clustering.

García and Herrera (2009) propose using an evolutionary algorithm to perform undersampling, which they evaluated on 28 data sets from the UCI machine learning database repository. The results are compared with 10 other methods of dealing with class imbalance, including random undersampling and clustering. They found that the evolutionary approach outperformed the other methods on the data sets with the greatest imbalance. However, they did not include oversampling methods, and limited the investigation to binary classification problems.

5.1.3 Algorithm level

According to Weiss (2004), one of the reasons many data mining and machine learning techniques learn poorly from imbalanced data is the evaluation criteria they adopt. Most techniques evaluate the performance of the classifier during training based on an overall accuracy/error. This biases the classifier towards the major class, which in cases of extreme imbalance, can lead to the minor class being ignored (Weiss and Provost 2003). ANNs and DTs are good examples of such classifiers (Chawla 2003, Jo and Japkowicz 2004).

Weiss (2004) and Kotsiantis *et al.* (2006) discuss several alternative evaluation metrics, including AUC[1], F-measure, and weighted metrics (cost-sensitive learning). The latter has become a popular approach, in which the error or classification rate for each objective is multiplied by a cost/weight. For example, Jo and Japkowicz (2004) consider problems with a 1:9 ratio between two classes. To balance the evaluation of the two classes, the error on the major class may be multiplied with a weight of 0.1, whilst the error on the minor class is multiplied by a weight of 1.0.

Typically, cost sensitive learning uses a weight matrix, corresponding to the confusion matrix obtained from the performance of the classifier, in which the diagonal (correct classification) has no penalty (Kotsiantis *et al.* 2006). Further domain knowledge may be incorporated into such a weight matrix. See, for example, the weight matrix used in the evaluation of the KDD Cup '99 competition, in Table 5.1. Although this weight matrix was only used to evaluate the results, and does not attempt to address class imbalance, it is a good example of incorporating domain knowledge in which the matrix puts particular emphasis on penalising misclassifications of *U2R* and *R2L* instances as *Normal*.

Table 5.1: Weight matrix for evaluating results for the KDD Cup '99 competition.

	Normal	Probing	DoS	U2R	R2L
Normal	0	1	2	2	2
Probing	1	0	2	2	2
DoS	2	1	0	2	2
U2R	3	2	2	0	2
R2L	4	2	2	2	0

Cost sensitive learning has been adopted in many recent techniques, such as the classifier combination methods discussed in the following section, and weighted rough sets, as proposed by Liu *et al.* (2008c;b). Other methods have been proposed for learning from imbalanced data, such as: modifying the decision threshold (moving it after training) (Weiss 2004, Zhou and Liu 2006); one-class learning algorithms (Kotsiantis *et al.* 2006) and learning only the rare class (Weiss 2004); active learning (Ertekin *et al.* 2007b;a); and specifically for DTs, alternative pruning techniques (Jo and Japkowicz 2004, Kotsiantis *et al.* 2006).

5.1.4 Classifier combination

There are many recent applications of classifier combination that address class imbalance. For more information on classifier combination, refer to Section 6.2 on page 105.

Several well known classifier combination techniques have been applied to problems with class imbalance, such as mixture of experts (Estabrooks and Japkowicz 2001), random forest (Khoshgoftaar *et al.* 2007), and boosting (Guo and Viktor 2004). These applications still address the problem at the levels discussed above. For example, bagging and boosting algorithms primarily operate on the data level, performing sampling. Bagging performs random sampling (with replacement) to train different classifiers, whilst boosting performs sampling based on a distribution that is continuously updated to increase the chances of sampling instances that are often misclassified.

The sampling concepts and cost-sensitive learning have been incorporated more explicitly in some methods, *e.g.*, balanced and weighted random forests (Chen *et al.* 2004, Kobyliński and Przepiórkowski 2008, Xie *et al.* 2009), balanced bagging (Hido and Kashima 2008), and cost sensitive boosting (Sun *et al.* 2007).

[1] When analysing Receiver Operator Curve (ROC) graphs, the Area Under the Curve (AUC) can be computed as a metric (Bradley 1997).

5.1.5 Discussion

Both sampling and cost sensitive approaches have been widely adopted in the literature. Initially, Jo and Japkowicz (2004) stated that cost sensitive learning was considered more successful. However, they argue that there is a significant limitation to the cost sensitive approaches since they address the imbalance at the class level. Sampling techniques offer more flexibility, allowing sampling of different parts of the data set, and can consider explicitly the problem of small disjuncts.

Kotsiantis *et al.* (2006) concurs with Jo and Japkowicz (2004), although the findings in the literature indicate that practitioners still take an *ad hoc* approach to applying these techniques. Weiss (2004) has provided several guidelines as to which methods can be recommended for dealing with specific problems with imbalanced data sets. However, current research indicates that the choice of sampling approach, and choice of distribution if not random sampling, depends on the method and problem at hand (Burez and van den Poel 2009, Weiss 2004). Similarly, for weighted approaches, determining optimal weights is also an *ad hoc* process, which becomes increasingly complex the more classes there are.

Multi-class problems have not been considered in much of the research on class imbalance. Zhou and Liu (2006), however, conducted a comprehensive empirical investigation with cost sensitive ANNs, which included 21 data sets from the UCI repository. They examined the effects of oversampling, undersampling, SMOTE sampling (Chawla *et al.* 2002) (oversampling by creating synthetic training samples based on existing samples), and threshold moving, for both single classifiers and ensembles. Although the techniques were effective on the two class problems, they were not on the multi-class problems; some even had a detrimental effect. One reason for this is that there are more classes that a sample can be misclassified as, which increases the complexity of the problem. One potential solution they suggest is to convert the problem to several binary classification tasks and combine classifiers based on each pair. This does not, however, change the complexity of setting weights; it does, as Zhou and Liu (2006) contend, add complexity to the task for the user.

5.2 Genetic algorithms

The Genetic Algorithm (GA) is a population based heuristic search/optimisation technique inspired by the analogy of biological evolution, proposed by Holland (1992) in 1975. An overview of the general process (evolutionary cycle) of the GA is provided in Section 5.2.1. Thereafter, the application of GAs is discussed in Section 5.2.2, which discusses the type of problems GAs are useful for and implementation/configuration considerations. Section 5.2.3 discusses the use of GAs for evolving ANNs.

5.2.1 Evolutionary cycle

Potential solutions to a given problem are represented as *individuals* (or *chromosomes*) in a *population*. From a random starting point, the search towards good solutions is conducted by selecting *fit* individuals (parents) from the population, which are recombined to create offspring solutions based on their genetic material. These offspring are subject to some (small) probability of mutation, and are evaluated based on some fitness measure. This fitness measure needs to be quantified in such a manner that the GA can attempt to minimise or maximise this value.

The premise of the search is that offspring will occasionally be *fitter* (better) than their parents, that the offspring of highly fit parents will tend to be better than those of poorly fit parents, and that by selecting parents that are of above average fitness, the average fitness of the population will tend to increase. The offspring are inserted into the (typically fixed size) population according to some replacement strategy, and the cycle is continued until some stopping criterion has been met. This criterion may be based on solution quality, measure of population convergence, or a maximum number of iterations of the evolutionary cycle. Once the stopping criterion is met, the 'best' individual is typically selected as the solution to the problem. Refer to Figure 5.1 for an illustration of this process.

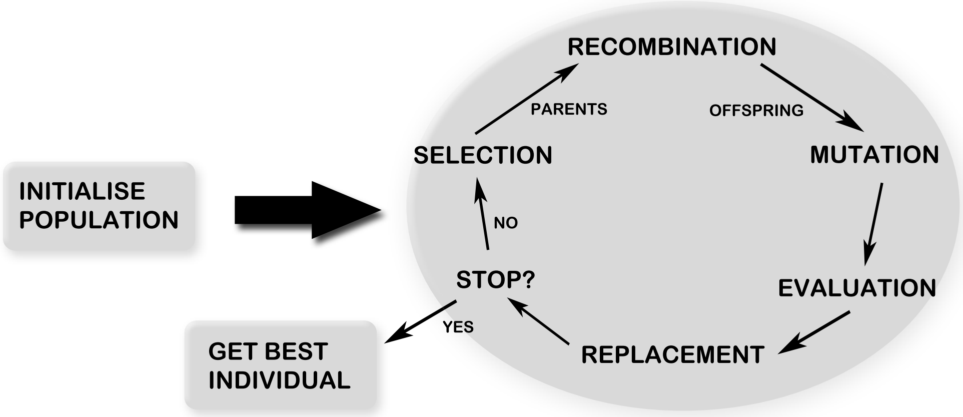


Figure 5.1: Evolutionary cycle of a genetic algorithm.

5.2.2 Application of genetic algorithms

One of the strengths of GAs is that they can be applied to complex domains for which there is insufficient knowledge of the problem, or a lack of a theoretical framework to solve the problem. Furthermore, GAs are well suited to find good solutions for problems that have a solution space that is too great for exhaustive approaches to consider. Since the GA is a heuristic technique, it cannot guarantee that the best solution will be found. It does, however, give near optimal solutions within reasonable time limits for complex problems where other algorithms fail.

GAs can be configured to explore the search space (global search), or exploit potentially good regions (local search). Exploration may be encouraged by increasing the population size and probability (and strength) of mutation. The performance of the algorithm, on a particular problem, will depend on the balance between exploration and exploitation; too much emphasis either way will degrade the performance. Some problems will require more exploration, whilst others exploitation. There are operators that attempt to address both exploration and exploitation, such as the exponential creep mutation (Davis 1989), which performs frequent small mutations with occasional large mutations. In this context, it is important to consider the *'no free lunch'* theorems (Wolpert and Macready 1997), which states that there is no 'best' algorithm for all problems. Similarly, there is no optimal configuration that will be ideal for all problems. Although there are theories on how to improve the genetic search (Booker 1987), the configuration parameters are therefore typically determined for each particular problem in an *ad hoc* manner.

As discussed above, GAs can operate with minimal domain knowledge, given that a potential solution to a problem can be encoded in a chromosome representation. A chromosome can be divided into a number of genes (typically fixed number), which correspond to the parameters of the problem that needs to be optimised/solved. Originally, the genes were encoded as binary strings. However, there are other representations that may be more intuitive for some problems, such as floating point numbers (Davis 1989, Eshelman and Schaffer 1993, Wright 1991), permutations (Whitley 2000) (for combinatorial problems, such as traveling salesman problems), and trees (Angeline 2000) (typical for genetic programming (Koza 1992)). For example, for the application considered here, a numerical representation is convenient since the parameters that are optimised are numerical weights of an ANN.

The evaluation function of a GA may incorporate domain knowledge, if it is available, to help guide the search towards optimal solutions, as discussed previously in Section 3.12.1 on page 47. Moreover, some problems may have multiple objectives, such as maximising the true positive rate of a classifier whilst minimising the false positive rate. Conventional techniques convert the objectives of such a problem into a single fitness value, using either *a priori* information or an *ad hoc* procedure of assigning weights to each objective. However, GAs can be applied to evaluate multiple objectives, providing a set of solutions that exhibit different trade-offs among the objectives. More details on this in Section 6.1 on page 97, as multi-objective optimisation is considered in the following chapter of this thesis.

5.2.3 Evolutionary neural networks

Evolutionary Neural Network (ENN) is a general term for using GAs, or another evolutionary algorithm, to optimise one or more parts of an ANN, such as (Yao 1999):

- Weights.
- Topology:
  - Number of layers.
  - Number of neurons in each layer.
  - The connections between them.
- Input features.

To some researchers, the term ENN refers to evolving both the topology and weights of ANNs, since this becomes a new, highly specialised, network for the problem at hand. In comparison, using a GA to optimise the weights of a Multi Layer Perceptron (MLP) does not change the fact that the ANN is an MLP; the GA merely becomes a different method of training.

Section 3.12.4 on page 51 reviewed several applications of ENNs to intrusion detection, reporting on much improved performance when allowing the GA to optimise the weights and topology of the networks. However, there are many potential benefits of using GAs to 'only' optimise the weights of MLPs compared with other training algorithms, such as:

- The GA is robust to noise (Goldberg 1989, Goldberg *et al.* 1992),
- is not as prone to 'getting stuck' in local optima as backpropagation (Yao 1993; 1999),
- is independent to the number of layers, and
- can be readily parallelised.

Related to the second point, Tian and Gao (2009) utilise a GA to provide initial weights for backpropagation. This gives the algorithm a better starting point, helping to prevent it from converging on a local optimum. Furthermore, as Yao (1999) contends, GAs are beneficial because they do not rely on gradient information of the error, which may not be available, or may be costly to calculate. The latter point, above, is a key part of this research, in which bespoke fitness measures may be adopted to find better solutions to problems where a global measure of error/accuracy is unfruitful.

When optimising the weights of MLPs, the chromosome of each individual in the population represents weights of an MLP, with each gene of the chromosome representing the weight of a connection between two neurons. Therefore, for large, fully connected MLPs, the number of weights, and, thus, the dimensionality of the problem, can be very high. For example, adopting all of the features of the KDD Cup '99 data set requires 41 input nodes, and 5 output nodes (one for each of the classes). If there is just one hidden layer of 20 nodes, the total number of connection weights that the GA has to optimise is 920, plus up to 66 bias weights depending on the configuration of the neurons in each layer.

To optimise the weights of MLPs, a real valued representation is a logical choice. A binary representation would require unnecessary transformation from binary to numerical values when the gene values are applied to MLPs as weights. Furthermore, not only did Janikow and Michalewicz (1991) find the numerical representation to be quicker, they found it to make the GA more robust and precise. However, if the topology is to be optimised, then other representations are necessary. Representations reported in the literature to optimise the topology, and weights, are tree structures (Stanley and Miikkulainen 2002) and, more recently, matrices (Han and Cho 2006, Kim and Cho 2008).

5.3 Method

The aims and motivations of this investigation are discussed in Section 5.3.1. Details of the proposed ENN method are provided in Section 5.3.2, followed by specifications of the MLP that is adopted for comparison, in Section 5.3.3. Section 5.3.4 describes the data set and validation method, followed by the metrics in Section 5.3.5. The empirical investigation is outlined in Section 5.3.6.

5.3.1 Aims and motivations

This second empirical investigation focuses on learning from imbalanced data, and has two main aims, as discussed below.

Aim 2.1: Determine whether class imbalance is the cause of poor detection rates of MLPs reported in the literature

As discussed in Section 4.1 on page 55, *U2R* and *R2L* intrusions have been particularly difficult to detect in existing studies using machine learning. The empirical findings from the previous investigation, in Chapter 4, suggest that class imbalance is a significant factor. This has not been considered in the existing studies as a reason for the poor performance on these classes. A few researchers have, however, touched on a related factor, arguing that the poor performance on *U2R* is due to a lack of data (instances to learn from) (Bouzida and Cuppens 2006a;b, Stein *et al.* 2005). This is what Weiss (2004) refers to as absolute rarity, which this study also acknowledges. The extreme class imbalance in the KDD Cup '99 data set also renders this a problem of relative rarity.

Bouzida and Cuppens (2006a;b) report that ANNs, specifically MLPs trained with backpropagation, are unable to detect the *U2R* class. This is a classifier that is known to be biased towards the major class (Jo and Japkowicz 2004), and, therefore, is prone to perform poorly on this class since it is a minor class. The first aim of this investigation is to determine whether it is indeed the class imbalance that has led to this poor detection. More specifically, to determine whether this is caused by the training algorithm, or whether it may be due to limitations of the classifier itself.

Aim 2.2: Develop an alternative method of training MLPs for imbalanced data

To explore class imbalance, Genetic Algorithms (GAs) are used here to optimise the weights of the MLPs. The benefits of this are two fold:

1. Provides an approach that allows for arbitrary transformations of classification rates and/or errors into fitness measures, which can be designed to be unbiased to any particular class. Therefore, giving a training method that may learn better from imbalanced data.

2. Aim 2.1 can be answered concretely, comparing the performance obtained with evaluation functions that evaluate the MLPs in the same manner as backpropagation with evaluation functions that are not biased towards the major class(es).

5.3.2 Evolutionary neural network

The ENN employs a real coded GA to optimise the connection weights of feed forward, fully connected, MLPs. Implementation details are provided below.

5.3.2.1 MLP specifications

According to Hornik *et al.* (1989), a three layer MLP (*i.e.*, having one hidden layer) can approximate any continuous function, assuming an infinite number of neurons in the hidden layer. However, both three and four layer MLPs are examined here since Pinkus (1999) suggests that two hidden layers are generally more promising for more complex problems, and Sontag (1992) notes that this is necessary to approximate arbitrary discontinuous functions.

The number of neurons in the hidden layer(s) is determined by a systematic investigation (enumeration over a given set of values), whilst the input and output layers are determined by the features of the data set and the classes (one output neuron per class). The range of neurons in the hidden layers was determined according to preliminary experiments, since few researchers include this information; of the existing work, as discussed in Section 4.1 on page 55, only Sabhnani and Serpen (2003) provide full details. They adopt three layered MLPs, examining the number of neurons in the hidden layer in the range 40 to 80, in increments of 10. These ranges are expanded here, due to the observations during preliminary experiments: for three layers, the following set of neurons in the hidden layer are investigated {30, 50, 70, 100, 120}, and for four layers {5-5, 10-10, 15-15, 20-20, 25-25}.

The predicted classification is determined according to the output neuron with the highest value. According to the benefits discussed in (Kalman and Kwasny 1992), a hyperbolic tangent (tanh) activation function is adopted, and the connection weights are in the range [-1, 1].

5.3.2.2 Genetic algorithm specifications

A binary and numerical chromosome representation is possible when evolving the weights of MLPs (Yao 1999). Floating point numbers are used here, in which each gene represents the real value of a connection weight in an MLP. The floating point representation has been found to be quicker, more robust and precise compared with the binary representation (Janikow and Michalewicz 1991). The use of a numerical representation requires different operators from the classical operators proposed for the GA as these rely on a binary representation. Crossover is referred to as *recombination*, because it is no longer particularly useful to literally cross over genes. Examples of numerical crossover operators include *inter alia* uniform crossover (Syswerda 1989), linear crossover (Wright 1991), blend crossover (BLX-α) (Eshelman and Schaffer 1993) and uni-modal normal distribution crossover (UNDX) (Ono and Kobayash 1997).

Michalewicz (1999, pp. 105-106) states that the performance of GAs can be enhanced due to the numerical operators, as they are capable of incorporating domain knowledge. For example, UNDX has been shown to be beneficial for epistatic problems (Ono and Kobayash 1997), and BLX-α can produce offspring within an extended region around the parent individuals, which removes bias towards the center of the population (Eshelman and Schaffer 1993). BLX-α has been adopted here.

Mutation normally occurs as a perturbation of genes, typically adding or subtracting an arbitrary small value. The perturbations may be uniformly distributed, or non-uniformly distributed, such as in Gaussian mutation (Beyer and Schwefel 2002), which is adopted here. Gaussian mutation is parameterised with a per-gene mutation rate, and a mutation strength, which defines the standard deviation (spread) of the Gaussian distribution. Note that a distribution with a mean of zero is used, and therefore mutations increase or decrease a given gene value with equal probability.

Two main population structures exist: *generational* and *steady-state* (Syswerda 1989). A generation refers to one reproduction cycle, which encompasses selection and replacement of individuals. In a generational GA, a reproduction cycle will replace a large block of individuals, or even the entire population. In a steady-state GA, only a small number of individuals is replaced, or even just one in each cycle. Due to the significant differences in the computational efforts of such iterations, dependent on the population structure, the number of fitness evaluations is generally used as a metric to measure the progress of the genetic search rather than the number of generations.

The GA implemented in this study has a steady state structure with binary tournament selection, due to the observations in (Whitley 1989). Whitley made a comparison between two well known algorithms: a steady-state GA, GENITOR (Whitley 1989), and a generational GA, GENESIS (Grefenstette 1990). Rank based selection yielded better results for both algorithms. Furthermore, the steady-state GA outperformed the generational GA. The combination of a steady-state population structure and tournament selection produces high growth rates, which is an element of controlling the convergence properties of the GA (Goldberg and Deb 1991). One of the strengths of such a GA is that the selection pressure can be directly controlled and is uniformly scaled across the population (Whitley 1989).

A steady-state GA requires a replacement strategy to determine how offspring are inserted into the population. There are several approaches to replacement, including *inter alia* random, probabilistic, replacing the oldest individual, or replacing the worst. Syswerda (1991) demonstrates empirically that random replacement is the least successful of these strategies. Replacing the worst individual is commonly used, which has the elitist benefits advocated in (De Jong 1975, Gordon and Whitley 1993). Elitism refers to guaranteeing that the best individual in the population remains in the population after replacement. This replacement scheme has therefore been adopted here; replacing the worst individual if the offspring is fitter.

The following configuration parameters were determined according to preliminary experiments: population size of 20; over-initialising with 100 additional, randomly generated, individuals (according to the benefits demonstrated by Bramlette (1991)); mutation rate of 0.2; mutation strength of 0.05; and α set to 0.5 for the crossover operator. Training is conducted for 20,000 function evaluations.

5.3.2.3 Evaluation functions

The following evaluation functions are examined:

Eval1: Minimise the error; mean of the distance of each classification from the expected output pattern.

Eval2: Eval1 in combination with rewarding correct classification.

Eval3: Minimise the sum of rewards for correct classification; the reward for each class calculated proportionally to the number of instances in the training set[2].

Eval4: A combination of Eval1 and Eval3.

Eval5: Maximise the sum of individual classification rates.

The first evaluation function is similar to the error measure used when training an MLP with backpropagation, which is a typical evaluation function adopted in ENNs (Yao 1993). Let N be the number of instances in the training set, then the mean error, e, is calculated as

[2] In practice, this is similar to the error measure proposed by Quinlan (1991), which takes into account the distribution of classes. However, Quinlan only considers this on a proportion of the training set.

$$e = \frac{1}{N}\sum_{i=1}^{N} |outputPredicted_i - outputExpected_i| \tag{5.1}$$

Eval2 is an extension of Eval1, which increases the selection pressure by subtracting a certain value from the fitness for each correct classification during training. During preliminary studies, values in the range $[0.1, 2]$ were examined, in which 1 was found to be the most beneficial, *i.e.*, obtaining comparatively good classification rates whilst training more quickly than Eval1. Too large a reward leads to a stronger bias towards the major class, causing the the GA to converge on a local optimum.

The three latter evaluation functions were developed to be unbiased to the class balance. The proportional classification reward r for class c in Eval3 is calculated by

$$r_c = 1 - (\frac{\#instances_c}{\#instances_{tot}}) \tag{5.2}$$

The combination of Eval1 and Eval3 is composed in the same way as Eval2, except that the reward is given proportionally to the instances of each of the classes. Eval5 is similar to Eval3, though yielding a different selection pressure. To clarify the difference between these two evaluation functions, consider a three class problem for which Eval5 will give a maximum fitness value of 300 if all three classes are detected 100%[3]. For example, if 99% *Normal*, 10% *U2R* and 95% *R2L* instances are correctly classified, the fitness score is 204 $(99+10+95)$. Eval5 can be used as a minimisation function by subtracting this value from 300, giving a fitness score of 96 $(300-204)$. Consequently, the best solution will be one that has a fitness score of 0.

5.3.3 MLP trained with backpropagation

Similarly to the machine learning software used in the previous investigation, an MLP implementation has been adopted from Weka (Anon 2006, Witten and Frank 2005). The training process of this algorithm is controlled by a learning rate and momentum, and it is stopped after a predefined number of epochs or by means of early stopping. Each of these configurations is considered below.

5.3.3.1 Learning rate and momentum

Negnevitsky (2005, p. 185) suggests that a common momentum value is 0.95, but such a high value can have a negative effect on the training process, as demonstrated by Haykin (1998, p. 194). Further, it appears that the optimum value for the momentum is dependent on the learning rate (Haykin 1998, pp. 214–216). Therefore, these parameters are examined systematically, with ranges capped at 0.5, following Haykin (1998) and preliminary experiments. High momentum values caused the MLP to converge too quickly during preliminary experiments, which led to very poor classification rates. Hence, the learning rate and momentum values examined are {0.05, 0.1, 0.2, 0.3, 0.4, 0.5} and {0.1, 0.2, 0.3, 0.4, 0.5}, respectively. Decay of the learning rate is enabled.

5.3.3.2 Epochs and early stopping

The number of epochs for the training process to run and the early stopping parameters are constant throughout this investigation. The aim is to allow early stopping to terminate the training process when the error increases on a validation set over a number of epochs (validation threshold).

Only three authors (of those discussed in Section 4.1 on page 55) provide information about the stopping criterion adopted. Pan *et al*. (2003) trained for 1500 epochs. Sabhnani and Serpen (2003) examined 40-100

[3] For the empirical investigation conducted in this chapter, a three-class subset of the KDD Cup '99 data set is used, comprised of *Normal*, *U2R* and *R2L*. Further details are provided in Section 5.3.4.

epochs, finding 60 to be optimal. Bouzida and Cuppens (2006, 2006a, 2006b) also examined a low number of epochs, 10-90, finding 25 to be optimal. These large differences in the number of epochs can be seen to relate to the size of the data set that is used for training. As shown by Haykin (1998, pp. 191–196) the number of epochs required to reach nearly an identical error measure and classification rate during testing decreased significantly with a larger training set (when using sequential training).

Experiments were conducted to determine the number of epochs and the validation threshold. For the sake of brevity, the results of this investigation are summarised here, whilst full details are provided in Appendix B.2.1 on page 204. According to that investigation, the following configurations are used: training for 100 epochs, which may be terminated by early stopping if the error on a validation set increases on 5 consecutive epochs. For the validation set, 20% of the training set is reserved.

5.3.4 Data set and preprocessing

The 10% version of the training set is used here, which avoids some of the methodological complexities related to the *R2L* class when the test set is used. Refer to Section 4.5 on page 71 for details. Furthermore, not adopting the test set focuses the study more on the class imbalance problem, avoiding unnecessary complexities in the analysis related to the detection of entirely new attacks. To focus the investigation further, a version of the data set has been produced that excludes *Probing* and *DoS* intrusions. As seen in Table 5.2, this gives a class balance that is arguably more realistic, according to criticisms of McHugh (2000) and Portnoy *et al.* (2001). Portnoy *et al.* (2001) assume that *Normal* traffic makes up 98% of the total amount of data, and, in a related investigation, Eskin *et al.* (2002) assume that *Normal* traffic makes up 98.50% – 99%.

Table 5.2: Proportions of attack classes in the 10% training set of the KDD Cup '99 data set.

Class	Instances (proportion)
Normal	97,278 (98.80%)
U2R	52 (0.05%)
R2L	1,126 (1.14%)

Regarding properties related to class imbalance, this data set has both absolute and relative rarity (Weiss 2004), and small disjuncts are likely for the *U2R* class in particular, which is composed of several attacks that have few instances (as low as 2).

Since the MLP and ENN are very time consuming to train, and the systematic sampling of the configuration space requires many trials, holdout validation is selected rather than cross validation. Similarly to the previous investigation, the data set is partitioned (chronologically) so that 80% of the instances are used for training and the remainder for testing, according to the findings of Kearns (1996). The partitioning is done according to individual attacks, ensuring that each attack has at least one instance in the test set. The classification, however, is based on the three classes, *Normal*, *U2R* and *R2L*, rather than the individual attacks. The nominal parameters are enumerated and all parameters are scaled within the range $[0,1]$, according to the benefits highlighted by Haykin (1998, p. 181) and LeCun (1998). The minimum and maximum values used for this scaling process were based on the data in the training and test sets.

5.3.5 Metrics

The same metrics are used as in the previous investigation Confusion matrices of classifications are produced, which present predicted and actual classifications. From this, trends in classification behaviour can be identified, and accuracy, true and false positive and negative rates are derived.

Since the machine learning techniques considered here are stochastic, it is desirable to obtain a measure of mean performance over several trials. However, due to the magnitude of the data and experiments conducted here, the training times do not allow for this in practice. Therefore, the best out of three trials is selected as an indication of the classifiers' 'best' performance.

5.3.6 Outline of empirical investigation

Since DTs were adopted in the previous study, and have also has been reported to have the same issues as MLPs with learning from imbalanced data, they are also included in this study to provide a consistent benchmark. Refer to Section 4.3.3 on page 61 for implementation details.

In addition to training the MLPs with GAs, an MLP trained with backpropagation is also used to provide a benchmark for conventional error based training, against which the performance of the evolved networks with each of the evaluation functions described above can be evaluated.

The DT and MLP are known to be biased towards the major class(es), and, thus, are likely to perform worse on *U2R* and *R2L* compared with *Normal*. The first two evaluation functions of the ENN exhibit the same bias, and should, therefore, give similar classification behaviour. The latter three evaluation functions do not exhibit the same bias. Eval3 and Eval5 are unbiased to the class balance, but induce different selection pressures. Eval4 is a combination of a general error measure (Eval1), which is biased towards the major class(es), and a proportional reward for correct classification according to Eval3. Therefore, these three evaluation functions are expected to allow the MLPs learn better from imbalanced data, leading to improved performance on *U2R* and *R2L*.

5.4 Results

Results for each of the classifiers are presented in their respective sections, below, followed by a summary and comparison of results in Section 5.4.4. The following abbreviations are used here: 3L and 4L for three and four layer MLPs respectively; HN for neurons in the hidden layer(s).

5.4.1 Decision tree

The confidence factor (for pruning) does not have much impact on the results. Confidence factors in the range 0.05–0.10 gave identical results, as did confidence factors in the range 0.15–0.5. The former group led to the best results, which are presented in Table 5.3. Some misclassifications of *Normal* and *R2L* intrusions occur with confidence factors above 0.10, leading to higher false positives and false negatives. Only 1 *U2R* intrusion was detected.

Table 5.3: Confusion matrix for DT with confidence factors of 0.05 and 0.10.

Actual\Predicted	Normal	U2R	R2L	%correct
Normal	19,454	1	1	99.99
U2R	10	1	0	9.09
R2L	0	1	225	99.56
%correct	99.94	33.33	99.56	

$Accuracy = 99.93\%$

$TPR = 95.78\% \quad FPR = 0.01\%$

5.4.2 MLP trained with backpropagation

The MLP trained with backpropagation is unable to detect *U2R* intrusions, which coincides with the findings of Bouzida and Cuppens (2006a;b). However, the FPR remains low, which can be seen in Table 5.4, where only 30 (out of 19,426) *Normal* instances are misclassified as *R2L*.

Table 5.4: Confusion matrix for best MLP in terms of true positive rate.

Actual\Predicted	Normal	U2R	R2L	%correct
Normal	19,426	0	30	99.85
U2R	11	0	0	0.00
R2L	5	0	221	97.79
%correct	99.92	0.00	88.05	

$Accuracy = 99.77\%$

$TPR = 93.25\% \quad FPR = 0.15\%$

Table 5.5 provides an overview of classification rates of MLPs that performed well with respect to different evaluation criteria. Note that the highest TPR (despite a high FPR) is not as high as the DT obtains, which, at the same time, obtains the lowest FPR.

Table 5.5: Overview of results obtained with the MLP trained with backpropagation, compared with the DT.

Technique	Accuracy%	TPR%	FPR%	Normal%	U2R%	R2L%
MLP, highest TPR: 3L, 30HN	99.77	93.25	0.15	99.85	0.00	97.79
MLP, lowest FPR: 3L, 120HN	99.78	83.12	0.02	99.98	0.00	87.17
MLP, highest accuracy: 3L, 50HN	99.83	92.83	0.08	99.92	0.00	97.35
DT	99.93	95.78	0.01	99.99	9.09	99.56

No significant benefit was found using four layers compared with three; the same behaviour was observed and the results are nearly identical. This coincides with the findings of Bouzida and Cuppens (2006a;b). Some trends were observed regarding the values of the learning rate and momentum, details of which, not being pertinent to the specific aims of this investigation, can be found in Appendix B.2.2 on page 206.

5.4.3 Evolutionary neural network

Results obtained with each of the five evaluation functions described in Section 5.3.2.3 are presented in their respective sections below. Similarly to the MLP trained with backpropagation, no benefits in adopting four layers were observed. Therefore, the results are listed mainly for three layer ENNs. A complete listing of results is provided in Appendix B.2.3 on page 208.

5.4.3.1 Eval1 and Eval2

These evaluation functions are biased towards the major class, and therefore exhibit similar classification behaviour as the MLPs trained with backpropagation. Most of the ENNs evolved with Eval1 classify everything as *Normal*, as in the example seen in Table 5.6. Although obtaining a high accuracy (98.80%) by doing so, this is obviously undesirable. The four layer ENNs generally classify everything as *Normal*, with the exception of four cases. Only one of these were able to correctly detect an intrusion class, detecting 36.36% *U2R*.

Table 5.6: Classification rates obtained with the ENN evolved with Eval1 and Eval2.

Technique	Accuracy%	TPR%	FPR%	Normal%	U2R%	R2L%
Eval1; 3L, 30 & 100HN	98.80	0.00	0.00	100	0	0.00
Eval2; 3L, 70HN	98.82	2.11	0.00	100	36.36	0.00
Eval2; 3L, 120HN	99.77	81.86	0.01	99.99	0.00	85.84
MLP; 3L, 30HN	99.77	93.25	0.15	99.85	0.00	97.79
DT	99.93	95.78	0.01	99.99	9.09	99.56

Compared with Eval1, a greater proportion of the ENNs evolved with Eval2 classify everything as *Normal*, *i.e.*, obtaining no false positives. Therefore, it can be observed that Eval2 induces a greater bias towards the major class. However, four ENNs were able to detect some intrusions; two of which are included in Table 5.6, above.

The highest accuracy obtained with Eval2 is the same as that achieved with one of the MLPs trained with backpropagation, which allows for a convenient comparison. As seen in Table 5.6, above, the MLP detects more *R2L* instances, but obtains a higher FPR.

5.4.3.2 Eval3

Results obtained with this evaluation function are significantly different from those obtained with Eval1, Eval2 and the MLP trained with backpropagation. With both three and four layers, all three classes are detectable, with not a single case of everything being classified as *Normal*. The FPR does increase as a trade-off for being able to detect the two classes of intrusions. However, it remains below 3% for all three layer ENNs. This is true for the four layer ENNs except for three trials that obtain false positive rates of 3.87%, 7.75% and 22.34%.

The FPR should be small for an IDS to prevent system operators becoming flooded by false alarms, which may prevent them from dealing with actual intrusions. Although a direct comparison is not possible due to the differences in data partitions adopted, a trade-off similar to the one observed here can be seen in Xiang *et al.* (2008), as they report better detection of *U2R* and *R2L* than previous studies, but obtain a FPR of 3.2%. For more details on this study, refer to the discussion in Section 4.6.1 on page 73. An overview of different trade-offs obtained with Eval3 is provided in Table 5.7. The two networks reported here with 50 HN are two different trials, exhibiting two different classification trade-offs.

Table 5.7: Classification rates obtained with a selection of ENNs evolved with Eval3.

Technique	Accuracy%	TPR%	FPR%	Normal%	U2R%	R2L%
Eval3; 3L, 50 HN	99.10	84.81	0.71	99.29	0.00	88.05
Eval3; 3L, 50 HN	97.87	99.16	2.10	97.90	0.00	100
Eval3; 3L, 100 HN	98.73	98.73	1.24	98.76	63.64	98.23

Two ENNs with 50HN demonstrate a typical trade-off observed here, one between FPR/accuracy and TPR, as seen above in Table 5.7. One obtains a lower FPR, but detects less *R2L* instances. However, both are unable to detect *U2R* intrusions. An ENN with 100HN was able to detect all three classes, but does not achieve as high a TPR or as low a FPR as the two other ENNs. The confusion matrix for this ENN is provided in Table 5.8.

Table 5.8: Confusion matrix for a three layer ENN with 100 neurons in the hidden layer, evolved with Eval3.

Actual\Predicted	Normal	U2R	R2L	%correct
Normal	19,214	17	225	98.76
U2R	1	7	3	63.64
R2L	2	2	222	98.23
%correct	99.98	26.92	49.33	

$Accuracy = 98.73\%$

$TPR = 98.73\% \quad FPR = 1.24\%$

5.4.3.3 Eval4

This evaluation function is similar to Eval2, except that the reward is given proportionally according to Eval3. Compared with Eval3, there are more cases of both intrusion classes being detected with three layer ENNs. With four layers, however, there is no such obvious difference. The highest detection rates of *U2R* (whilst still obtaining high detection rates on *Normal* and *R2L*) are obtained with this evaluation function, reaching 81.82%. A confusion matrix for this ENN is provided in Table 5.9, below.

Table 5.9: Confusion matrix for a three layer ENN with 100 neurons in the hidden layer, evolved with Eval4.

Actual\Predicted	Normal	U2R	R2L	%correct
Normal	19,008	37	411	97.70
U2R	2	9	0	81.82
R2L	3	1	222	98.23
%correct	99.97	19.15	35.07	

$Accuracy = 97.69\%$

$TPR = 97.89\% \quad FPR = 2.30\%$

5.4.3.4 Eval5

As with Eval3 and Eval4, detecting both *U2R* and *R2L* is possible. In most cases, both intrusions are detected; *R2L* intrusions generally around 90% and *U2R* between 60–80%. However, this function suffers from an increase in false positives, which, in some cases, can lead to only ~70% of *Normal* instances being classified correctly. The FPR is greater than 3% for 11 out of 15 trials with three layers and 8 out of 15 trials with four layers. Although good performance is obtained in the other trials, this problem was not observed with the previous evaluation functions.

This is the only evaluation function that made it possible to detect *U2R* with 100% accuracy. However, this is one of the cases where *Normal* instances are not detected well; merely 76.86%, with the vast majority of misclassifications as *U2R*. The trade-off between the classification rates, as observed previously, is more prominent with this evaluation function. A selection of ENNs exhibiting different classification trade-offs are presented in Table 5.10.

Table 5.10: An overview of results obtained with the ENN evolved with Eval5.

Topology	Accuracy%	TPR%	FPR%	Normal%	U2R%	R2L%
3L, 30HN	76.96	98.73	23.14	76.86	100	84.07
3L, 70HN	98.21	98.31	1.76	98.24	72.73	97.35
4L, 15HN	98.53	97.05	1.32	98.68	63.64	87.61

As observed above, there is negligible difference between the results obtained with three and four layers. The highest TPR was obtained with a three layer network with 70 neurons in the hidden layer, whilst the lowest FPR was obtained with a four layer network with 15 neurons in each of the hidden layers.

5.4.4 Summary and comparisons

An overview of results obtained with the three classifiers is presented in Table 5.11, which displays the best results for each classifier according to the TPR.

Table 5.11: Overview of the best performing classifiers, selected based on true positive rates.

Technique	Accuracy%	TPR%	FPR%	Normal%	U2R%	R2L%
DT	99.93	95.78	0.01	99.99	9.09	99.56
MLP	99.77	93.25	0.15	99.85	0.00	97.79
ENN (Eval3)	98.73	98.73	1.24	98.76	63.64	98.23

In terms of accuracy and FPR, the DT offers the best performance, but suffers from poor classification rates on *U2R*. The MLP trained with backpropagation exhibits a similar classification trade-off, but is unable to detect any *U2R* intrusions. Similar classification rates were obtained with the ENNs evolved with evaluation functions 1 and 2, which are biased towards the major class (as are the DT and MLP trained with backpropagation). However, the ENNs evolved with evaluation functions 3–5 obtain the highest TPR, detecting all classes with comparatively high accuracy. It can be observed that these ENNs provide a different classification trade-off, resulting in a higher FPR. An overview of the performance of ENNs for each evaluation function is provided in Table 5.12, which were selected based on their TPR, with preference to ENNs that detect both intrusion classes and obtain a low FPR.

Table 5.12: Overview of classification rates for the ENN for each evaluation function. The best classification rates are highlighted in bold.

Function	Accuracy%	TPR%	FPR%	Normal%	U2R%	R2L%	Topology
Eval1	98.80	0.00	**0.00**	**100**	0.0	0.0	3L, 30 & 100HN
Eval2	**99.77**	81.86	0.01	99.99	0.0	85.84	3L, 120HN
Eval3	98.73	**98.73**	1.24	98.76	63.64	**98.23**	3L, 100HN
Eval4	97.69	97.89	2.30	97.70	**81.82**	**98.23**	3L, 70HN
Eval5	98.21	98.31	1.76	98.24	72.73	97.35	3L, 70HN

5.5 Discussion

The results obtained in this study demonstrate clearly the benefit of evolving the MLPs, as up to 81.82% *U2R* intrusions can be detected, compared with 9.09% for the DT and none with the MLP trained with backpropagation. However, similar poor performance on the minor classes was obtained with the ENN when using evaluation functions 1 and 2, which are also biased towards the major class(es).

In terms of accuracy, the DT and MLP trained with backpropagation do well, since this is what they are trained for. Their bias towards the major class (*Normal*) leads to high TNRs and low FPRs. However, as stated above, the detection of *U2R* is particularly poor, in line with the findings of Bouzida and Cuppens (2006a;b). The findings with the ENN (specifically, the difference in behaviour between biased and unbiased evaluation functions) demonstrate that this is indeed caused by an inability to learn effectively from data with such extreme class imbalance. More specifically, the problem lies with the training algorithm, rather than the classifier itself, since it has been shown that the ENN is indeed capable of obtaining high detection rates on both *U2R* and *R2L* with evaluation functions 3–5, which do not have the same bias towards the major class(es).

MLPs are often used for classification tasks as they are universal approximators (Hornik *et al.* 1989), and the use of backpropagation (or variants thereof) for training is common. As the findings in this investigation demonstrate, backpropagation (and other approaches based on biased error measures) is not appropriate for problems with an extreme class imbalance, if the detection of minor classes is important, as would reasonably be expected in many application domains, including intrusion detection. Using backpropagation in conjunction with cost-sensitive learning may provide a successful method of learning from imbalanced data. That is, the error on each class may be multiplied by a cost matrix that balances out the classes, which is discussed further in Section 5.1 on page 77. However, there are two particular benefits of the evolutionary approach: (1) the flexibility in the composition of the fitness measure, allowing for an arbitrary transformation of classification rates and/or measures of error, which can operate on both class and instance level, and (2) that an entire population of solutions may be created in a single run of the algorithm.

In this investigation, only a single solution from the evolved population of MLPs was selected from each run, based on the best fitness score. However, since the fitness scores are calculated as an aggregate of several separate objectives, two MLPs that obtain the same fitness score may exhibit different classification properties. For example, with Eval5, one MLP may detect 90% *Normal*, 50% *U2R* and 50% *R2L*, whilst another could obtain an identical fitness score by detecting 50% *Normal*, 90% *U2R* and 50% *R2L*. Therefore, in theory, there may be several solutions that obtain high fitness scores that exhibit different classification *trade*-offs, although GAs are known to converge eventually to a single solution without specific mechanisms to promote diversity. This diversity may be useful in cases where the trade-off exhibited by one MLP is unacceptable for a given application scenario.

In order to exploit the potential for producing diverse but effective classifiers, offering varying trade-offs, a single objective approach is neither convenient nor predictable. Indeed, the results from individual trials of a given network configuration show that the evolutionary process can lead to networks with significantly different properties, as the search is non-deterministic and the landscape is potentially complex and deceptive. The problem that needs to be resolved is multi-objective in nature; hence, it is the transformation to a single-objective problem that introduces complications. This is true for all classification problems that cannot be solved trivially (*i.e.,* those that are not readily separable, such that some misclassifications are inevitable). Therefore, in the following chapter, a multi-objective extension of the ENN is proposed that is able to provide the user with a set of solutions that exhibit different classification trade-offs. Furthermore, the potential for classifier combination is investigated as a way to further improve on the performance of the individual classifiers.

CHAPTER 6

Multi-objective evolution of classifier ensembles

Although able to address, to some degree, learning from imbalanced data, the utility of the ENN approach proposed in the previous chapter is limited by converting a naturally multi-objective problem into a single-objective problem. The consequence is that only a single solution is obtained for which we have no control of the performance trade-off among the objectives (classification rates for each class). Eval5, which maximises the sum of the classification rates of each class, is a good example of this. Although high classification rates were obtained on *U2R* and *R2L*, the trade-off is a poor classification rate on *Normal*, resulting in a high FPR. Instead of converting individual classification rates into a single-objective problem, the MLPs can be evolved with a Multi-Objective GA (MOGA) that treats the classification rates of each class as a separate objective, yielding a set of solutions with different performance trade-offs. The extension of GAs to multiple objectives is discussed in Section 6.1.

While an extension to multi-objective optimisation has pragmatic benefits, giving a user a choice between different MLPs that exhibit different classification trade-offs, there is further potential in the approach. MLPs evolved by a MOGA can be seen as a diverse pool of base classifiers that can be combined to form classifier ensembles. Many researchers have demonstrated the success of classifier combination, which has become increasingly popular over the last decade. Moreover, in recent years, evolutionary algorithms have become more established in creating classifier ensembles. A general treatment of classifier combination is given in Section 6.2, followed by a review of current approaches to evolving classifier ensembles in Section 6.3.

MOGAs have been used to create base classifiers, in which the final set of *non-dominated*[1] solutions are combined to form an ensemble (Abbass 2003a, Chandra and Yao 2004; 2006b, Ishibuchi and Nojima 2006). MOGAs have also been used for selection of ensemble members, however, the current approaches share one particular drawback; the same drawback shared by the ENN proposed in the previous chapter: only a single solution is created, for which there is no control of the classification trade-off it will exhibit. This is a general problem with most existing machine learning approaches, which has not been considered in the literature.

Although ensembles have been shown to be capable of outperforming the individual classifiers, they are not guaranteed to do so (Kuncheva 2004, Quinlan 1996). The multi-objective approach taken here to create MLPs, and ensembles thereof, offers a different perspective on this, using concepts of dominance and

[1] Non-dominated solutions refers to solutions with different trade-offs among the objectives, in which one cannot be deemed better than another without domain knowledge. This is considered further in Section 6.1.

Pareto optimality to demonstrate how the selection of base classifiers affects the performance of the resultant ensemble(es). An empirical investigation in this chapter demonstrates how existing approaches that do not consider the trade-off problem are prone to creating solutions with unfruitful performance trade-offs.

The research method is presented in Section 6.4, followed by results and discussion in Sections 6.5 and 6.6, respectively.

6.1 Multi-objective genetic algorithms

Since the first implementation by Schaffer (1985), Genetic Algorithms (GAs) have become well established in the field of multi-objective optimisation, as demonstrated by their diverse application (Coello Coello 1999, Deb 1999, Dimopoulos 2004, Fonseca and Fleming 1993b, Handl *et al.* 2006, Van Veldhuizen and Lamont 2000). In recent years, they have also been used for creating classifier ensembles (reviewed in Section 6.3 on page 110). Section 6.1.1 discusses the motivations for performing multi-objective optimisation. Section 6.1.2 discusses those modifications that need to be made to a GA in order to optimise simultaneously multiple objectives. Several well established algorithms have been proposed in the literature, which are considered in Section 6.1.3. Due to the additional computational expense of MOGAs, parallel implementations are discussed in Section 6.1.4.1.

6.1.1 Motivations for multi-objective optimisation

Early techniques dealt with multiple evaluation criteria by converting multiple objectives (a vector) into one objective (a scalar) (Deb 2001, p. 7). This approach was taken with the ENN, which introduces some complications since single-objective optimisation only produces one solution, whereas a multi-objective problem often has a set of *non-inferior* solutions that represent different trade-offs between the objectives. This trade-off problem was been recognised as early as 1896 by Vilfredo Pareto (1972), which is now referred as to as the concept of a *Pareto Optimum*. The concept is illustrated in Figure 6.1, depicting a maximisation problem with two objectives that does not have a single best solution. There exists an optimum *Pareto front* (in blue), however, with a number of solutions that represent different trade-offs between the objectives. Without domain knowledge, none of the solutions in the Pareto front can be deemed better than another.

Conventional techniques transform multi-objective problems into single-objective optimisation problems, often by using *a priori* knowledge about the problem, or heuristics to guide the algorithm to a single solution (Coello Coello 2000, Deb 2001). The algorithm is run once to find one solution in the Pareto front. More solutions may be found by executing successive runs whilst changing the preference parameters, *e.g.*, weights for each of the objectives. However, according to Deb (2001, p. 75), for some techniques, it is not guaranteed that all solutions in the optimum Pareto front can be found in this manner, especially if the Pareto front is non-convex. Furthermore, Deb (2001, p. 79) also states that most conventional techniques are designed for specific problems. This demands considerable knowledge from the engineer, both about techniques to use and specific knowledge about the problem that needs to be solved. In contrast, a MOGA may sample the Pareto front in a single run without incorporating any domain knowledge about the problem.

6.1.2 Required modifications

Due to the population based approach to optimisation, MOGAs can find a set of solutions on a Pareto front in a single run (Coello Coello 1999, Deb 2001, Fonseca and Fleming 1995). However, due to finite populations, stochastic errors and genetic drift, a GA naturally converges to a single solution (Deb and Goldberg 1989). Therefore, some modifications are necessary to sample an entire Pareto front. First, the

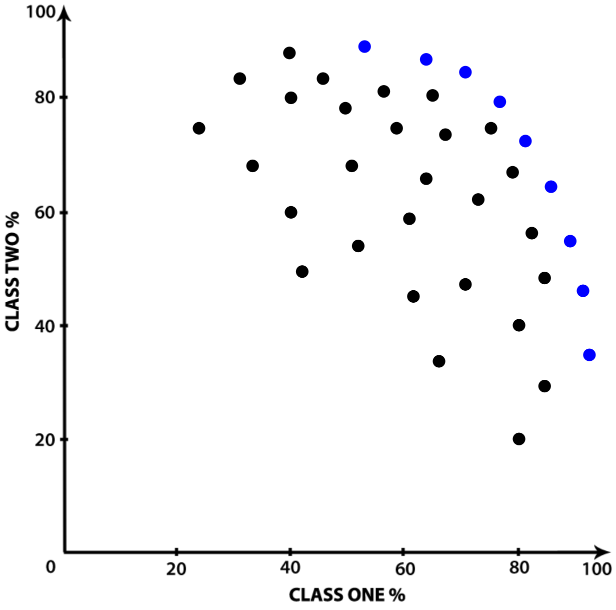


Figure 6.1: An example of a multi-objective problem with two classes, in which the objectives are to maximise the classification rates. From a set of solutions, the ones in blue signify the *Pareto front*.

selection process needs to consider multiple objectives, which may be achieved by transforming fitness scores or modifying the selection algorithm. Second, additional mechanisms are required to distribute samples across the Pareto front. A selection of commonly used techniques are described below.

6.1.2.1 Niching

Traditionally, niching methods encourage the formation of stable sub-populations on multiple optima in multi-modal problems (Deb and Goldberg 1989). In multi-objective optimisation, niching methods are utilised to distribute samples across the Pareto front by discouraging their aggregation. Two niching techniques are described here: *crowding* and *sharing*. These two techniques are representative of fundamentally different approaches: one directly modifies the selection method (crowding), whilst the other transforms the fitness scores (sharing) prior to selection.

Crowding was proposed by Holland (1992), and aims to identify where several individuals dominate an environmental niche. This situation results in lower life span and birth rates, and is mitigated by incorporating a replacement strategy. A commonly used strategy was suggested by DeJong, in which a new offspring replaces the individual that is the most similar to itself (Coello Coello 2000). Crowding is closely related to mating restrictions, which is discussed further below.

Goldberg and Richardson (1987) propose a niching technique called *sharing*, which modifies the fitness of individuals in the niches according to the niche size; the larger the niche, the less fit the individuals will appear. A parameter, σ_{share}, signifies a distance threshold for determining if an individual is considered to be a part of a particular niche. It is usually implemented such that, for a given individual, a fitness penalty is applied for all others that are within σ_{share} distance, with the penalty typically decreasing linearly with increasing distance. Hence, tight clusters of samples are heavily penalised and representatives are not then as likely to be selected for reproduction. Deb and Goldberg (1989) present empirical results that show benefits of using sharing for multi-modal landscapes. It has also been a popular choice in MOGAs (Coello Coello 1999; 2005).

Recombination of individuals between niches can lead to a high proportion of lethals. Goldberg and

Richardson (1987) suggest a combination of sharing with mating restrictions, which was shown by Deb and Goldberg (1989) to improve the performance of sharing. The mating restriction they suggest only allow individuals to reproduce if their phenotypic distance is within a certain threshold. Eshelman and Schaffer (1991) took the opposite approach by restricting mating to dissimilar individuals, suggesting that this is beneficial to prevent premature convergence. However, this observation was made in the context of single-objective optimisation. For multi-objective optimisation, it is predominately mating restrictions between different niches that is of interest.

6.1.2.2 Dominance

Dominance is an evaluation concept introduced by Goldberg (1989, pp. 197–201) to handle multiple objectives. Individuals need to be evaluated with equal weight to every objective, which is achieved by consider dominance. Individual x *dominates* individual y if the following conditions are fulfilled:

1. x is no worse than y on all objectives, and
2. x is better than y on at least one objective

In the same way, individuals can be identified as *non-dominated* if the individual is not dominated by other individuals. The evaluation process is illustrated with an example in Figure 6.2. The concept can be applied in two ways, either by modifying the selection (*e.g.*, dominance based tournament selection) or fitness transformation by dominance ranking. Regarding the latter, all the individuals in a population may be sorted into non-dominated sets, as in seen in Figure 6.2d. In a practical scenario, there may be hundreds of such non-dominated sets, all of which can be assigned a rank. The rank is typically a whole number, which is incremented for each set to reflect how many sets dominate it.

Both ranking and the dominance concept can be used directly in selection and replacement. For example, for dominance based tournament selection, two individuals may be selected from the population as a potential parent, and the dominant individual selected. Similarly, if ranks have been calculated, the individual with the lowest rank value is selected.

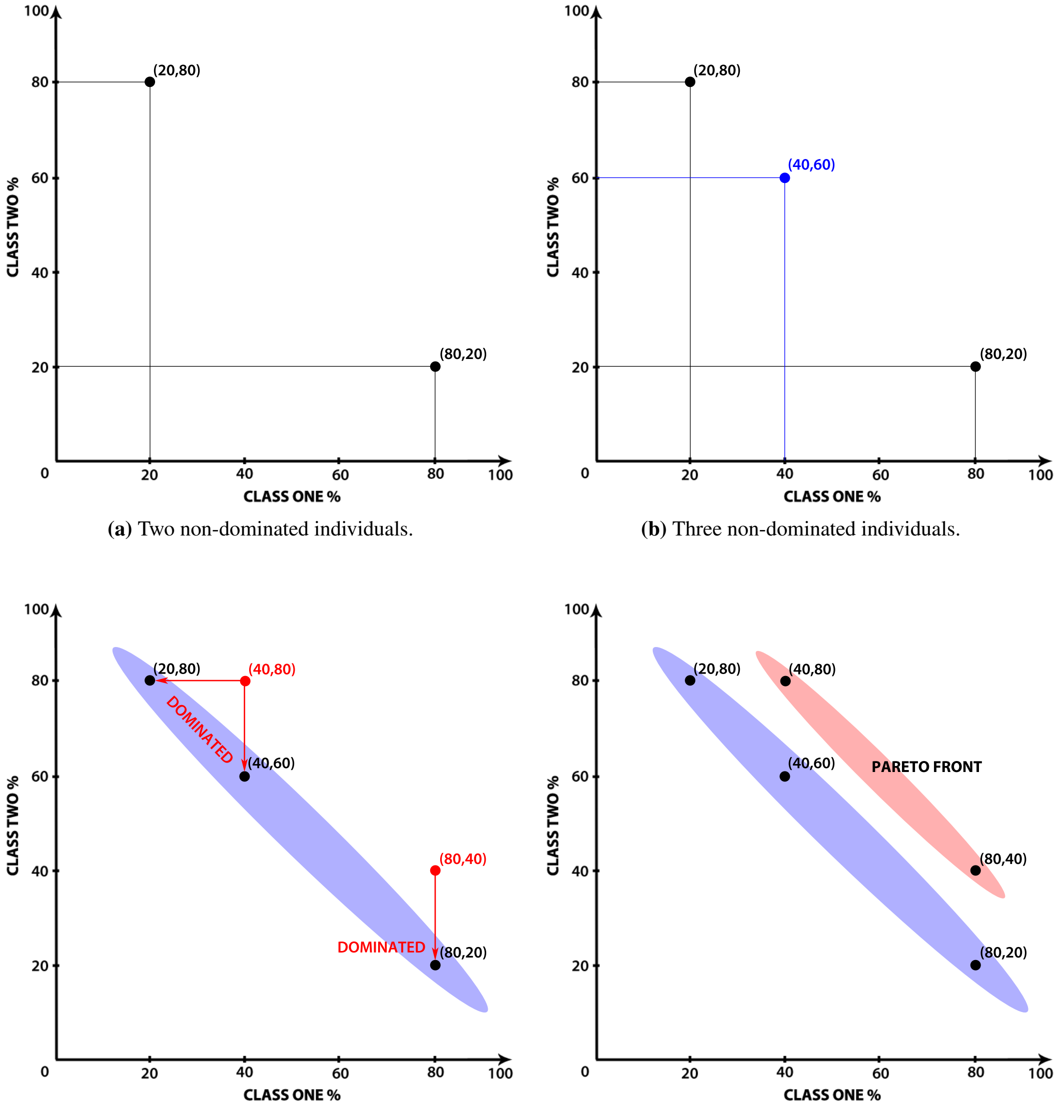
The potential problem with these approaches is that two non-dominated individuals may be selected (therefore with the same rank). This is a case in which niching, such as sharing, is useful, in which the individual in the less crowded region can be selected to prevent the MOGA from converging on a single solution.

6.1.2.3 Archive function

A mechanism that has been proposed in more recent MOGA algorithms is an *archive* of all non-dominated solutions discovered since the start of the algorithm (Deb *et al.* 2000, Zitzler and Thiele 1999). An archive may be necessary for some problems to be able to sufficiently sample the Pareto front when practical population sizes are not large enough to do so. Furthermore, an archive may provide a mechanism to reintroduce important genetic material that may become lost in a population as the search progresses in a particular direction.

6.1.3 Algorithms

The most common MOGAs are described here in chronological order according to their development. For further information, refer to surveys by Coello Coello (1999, 2000, 2005) and a more comprehensive text by Deb (2001).



(a) Two non-dominated individuals.

(b) Three non-dominated individuals.

(c) Two individuals that dominate the previous three.

(d) Two sets of non-dominated individuals.

Figure 6.2: An illustration of dominated and non-dominated sets of individuals. Relating this to the application here, each individual represents the classification rates of MLPs on two objectives, *intrusion* and *normal*.

6.1.3.1 VEGA

The first MOGA was implemented by Schaffer (1985). This algorithm, called the Vector Evaluated Genetic Algorithm (VEGA), incorporates a modified selection process to cope with multiple evaluation criteria. The population is divided into *n* sets, equal to the number of objectives, where selection in each set is based on a single objective. Mating restrictions are employed, constraining recombination to individuals within the same set. By means of pair-wise comparison of all individuals, one can identify those that are dominated and flag them as inferior. At the end of a reproduction cycle, the set of non-inferior individuals represents the current best Pareto front.

There is, however, a drawback with this technique that can prevent the location of the Pareto front. Due to the selection process, individuals that excel in one dimension are favoured over individuals with a balanced performance across all dimensions. As a result, the VEGA tends to converge to individually best solutions only (Deb 2001, p. 173).

6.1.3.2 MOGA

Fonseca and Fleming (1993a) proposed the Multi-Objective GA (MOGA), which incorporates both niching and the concept of dominance with rank based fitness assignment. Individuals that are non-dominated with respect to each other are grouped into sets. All individuals in each set are assigned an identical rank, starting at 1. Consecutive sets that are dominated by the prior set are assigned an incremented rank (2, 3, *etc*.). Diversity is maintained within each set of non-dominated solutions by use of sharing. Another novelty in this algorithm is a dynamic update mechanism for σ_{share}. This approach has one major drawback: due to large niche counts, the shared fitness may be small. This affects the selection pressure of all solutions of higher rank, which leads to slow convergence. In turn, this can prevent the algorithm from finding the optimum Pareto front (Deb 2001, p. 196).

6.1.3.3 NSGA

Srinivas and Deb (1994) proposed the Non-dominated Sorting GA (NSGA), which, like MOGA, incorporates both the concept of dominance and sharing. Before selection, the population is ranked by giving all non-dominated solutions a fitness value proportional to the population size. All solutions in the highest ranking set are given a fitness value equal to the population size, then progressively lower fitness is assigned to individuals in sets of lower rank. The NSGA then uses sharing to maintain diversity among the individuals, by adjusting the fitness to make less crowded regions more fit. Unlike MOGA, NSGA does not incorporate any dynamic niche updating strategy, which makes performance dependent on the value of σ_{share}.

6.1.3.4 NPGA

Horn *et al*. (1994) proposed a Niched Pareto GA (NPGA), which is also based on the dominance concept and uses sharing. It differs from earlier algorithms in adopting a modified tournament selection rather than proportional selection. The selection strategy works as follows: two individuals are selected together with a random comparison set. The selected individuals are compared with those in the comparison set according to dominance. If one individual is not dominated by the set, whilst the other is dominated, the former is selected. If both are either dominated or non-dominated, then the individual with the lowest niche count wins the tournament.

6.1.3.5 SPEA

Towards the end of the 1990s, NSGA was found to be the best performing algorithm, when compared with VEGA, NPGA, NSGA, a weighted sum, and a random approach (Zitzler and Thiele 1998). Since research at this point had established several GAs that were capable of performing multi-objective optimisation, efforts were then directed at developing algorithms with improved performance. Coello Coello (2005) describes these as second generation algorithms. For example, Zitzler *et al*. (1999, 2000) proposed one of the first *elitist* MOGAs, the Strength Pareto Evolutionary Algorithm (SPEA). Elitism was found to improve significantly the convergence properties of the MOGAs, which was then also incorporated into a new version of NSGA: NSGA-II, which is discussed further below.

SPEA is one of the first MOGAs to maintain an *archive* of non-dominated solutions from the start of the algorithm, which was also incorporated in a multi-objective evolution strategy algorithm proposed by Knowles and Corne (2000). The archive is external to the main population, but takes part in the fitness calculation. Since the archive can grow very large for some problems, Zitzler and Thiele (1999) propose a pruning mechanism to keep the size of the archive within a certain limit, to prevent the selection pressure from dropping.

6.1.3.6 NSGA-II

Deb *et al.* (2000) propose a fast, elitist, version of the NSGA, called NSGA-II, which they found to outperform SPEA on five difficult test problems. Unlike NSGA, NSGA-II preserves diversity among non-dominated solutions by use of crowding (instead of sharing). Another major difference is that the NSGA-II uses tournament selection.

NSGA-II is a generational algorithm that utilises the concept of dominance to determine the new population after each generation (at which point selection and recombination has taken place to produce an offspring population of the same size as the old population). Before classifying individuals according to the dominance concept, parent and offspring populations of size n are combined to form a global population of size $2n$. Non-dominated sorting is then applied to this global population, which identifies sets of non-dominated individuals, as illustrated in Figure 6.2d on page 100. A new population of size n is then created from the sets according to their 'rank'. If all individuals of a set cannot fit into the population, niching is used to select individuals in the least crowded region of the set. Consecutive sets, if any, are discarded.

6.1.3.7 SPEA2

As mentioned above, SPEA required a pruning mechanism to prevent the archive of non-dominated solutions from growing too large. In an improved version of the algorithm, SPEA2, Zitzler *et al.* (2002) propose a new pruning mechanism that ensures that boundary solutions are retained in the archive. SPEA2 also incorporates a more fine grained fitness function, which takes into account the number of individuals that dominate a given individual, and how many it dominates. Furthermore, the fitness function includes density information based on a k-NN algorithm.

Zitzler *et al.* (2002) found that SPEA2 performed well in comparison with NSGA-II, and their findings indicate that SPEA2 performs better in high dimensional search spaces. However, their test suite was limited.

6.1.4 Parallel MOGAs

The additional mechanisms in the MOGAs to deal with multiple objectives, such as non-dominated sorting, are time consuming. Furthermore, since many real world problems are complex, with many objectives, larger population sizes are required to sample the Pareto front, which can be computationally intensive (Deb *et al.* 2003). In such cases, serial MOGAs may be unacceptably slow. Fortunately, GAs are well suited to parallelising (Cantú-Paz 1998), and several successful models have been proposed in the literature. These are presented in Section 6.1.4.1, followed by a discussion of specific parallel MOGA implementations in Section 6.1.4.2.

The principles of parallel programming are not covered here. Readers are referred to texts by Andrews (2000), Bal (1990), Foster (1995), Hwang and Briggs (1986), and Kumar *et al.* (1994). Further information on parallel GAs can be found in Cantú-Paz (1998; 2001), Alba and Troya (1999), and Talbi *et al.* (2008).

6.1.4.1 Parallel models

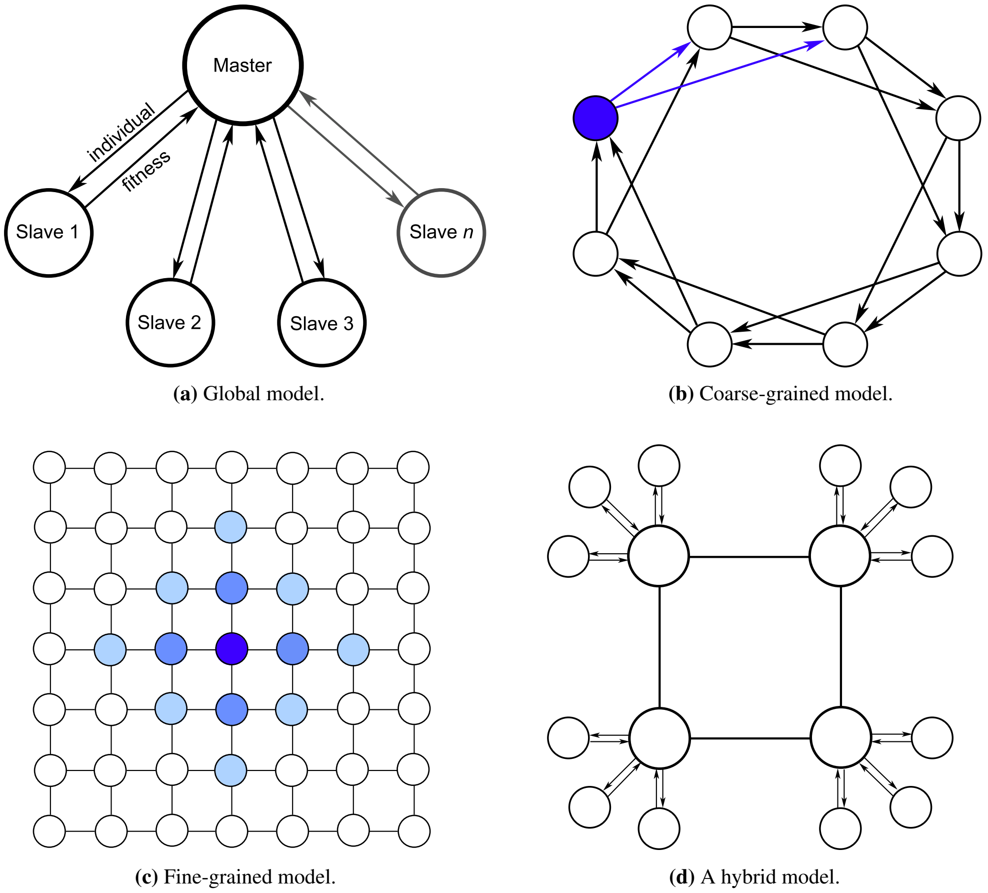
Alba and Troya (1999) identified four classes of parallel implementation (as illustrated in Figure 6.3):

Global model: A global population is maintained on one node, whilst fitness evaluations are farmed out to slave nodes. Hence, this is also known as a farming model (Kohlmorgen 1995) or master-slave model (Cantú-Paz 1998). See Figure 6.3a.

Coarse-grained model: This model consists of multiple sub-populations, referred to as *demes*, which are executed on different nodes. As illustrated in Figure 6.3b, demes communicate by migrating individuals between them, following various topologies. Hence, this is also referred to as an island model (Gordon and Whitley 1993), migration model (Kohlmorgen 1995), or, more generally, a distributed GA (Cantú-Paz 1998).

Fine-grained model: While the two models above function well with few nodes, this model exploits massively parallel hardware (Hart *et al.* 1996). The model sustains one population, where each individual is assigned to a single node. Different selection and replacement mechanisms are implemented, operating within local neighbourhoods, as illustrated in Figure 6.3c. Hence, this approach is also known as a diffusion model.

Hybrid model: Also known as hierarchical models, which are compositions of the three models discussed above. For example, a coarse-grained model at the top level, and a global model at the lower level, which is illustrated in Figure 6.3d. This is particularly desirable since Cantú-Paz and Goldberg (1997) found that there was an optimum number of demes in a coarse-grained algorithm. Such a hybrid model may allow better exploitation of some configurations of parallel hardware.



(a) Global model.

(b) Coarse-grained model.

(c) Fine-grained model.

(d) A hybrid model.

Figure 6.3: Parallel models.

The global model is considered the simplest parallel model, since the semantics of the GA remain unchanged. Consequently, as Cohoon *et al.* (1987) state, it can only offer hardware acceleration, whereas other models have been found to offer potential search benefits (Gordon and Whitley 1993). The global model may provide near linear speedup if the fitness function is sufficiently computationally intensive in relation to communication overheads. As the number of slave nodes increases, the communication overhead reduces efficiency (Abramson and Abela 1993).

Similarly to the global model, coarse-grained algorithms are widely applicable, because they can be readily implemented on common hardware (Tomassini 1999). The coarse-grained implementation does, however, introduce several additional parameters which need to be configured:

- The topology (Figure 6.3b illustrates the common ring topology).
- The number of demes.
- Connectivity between the demes.
- The migration interval.
- The number of migrants.
- Migrant policy (selection and replacement).

Despite the additional complexity in the model, it comes with a significant benefit: it is possible to obtain better search performance compared with a serial GA (Doorly *et al.* 1996, Gordon and Whitley 1993). Therefore, it may even be desirable to implement a coarse-grained model on a serial machine to solve difficult problems.

Despite theoretical and empirical research into the properties of fine-grained PGAs, there are few practical applications reported in the literature. This may be due to limited availability of massively parallel hardware and the rise in popularity of coarse-grained platforms such as clusters. However, Logar *et al.* (1992) report success in parallelising a fine-grained algorithm onto a SIMD (Single Instructions, Multiple Data) architecture, and Cantú-Paz (1998) discusses a few studies from the early 1990's.

6.1.4.2 Implementations for multi-objective optimisation

Despite the success of parallelising single-objective GAs, there has been limited research on parallelising MOGAs (Van Veldhuizen *et al.* 2002). Existing papers indicate that parallelising multi-objective GAs is not as straightforward as parallelising single-objective GAs due to different evaluation and diversity mechanisms. Hence, the global model is the simplest to adopt, as it typically only distributes fitness evaluations, and, therefore, avoids complications related to the other models. In a recent survey, Talbi *et al.* (2008) contend that the global model is widely used and is very successful when the fitness evaluation is time consuming.

According to Rowe *et al.* (1996), a fine-grained parallel structure is ideal for multi-objective optimisation as it has a natural niching behaviour due to the spatially distributed nature of the population. Since non-dominated sorting of individuals is a local process, the overhead of calculating ranks for the entire population is avoided. Their parallel implementation improved search performance compared with a serial GA with sharing. One drawback of the fine-grained approach, however, is that individuals in the same niche may never compete in the selection process if they are physically separated. This may, therefore, conflict with mechanisms such as mating restrictions, and may lead to sustaining low fitness niches.

Hiroyasu *et al.* (1999) used a coarse-grained (island) model for multi-objective optimisation. To preserve diversity, sharing is applied to the sets of Pareto solutions when they are too crowded. Sharing is performed both locally, in each sub-population, and globally, to shorten calculation time. They reported high accuracy of the search, but the algorithm is computationally intensive because of the sharing mechanism. Moreover, Sastry *et al.* (2005) conducted experiments on additively separable problems and found that failures in niching mechanisms lead to exponential scale-up in computation time. This failure occurs when there is a large number of solutions in a Pareto front. A common island model has, however, been applied more recently to generate classifier ensembles (Duell *et al.* 2006), which is considered further in Section 6.3 on page 110.

A few researchers have attempted to utilise the parallel structures by assigning parts of the search space to different processors. Deb *et al.* (2003) implemented an allocation plan that assigns a particular part of the Pareto front to each processor. All processors still cover the entire search space, but are guided by the allocation plan. After a run, solutions from all processors are aggregated to form a diverse set of solutions. It was observed that multiple processors helped to sample a diverse set of the optimum Pareto front. Further work by Branke *et al.* (2004) included explicit constraints to allocate areas of the fitness space to the processors. The results were promising for two dimensions, but were not when a third dimension was added. Both studies restricted test problems to a maximum of three dimensions. Generalising such techniques to arbitrary dimensions is non-trivial.

6.2 Classifier combination

Throughout this chapter, the terms 'ensemble', 'classifier combination' and 'combination of classifiers' can be considered synonymous, and are used interchangeably. Section 6.2.1 introduces classifier combination, and establishes the potential benefits over monolithic classifiers. A taxonomy of classifier combination approaches is presented in Section 6.2.2. According to this taxonomy, Sections 6.2.3 to 6.2.6 further discuss the different approaches to classifier combination. Section 6.2.7 discusses the importance of diversity among the classifiers comprising an ensemble.

6.2.1 Motivations

The main motivation for classifier combination is stated by (Kuncheva 2004, p. 295):

> *"If we have a perfect classifier that makes no errors, then we do not need an ensemble. If, however, the classifier does make errors, then we seek to compliment it with another classifier, which makes error on different objects."*

Further, Dietterich (2000b) offers three specific reasons why classifier combinations can be beneficial:

1. ***Statistical***: if the amount of training data is insufficient to model the hypothesis space with one classifier, then the combined knowledge of an ensemble of classifiers may reach more accurate predictions.

2. ***Computational***: algorithms can become stuck in local optima and it may be computationally expensive to find the global optimum. Instead, it may be better to execute several local search algorithms from different starting points and combine these.

3. ***Representational***: the optimal classifier may not be found, because the technique is incapable of modelling the hypothesis space to fit the true function of the problem. Kuncheva (2004, p. 103) offers a simple example to demonstrate this: a linear classifier cannot perform nonlinear classification, however, a combination of linear classifiers can.

Garcia-Pedrajas *et al.* (2005) provide another specific example of the pragmatic benefit of performing classifier combination:

> *"Although theoretically, a single neural network with a sufficient number of neurons in the hidden layer would suffice to solve any problem, in practice many real-world problems are too hard to construct the appropriate network that solve them. In such problems, neural network ensembles are a successful alternative."* Garcia-Pedrajas *et al.* (2005)

Several authors report that ensembles often outperform the individual best base classifier (Bauer and Kohavi 1999, Brown *et al.* 2005, Dietterich 2000a, Tan and Gilbert 2003). However, Kuncheva (2004, pp. 103–104) notes that classifier combinations are not guaranteed to do so. For example, Quinlan (1996) investigated the performance of boosting (see Section 6.2.6) and observed that it did not improve the performance on all data sets. However, this is the nature of this domain; there is no single best classifier/approach (Kuncheva 2004, p. 229) (as established for optimisation in the *no free lunch theorems* (Wolpert and Macready 1997)).

6.2.2 Taxonomy

Since research on classifier combination is evolving continuously, there is no existing taxonomy that covers every aspect of the existing approaches. The taxonomy adopted and presented here stems from Kuncheva (2004, p. 105), and additional aspects from Valetini and Masulli (2002) are discussed:

A) *Combination level*: considers how classifiers are combined.

B) *Classifier level*: determining which base classifiers to combine.

C) *Feature level*: using different feature subsets to train the base classifiers.

D) *Data level*: using different data subsets to train the base classifiers.

Kuncheva (2004, p. 106) further describes two approaches to developing ensembles.

1. ***Decision optimisation***: *"methods to choose and optimize the combiner for a fixed ensemble of base classifiers*" (corresponds to level A)

2. ***Coverage optimisation***: *"methods for creating diverse base classifiers assuming a fixed combiner"* (corresponds to levels B, C and D)

These are similar to the main approaches according to Valetini and Masulli (2002), which they refer to as *non-generative* and *generative methods*. The non-generative methods are similar to decision optimisation, in which the goal is to combine a set of existing base classifiers. Generative methods, on the other hand, actively generate classifiers in an attempt to increase the accuracy and diversity of the ensemble.

Valetini and Masulli (2002) extend the coverage optimisation group to encompass *mixture of experts* (discussed further below), *output coding methods* (Dietterich 2000b, Masulli and Valetini 2000), *test and select methods*, and *randomized ensemble methods*. Output coding methods transform multi-class problems into binary classification problems, which are then combined in the ensemble to solve the multi-class problem. In test and select methods, base classifiers are added to the ensemble successively. The most simple approach for this is a greedy method (Perrone and Cooper 1993), in which a new classifier is added to the ensemble only if the accuracy is improved. The randomised ensemble methods create a diverse set of base classifiers by randomly initialising the configuration parameters of the classifiers before learning, *e.g.*, for ANNs, this may include the weights, learning rate and momentum.

Each of the levels in the taxonomy of Kuncheva are discussed further in their respective sections below.

6.2.3 Combination level

Kuncheva (2004, p. 106) considers two main approaches to combining classifiers, which can be considered a subgroup of decision/coverage optimisation:

1. ***Fusion***: also referred to as competitive classifiers, ensemble approach or multiple topology

    - Each technique is trained on the complete feature set.

    - Typically, voting or an average is used to determine the ensemble output.

2. ***Selection***: also referred to as cooperative classifiers, modular approach or hybrid topology
    - Each classifier is trained on different feature sets.
    - Usually one classifier is chosen to give the ensemble output.

Fusion approaches generally fall under coverage optimisation, whilst selection approaches fall under decision optimisation. In addition to the fusion and selection models, there exists a *mixture of experts* model (Egmont-Peterson *et al.* 1999, Gama and Brazdil 2000), which trains both the base classifiers and the combination method at the same time.

Two additional components are incorporated into a mixture of experts model: a gating network and a selector. The gating network receives the same input vectors as the classifiers in the ensemble, but its function is to calculate probabilities for each classifier as to how competent they are to classify the given input. These probabilities, accompanied by the predictions of each of the classifiers, are passed on to the selector, which then determines the final output. This final classification can be determined in three ways (Kuncheva 2004, p 201):

1. Stochastic selection according to the distribution of the calculated probabilities.
2. Winner-takes-all; the maximum probability leads to selection.
3. Weights; used for soft output.

Fusion is the simpler approach, and, thus, is attractive. A simple, but effective, approach to deciding the output label (classification) of an input vector is to choose the label that the majority of the classifiers in the ensemble predict, referred to as a majority vote combiner. This approach assumes that all the classifiers are equally important, which they may not be. In contrast, in the selection approach, generally only one classifier is chosen to label an input vector. However, selection requires that the ensemble is further trained to obtain mechanisms for deciding which classifier should be chosen to label a given input. In this sense, the selection model is similar to the mixture of experts model.

6.2.4 Classifier level

Classifier combination on this level has been considered for intrusion detection due to observations in the literature that different classifiers perform well on different classes of intrusion (Anuar *et al.* 2008, Gharibian and Ghorbani 2007, Pan *et al.* 2003, Peddabachigari *et al.* 2007, Sabhnani and Serpen 2003). For example, Sabhnani and Serpen (2003) combined three different machine learning techniques, namely an ANN, *k*-means clustering and a Gaussian classifier, as these obtained the highest accuracies on the different classes of intrusion. However, they do not provide further implementation details about how the classifiers are trained, nor how the ensemble output is determined. According to their model, all classifiers process the same input vectors, but it is not clear how an output is determined if, for example, two classifiers predict that the input vector is an instance of the class of intrusion they have been assigned to detect. Nevertheless, Sabhnani and Serpen (2003) report that the classifier combination approach improved the classification rates.

The hierarchical hybrid system proposed by Xiang *et al.* (2008), as discussed in Section 4.6 on page 73, can be categorised within this level. This system, comprised of a DT and an AutoClass classifier, was able to achieve a higher true positive rate than previously reported in the literature on the original training and test sets of the KDD Cup ’99 data set. However, this was at the expense of a higher false positive rate.

In the general literature on classifier combination, Kuncheva (2004, p. 105) states that there is no evidence supporting the use of base classifiers of the same type or different types. As considered in the previous chapter of this thesis, there are discrepancies in the results reported in the literature on intrusion detection. Although individual studies report improvements when specific classifiers are combined, the same combination of classifiers may be poor in another application.

6.2.5 Feature level

The motivations for using different feature sets (level C) is to improve the computational efficiency of the ensemble and to increase the accuracy (Kuncheva 2004, p 237). A particular benefit of this approach is that it alleviates the curse of dimensionality (Friedman 1997). There are several approaches to feature selection for ensemble creation; one popular approach is the *random subspace* method (Ho 1998b;a). Such a random feature subspace method is also used to train DTs in Random Forests (RFs), which is a classifier combination approach that is mainly associated with the data level, as discussed in the following section. Other methods include Principal Component Analysis and Genetic Algorithms. Refer to (Valentini and Masulli 2002) for further information on the different feature selection approaches.

6.2.6 Data level

Some of the most popular classifier combination methods utilise the data level. Three common approaches are described here: bagging, boosting and random forest (RF).

Bagging (**b**ootstrap **agg**regat**ing**), proposed by Breiman (1996), relies heavily on the instability (sensitivity to configuration and/or training data) of the classifiers. Bagging follows a fusion approach, using a majority vote combiner to determine the output of the ensemble. Each of the classifiers is trained on different data. Ideally, the data should be randomly sampled from the original training set. However, this is not always possible due to the size of the data set. Therefore, the different training subsets are sampled with replacement (bootstrap replicates) from the original training set.

Another popular approach is boosting (Schapire 1990, Freund 1995, Freund and Schapire 1997), which Kuncheva (2004, p. 221) believes is the most rapidly growing subarea of classifier combination research. In boosting, the ensembles are populated one classifier at the time; each classifier is trained on selective data from the original training set. For the first base classifier, the data is selected uniformly. For successive classifiers, the sampling distribution is continuously updated so that instances that are more difficult to classify are selected more often than those that are easy to classify. The classification outputs of the ensemble are determined according to weighted votes, which are based on the accuracy of the classifiers.

AdaBoost (**Ada**ptive **Boost**ing) is perhaps the most well known boosting algorithm, which, in its original form, deals only with binary classification (Freund and Schapire 1997). However, AdaBoost.M1 and AdaBoost.M2 are able to deal with multi-class classification (Freund and Schapire 1997), and AdaBoostR has been developed for regression problems (Drucker 1997). AdaBoost can use two approaches to building the ensemble. The first, *resampling*, is the approach described above. The second, *reweighting*, *"...assume*[s] *that the base classifiers can directly use the probabilities on* [the data set] *as weights. No sampling is needed in this case, so the algorithm becomes completely deterministic"* (Kuncheva 2004, p. 215). Neither approach has been shown to be more beneficial than the other (Breiman 1998, Freund and Schapire 1998). Quinlan (1996) has found that AdaBoost is largely successful due to overfitting the training data, although it does not improve the performance for all data sets in that investigation.

Dietterich (2000a) compared empirically the performance of three methods of creating ensembles of DTs: bagging, boosting and a simple randomisation method. The latter method introduces some randomness into the configuration of the DTs, intended to ensure diversity among the different base classifiers. The results indicate that bagging is the most successful approach when the classification data is noisy. However, on noise-free data, boosting leads to the highest accuracy, and randomisation generally outperforms bagging.

Random Forest (RF) (Breiman 2001) is a version of bagging that has received recently much attention in the research community. As the name suggests, RFs are comprised of DTs. Similarly to the basic principle of bagging, each DT is trained on different data and they are combined by a majority vote. The DTs may also be trained on different feature sets. The performance of RF is comparable to AdaBoost, but is more robust to noise (Breiman 2001), which is the same observation Dietterich (2000a) made regarding bagging.

In a more recent study by García-Pedrajas *et al.* (2007), an approach based on the notions of boosting and the random subspace method is proposed, which is less sensitive to noise. However, the approach is still more sensitive to noise than bagging. Another benefit of both RF and bagging, is that they are parallel in nature in both the training and classification stages, whilst the boosting algorithms are sequential.

6.2.7 Importance of diversity

The term 'diversity' in the context of classifier combination does not just imply that the base classifiers should be different; it also implies that the base classifiers make errors on different instances (Brown *et al.* 2005, Hansen and Salamon 1990, Kuncheva 2004, p. 295). This may be referred to as negative correlation (Yao and Islam 2008). In contrast, if the classifiers make errors on the same cases, they are positively correlated, and their combination would not yield any improvement. As initially stated in Section 6.2.1, the goal is to obtain a set of classifiers that complement each other. To obtain diversity, it is fruitful to use unstable base classifiers, such as DTs and ANNs, which can produce significant differences in the models when the training data and/or configuration parameters are changed. Interestingly, diversity is so important that weakening the base classifiers can be a successful approach to building ensembles (Kuncheva 2004, p 295).

Two approaches can be considered for producing diverse base classifiers: *explicit* and *implicit* (Brown *et al.* 2005, Tang *et al.* 2006). Implicit approaches do not use directly diversity as a measure in the process of producing the ensemble (or the base classifiers), whilst explicit approaches do. Kuncheva (2004, p 295) notes that *"trying to measure diversity and use it explicitly in the process of building the ensemble does not share the success of the implicit methodologies"*. There are many metrics of diversity (Kuncheva and Whitaker 2003) and approaches to producing a diverse set of base classifiers. However, they are still not fully understood.

Kuncheva and Whitaker (2003) undertook a comprehensive empirical investigation attempting to answer the following five questions:

1. How can diversity be defined and measured?
2. How do diversity measures relate to each other?
3. How do diversity measures relate to the ensemble accuracy?
4. Can one measure of diversity be determined as 'best' in terms of minimising the error of an ensemble?
5. How can diversity measures be used in designing a classifier ensemble?

They examined ten existing diversity measures (for binary classification), four of which were pairwise measures and six non-pairwise. Most of their observations are inconsistent, making it difficult to draw any accurate conclusions; *"... the notion of diversity is not clear-cut"* (Kuncheva and Whitaker 2003). With respect to question four and five, Kuncheva and Whitaker state that any conclusions are speculative, and even question the importance of measuring the diversity in the ensemble. They conclude that diversity is important, but the use of direct measures in the process of building ensembles remains an open issue.

Brown *et al.* (2005) propose a taxonomy of diversity creation methods, which considers three main groups:

1. ***Starting point in hypothesis space:*** for ANNs, this would be randomly initialising the weights. This, they state, is the least effective method.

2. ***Set of accessible hypotheses:*** this includes manipulation of training data and architectures (*e.g.*, nodes in the hidden layer(s) of MLPs).

3. ***Traversal of hypothesis space:*** this relates to the training process, *e.g.*, penalty methods (negative correlation learning) and evolutionary methods, both of which are discussed in Section 6.3.

Brown *et al.* accept that the taxonomy may not be complete. For example, their taxonomy does not explicitly include approaches based on feature selection. Nevertheless, it gives a foundation for understanding the different approaches to creating diverse ensembles. Further work on this aspect of classifier combination has been conducted by Tang *et al.* (2006), aiming to establish a better understanding of the diversity mechanisms. They consider the combination of classifiers to be a more challenging task than producing diverse base classifiers. Ruta and Gabrys (2001a, 2005) have conducted thorough empirical investigations of diversity metrics used in fusion ensembles, using a majority vote combiner. They demonstrate that the correlation between the diversity measures and the combiner are important. However, they also found that using the error of the majority voting combiner as a selection criterion lead to the highest accuracy of the ensembles.

As Brown *et al.* (2005) states, ensemble approaches to classification and regression problems have very different theoretical foundations. The latter has a well established theoretical basis, which is not the case for classification. This section has discussed several studies on diversity in classifier ensembles, but one particular limitation of the investigations is that they have only considered binary classification problems. Furthermore, there are uncertainties and inconsistencies in the empirical results reported in the studies examining diversity mechanisms, which some of the authors accept (*e.g.*, Kuncheva and Whitaker (2003)). For example, Tang *et al.* (2006) promote using diversity measures directly when creating base classifiers, since methods that utilise the data level may not produce sufficiently diverse classifiers. Moreover, they argue that it is unimportant which diversity metric that is used. This, however, contradicts the observations of Kuncheva (2004, p. 295), claiming that implicit methods are more effective. Nevertheless, the general consent is that diversity is important in the classifiers ensembles, regardless of how it is achieved.

6.3 Evolving classifier ensembles

Evolving classifiers for an ensemble was first considered by Yao and Liu (1996), who found that the aggregated knowledge of the population was important, leading to better results compared with the single best individual. There are three ways in which evolutionary algorithms have been used in classifier combination:

1. Evolving the base classifiers for the ensemble.

2. Evolving the combination of a set of existing base classifiers.

3. Evolving the base classifiers and ensemble concurrently.

The first approach may assume a fixed combiner, such as majority vote, and the evolutionary process is only used to produce a set of diverse base classifiers. Other methods may be used to determine which base classifiers are used in the ensemble and how they are combined. The second approach assumes an existing set of base classifiers and uses an evolutionary algorithm to optimise the combination of them.

These two approaches can be seen as two independent operations, whilst the third approach performs both concurrently. The three approaches are discussed further in Sections 6.3.1 to 6.3.3.

6.3.1 Evolving base classifiers

One notion that underpins the evolution of base classifiers is the reduction in the complexity of a problem by breaking it down into several modules that specialise in solving particular parts of the problem. This is referred to as modularisation (Yao and Islam 2008), which is at the heart of classifier combination. There are several ways in which this can be achieved with evolutionary algorithms. For example, Darwen and Yao (1997) propose using speciation to achieve automatic modularisation. Speciation is synonymous with niching methods such as fitness sharing and crowding (Deb and Goldberg 1989, Section 6.1.2 on page 97), which are modifications to the GA to help prevent the population from converging on a single solution.

Darwen and Yao (1997) apply speciation to evolve rule based systems (RBSs) for playing prisoner's dilemma games, in which each species in the population is a RBS representing a particular strategy. A gating algorithm is then employed to make use of the species in an integrated system. Darwen and Yao (1997) adopted the majority vote combiner, which performed well, giving a *"versatile response against unseen opponents"*.

Although Darwen and Yao evolved RBS for game strategies, the approach can be used to evolve other base classifiers for classifier combination (Yao and Islam 2008). Speciation has been adopted in recent studies, which propose modifications to the approach of Darwen and Yao (1997). For example, in the approach proposed by Ando (2007) (to evolve an ensemble of ANNs), the similarity measure that is used to determine the species is changed. Ando (2007) argues that similarity measures such as Hamming and Euclidean distances can be inappropriate as they may not represent properly the relationship between the genotype and phenotype. However, they do contend that it is difficult to determine an appropriate measure when the genotypes represent a structure, such as an ANN. They propose using probabilistic distributions to define the species, based on *minority detection* (Ando and Suzuki 2006), which Ando (2007) describes as *"a method for identifying a small subset whose distribution has a significant divergence from that of the rest of the data set"*. In another recent application, using explicit fitness sharing, Kim and Cho (2008) adopt two additional measures, Pearson correlation, and Kullback–Leibler entropy distance, which yielded more diverse classifiers.

It has become common to adopt explicit diversity measures, although the literature is inconclusive about their success. Negative Correlation Learning (NCL) (Liu and Yao 1998, Liu *et al.* 2000) is a popular algorithm that incorporates explicit fitness sharing, and a correlation penalty in the fitness function. NCL performs *simultaneous* evolution of the ensemble, and is, therefore, discussed further in Section 6.3.3.1. The ADDEMUP algorithm (Opitz and Shavlik 1999) evolves ensembles in an *iterative* manner, and is only briefly mentioned here as another example of a study that incorporates a direct measure of diversity. The diversity is calculated for each individual (an ANN classifier) as the squared differences in outputs for the current individual compared with the rest of the population.

Other approaches to evolving base classifiers are more focused on the feature and data levels in the taxonomy described in Section 6.2.2 on page 106. For example, Chen and Yao (2006), Oliveira *et al.* (2005) and Zio *et al.* (2008) apply multi-objective optimisation to select different feature sets for different base classifiers. Ahmadian *et al.* (2007a;b) propose an approach to creating base classifiers by evolving the training samples used to train them. Therefore, they compare their approach with boosting and bagging. More details on this study in Section 6.4.1 on page 116.

6.3.2 Evolving the combination of classifiers

Two approaches to evolving classifier ensembles may be identified:

1. Selecting ensemble members from a pool of base classifiers, assuming a fixed combiner such as majority voting.

2. Optimising the combiner (*e.g.*, weights of a weighted majority vote combination) of a static set of base classifiers.

It is common for population based search/optimisation techniques, such as GAs, to produce a large pool of base classifiers. Although it has been established that the combined knowledge of the classifiers in the population can be important (Yao and Liu 1996), it may be too computationally expensive to combine them all. Hence, it may be desirable to select a smaller subset of classifiers for the ensemble. This may also have performance benefits, according to the findings of Zhou *et al.* (2002). This is referred to as an *overproduce and choose* strategy (Roli *et al.* 2001), which is becoming more common for approaches involving evolutionary algorithms.

In the literature, three types of metrics have been used in fitness functions to evaluate the ensembles: diversity among the classifiers, error rate (prediction), and complexity (size) of the ensemble. Many studies have adopted one or more of these objectives, demonstrating that they can be used successfully to generate classifier ensembles with evolutionary algorithms (Abbass 2003a, Ahmadian *et al.* 2007a;b, Chandra and Yao 2004; 2006a, Chen *et al.* 2005b, Ruta and Gabrys 2001b; 2005, Zhou *et al.* 2001; 2002). The most recent studies have adopted multi-objective optimisation techniques, taking into account several objectives at once (Abbass 2003a, Ahmadian *et al.* 2007a;b, Chandra and Yao 2004; 2006a, Chen *et al.* 2005b). Differing from the diversity metrics typically adopted in the other studies, Chandra and Yao (2004) and Chen *et al.* (2005a) measure this with negative correlation.

Dos Santos *et al.* (2006) investigate empirically the success of the three types of objectives, used in both single and multi-objective optimisation. Twelve diversity metrics were examined: the ten from Kuncheva and Whitaker (2003) (Section 6.2.7 on page 109), fault majority (Ruta and Gabrys 2005) and ambiguity (Zenobi and Cunningham 2001). Ensembles were generated from a population of 100 k-NN base classifiers created using the random subspace approach (Ho 1998b;a), assuming a fixed majority vote combiner. Their results indicate that single optimisation of the error rate leads to the highest accuracy. A multi-objective optimisation of the diversity and error rate was found to be best in terms of reducing the complexity of the ensemble whilst maintaining a high accuracy. This was more beneficial than single optimisation of the diversity alone.

Dos Santos *et al*. (2008) later extend their work by applying Pareto front spread quality measures to analyse the relationship between the three objectives considered in (dos Santos *et al.* 2006). They argue that it is important that the objectives are conflicting to ensure that the population is spread across the Pareto front. Using this notion, they found that size and diversity do not produce conflicting objectives, and, therefore, are not able to reduce the error rate of the ensemble. However, they do state that the smallest ensembles are often found with this combination of objectives. They found that when the error rate is used as an objective, regardless of whether it is paired with size or diversity, conflicting objectives are obtained. Similarly to their previous findings, a combination of the error rate and diversity led to the best trade-off solution in terms of accuracy and ensemble size.

A different complexity objective has been examined by Ishibuchi and Nojima (2006) to evolve ensembles of fuzzy rule based classifiers. Their complexity metric is calculated as the number of fuzzy rules and antecedent conditions. In addition, they measure the accuracy of the ensemble as a second objective. Similarly to other recent studies adopting multi-objective optimisation algorithms (Abbass 2003a, Chandra and Yao 2004; 2006b), they create an ensemble by combining the set of non-dominated classifiers in the Pareto front.

6.3.3 Evolving the classifiers and the ensemble

There are two approaches to evolving the classifiers and the ensembles in the same process (Ando 2007, Garcia-Pedrajas *et al.* 2005):

1. **Simultaneously:** the base classifiers and the ensemble combination are evolved at the same time.
2. **Iteratively/sequentially:** the ensemble is populated iteratively whilst the base classifiers are evolved specifically to cooperate with the current ensemble.

These two approaches are discussed further in their respective sections below.

6.3.3.1 Simultaneous learning

Arguably the best known algorithm proposed for simultaneous learning is Negative Correlation Learning (NCL) (Liu and Yao 1998, Liu *et al.* 2000), which incorporates explicit fitness sharing and a correlation penalty to the fitness function to ensure diversity. NCL does not optimise directly which individuals (classifiers) are used for the ensemble. Originally, all individuals in the population were used in the ensemble, assuming a fixed combiner such as majority voting. In (Liu *et al.* 2000), an additional mechanism was incorporated to automatically determine the number of classifiers for the ensemble. This was achieved by choosing the best individual of each species, as determined after clustering with a *k*-means algorithm. More recently, Chen and Yao (2007) propose using NCL in conjunction with bagging and random feature subspace selection. This outperformed bagging, random forest and boosting (with DTs), although the basic NCL was not included in the investigation for comparison.

Liu *et al.* (2000) argue that NCL has an advantage over iterative approaches (or independently evolving classifiers) since they do not allow the individual classifiers to interact with one another when they are evolved. Duell *et al.* (2006) contend that NCL produces more diverse ensembles, but observes that this does not imply a higher performance than implicit methods, such as (implicit) fitness sharing (speciation). This corresponds with observations of Kuncheva (2004, p 295). Furthermore, simultaneous learning is not the ideal approach for all applications, such as incremental learning, although Lin *et al.* (2008), Minku *et al.* (2009) and Tang *et al.* (2009) propose incremental NCL algorithms, which are discussed further in the following section.

As mentioned above, the NCL algorithm determines the final members of the ensemble with a *k*-means clustering algorithm. Abbass (2003a) argues that this is an inconvenient approach, since it is not clear how to determine the value of *k*. Instead, Abbas proposes creating an ensemble using a MOGA, referred to as MPANN (Memetic Pareto Artificial Neural Network). In (Abbass 2003b), the MOGA is used to speed up the learning process of ANNs trained with backpropagation. The final population provides a Pareto set of non-dominated solutions, which, in (Abbass 2003a) is used to form an ensemble. Thus, no clustering algorithm is required to determine the final set of base classifiers.

Chandra and Yao (2004) propose a multi-objective approach to ensemble generation referred to as DIVACE (DIVerse and ACcurate Ensemble learning algorithm), which incorporates elements of NCL and MPANN. They consider diversity (negative correlation) and accuracy of the ensemble as separate objectives, aiming to find an optimum trade-off between these. Similarly to Abbass (2003a), the final set of non-dominated base classifiers are combined. Chandra and Yao (2004) consider three combiners: simple average, majority vote and winner-takes-all. Later, Chandra and Yao (2006b) propose a new diversity objective for DIVACE, which they refer to as Pairwise Failure Crediting (PFC). Whilst NCL adopts a probabilistic measure of the error of the classifiers, PFC measures the errors of each classifier on the training set directly. PFC led to improved classification rates compared with the NCL measure of diversity, and also outperformed MPANN (Chandra and Yao 2006b).

Chandra and Yao (2005; 2006a) propose a framework for evolving hybrid ensembles that incorporates DIVACE. Hybrid ensembles refer to combinations of classifiers that have been trained/created using different algorithms. They implement the framework in a system referred to as DIVACE-II. This framework includes several additional components, including bagging, boosting and clustering. Utilising DIVACE in the proposed framework, led to significant performance improvements (Chandra and Yao 2006a). On the two data sets adopted, the *Australian credit card* and *Diabetes* data sets from the UCI repository, DIVACE-II obtains the lowest average error rates on the test sets compared with 28 other techniques, including DIVACE, MPANN, the original NCL algorithm (Liu *et al.* 2000), bagging, AdaBoost, Arcing, and single classifiers such as MLPs trained with backpropagation, C4.5 DT, NB and *k*-NN. Moreover, the results indicate that the PFC diversity metric proposed in (Chandra and Yao 2006b) is not advantageous compared with NCL. However, the results on these two data sets are not conclusive on this matter, since the performance obtained with each metric varied significantly on the different data sets.

Differing from the other studies discussed thus far, Garcia-Pedrajas *et al.* (2005) apply cooperative coevolution (Potter and De Jong 2000) to simultaneously evolve base classifiers and ensembles. This evolutionary process includes a MOGA, as in other recent studies, to evaluate ANN classifiers. The classifiers are evolved in different subpopulations, which provides an implicit method of obtaining diverse classifiers. These subpopulations evolve independently, and one classifier from each population is selected to contribute to an ensemble. The ensemble size is, therefore, controlled by the number of subpopulations. Findings on 10 data sets from the UCI repository demonstrate the success of this cooperative model over other ensemble methods: bagging, arcing, AdaBoost, and the generalized ensemble method of Perrone and Cooper (1993). According to their analysis, it appears that the cooperative method better performs modularisation, in which each of the classifiers specialises on particular subsets of data. When combined in an ensemble, the classifiers collectively achieve a broader coverage of the data. It should be noted that they use a weighted combiner, in which the weights of these individual expert classifiers are also optimised simultaneously.

6.3.3.2 Iterative

The first approach to explicitly obtain uncorrelated base classifiers was proposed by Rosen (1996), from which NCL is an extension (Brown and Wyatt 2003). However, whereas NCL evolves the classifiers and the ensemble simultaneously, Rosen (1996) evolves the classifiers in an iterative manner. As discussed in Section 6.3.1, the ADDEMUP algorithm proposed by Opitz and Shavlik (1999) also creates ensembles in an iterative manner. As ANNs are trained (using a modified backpropagation algorithm), examples misclassified by the current ensemble are emphasized. In this respect, the principle is similar to boosting. When a network has been trained and added to the ensemble, the whole population is scored on accuracy and diversity. Additionally, a λ parameter is employed, which can be considered as a weight of the diversity measure, and remains the same whilst the error is decreasing. However, if the population error is not decreasing and the population diversity is decreasing, λ is updated to emphasise more strongly the diversity.

Speciation has been used in early studies to evolve diverse base classifiers. Ando (2007) adopts speciation in an iterative process to create a classifier ensemble. When a new species (of ANN classifiers) is discovered, it is added to the ensemble if it enhances its performance. These species are evolved with a single objective GA, which also includes algorithms to determine weights of the networks in the ensemble at each generation of the GA, or at the discovery of a new species.

As mentioned above, NCL has also been applied to incremental learning problems. Minku *et al.* (2009)

propose two algorithms based on NCL: Fixed Sized NCL (FSNCL) and Growing NCL (GNCL). In the former, an entire ensemble is trained with NCL on initial training data. When new data is available, the entire ensemble is copied and retrained on this data. That is, the weights of the ANNs in the previous ensemble are used as initial weights of the new one, but it is trained with NCL only on the new data. In GNCL, only one ANN is trained on the initial training data. When new data is available, a new ANN is trained on this and is added to the previous ANN to form an ensemble. This iterative process is continued with every new batch of data.

The study of Minku *et al.* (2009) served well to demonstrate that NCL can be used in incremental learning. Moreover, the iterative GNCL algorithm was shown to be robust to *catastrophic forgetting*[2], which is an important issue in incremental learning. However, GNCL has worse generalisation than FSNCL, and may be limited by few classifiers in the ensemble.

Lin *et al.* (2008) and Tang *et al.* (2009) propose a Selective NCL (SNCL) algorithm, which incorporates parts of both FSNCL and GNCL algorithms. SNCL addresses drawbacks of the previous approaches, aiming to retain the benefits of generalisability of FSNCL and the robustness to catastrophic forgetting of GNCL. An ensemble of ANNs is trained with NCL on initial data, as in FSNCL. Similarly, when new data is available, the old ensemble is cloned and is trained on the new data with NCL. However, instead of discarding the previous ensemble members, they are merged to form a larger ensemble. Based on the new pool of classifiers, they use a GA to perform selection of ensemble members, according to a predefined ensemble size. Although the SNCL algorithm alleviates the drawbacks of the FSNCL and GNCL algorithms, there is a significant increase in training time due to the selection process.

6.3.4 Summary

Combining classifiers is a popular approach to improve on the performance of a single (monolithic) classifier (Bauer and Kohavi 1999, Brown *et al.* 2005, Dietterich 2000a, Tan and Gilbert 2003). The premise of classifier combination is that the individual classifiers that are combined are diverse, *i.e.*, they make errors on different instances (Brown *et al.* 2005, Hansen and Salamon 1990, Kuncheva 2004, p. 295). Some common approaches such as bagging, Random Forests (RFs) and AdaBoost rely on the *instability* of the classifiers to achieve diversity, *i.e.*, classifiers trained on different data subsets, feature subsets or with different configuration parameters should yield different classification performance. These approaches achieve diversity in an *implicit* manner, since they do not adopt a measure of diversity when creating the classifiers. Other approaches consider diversity *explicitly*, which Tang *et al.* (2006) argue is necessary when such approaches are unable to produce sufficiently diverse classifiers. Many explicit diversity measures have been proposed, ten of which were considered in (Kuncheva and Whitaker 2003). Their findings are inconclusive regarding the success of the different diversity metrics, however, Tang *et al.* (2006) argue that it is unimportant which diversity metric is used.

Genetic Algorithms (GAs) (and other evolutionary algorithms) have been used in different stages of creating classifier ensembles: evolving the base classifiers for the ensemble; evolving the combination of a set of existing base classifiers; evolving the base classifiers and ensemble concurrently. Explicit diversity metrics are typically adopted when evolving base classifiers, but not necessarily when the combination of base classifiers is evolved. Negative Correlation Learning (NCL) (Liu and Yao 1998, Liu *et al.* 2000) is a popular evolutionary approach that aims explicitly to evolve diverse classifiers (that are negatively correlated) by employing a correlation penalty in the fitness function of the GA. NCL evolves the classifiers and ensemble concurrently; originally simultaneously, but recently also iteratively (Lin *et al.* 2008, Minku *et al.* 2009, Tang *et al.* 2009).

[2] Catastrophic forgetting refers to a classifier experiencing a significant decrease in performance on older data after it has been retrained on new data.

In addition to obtaining diversity among the base classifiers, determining the size of the ensemble is one of the challenges that needs to be addressed when employing evolutionary algorithms to create classifier ensembles (Abbass 2006, Yao and Islam 2008). In the original NCL algorithm, all individuals (classifiers) in the population were combined. However, due to computational costs, this may not be possible in practical applications. Furthermore, according to the findings of Zhou *et al.* (2002), combining smaller subsets can yield performance improvements. This has given rise to a strategy referred to as *overproduce and choose* (Roli *et al.* 2001). According to Yao and Islam (2008), clustering is a simple approach to reducing the size of an ensemble without significant loss in accuracy. However, this approach constrains the selection to a uniform sample and assumes that a fusion combiner is used. Therefore, an evolutionary approach to selection may be more fruitful. Furthermore, in recent years, multi-objective algorithms have been shown to be very successful in this domain, which may take consideration of different objectives concurrently, such as diversity, ensemble size and accuracy (Abbass 2003a; 2006, Chen *et al.* 2005a, Chen and Yao 2007, dos Santos *et al.* 2006; 2008, Ishibuchi and Nojima 2006, Kondo *et al.* 2007).

Although specific benefits of combining classifiers have been discussed in the literature (Dietterich 2000b, Garcia-Pedrajas *et al.* 2005), this is not guaranteed to improve upon the performance of an individual classifier (Quinlan 1996; Kuncheva 2004, pp. 103–104). However, these observations are based on ‘conventional’ combination approaches such as bagging and boosting. Although evolutionary algorithms may determine better combinations of classifiers, all current approaches (including non-evolutionary approaches) are limited by the exclusive focus on the accuracy of the resultant ensemble (and ensemble size). That is, none of the existing studies consider the classification trade-off discussed at the beginning of this chapter. Selection of base classifiers is one of the main challenges in classifier combination (Abbass 2006, Yao and Islam 2008), and is arguably a key factor in determining the classification trade-off. Consideration of the classification trade-off and an analysis of the selection process may give significant insights into why classifier combinations are not always successful. This is investigated further in the following sections.

6.4 Method

Related research is discussed in Section 6.4.1, followed by the aims of this investigation in Section 6.4.2. The proposed method, MABLE (**M**ulti-Objective Evolution of **A**rtificial Neural Network Ensem**ble**s) is discussed in Section 6.4.3, followed by implementation details in Section 6.4.4. Other classifiers (and classifier ensemble approaches) that are adopted for comparison of results are discussed in Section 6.4.5. Details of the data set and metrics are presented in Sections 6.4.6 and 6.4.7, respectively. An outline of the empirical investigation is provided in Section 6.4.8.

6.4.1 Related work

Parrot *et al.* (2005) propose using an evaluation function (hereafter referred to as the Parrot function) for multi-objective genetic programming, which considers the accuracy of each class as separate objectives. The Parrot function is nearly identical to Eval5, except that the error for each class is calculated as the sum of misclassified instances, whilst Eval5 uses a percentage measure.

The aim in the approach of Parrott *et al.* (2005) is to obtain a single classifier that should have a perfect score on all objectives. If this does not occur, the individual with the lowest aggregated error is chosen. In this sense, the algorithm shares the same weakness as backpropagation, being biased towards the major class(es). Arguably, it is unlikely to obtain one solution that scores perfectly on all objectives; Coello Coello and Christiansen (2000) state that this rarely occurs for multi-objective optimisation, and for classification, implies clean and separable data. Although the Parrot function was not proposed for learning from imbalanced data, it is therefore, not well suited for such an application although the approach bears similarities with the one taken here.

The Parrot function has been adopted by Ahmadian *et al.* (2007a, 2007b) to evolve an ensemble of classifiers. Both methods that Ahmadian *et al.* propose follow a two-phased approach to creating the final ensemble using a modified version of NSGA-II (Deb *et al.* 2000) in both phases. The first phase generates the classifiers and the second optimises the composition of the classifiers for a fusion ensemble, using a majority vote combiner. The influence of the Parrot function is utilised in the first step of the method, but the specific implementations differ between Ahmadian *et al.* (2007a) and Ahmadian *et al.* (2007b). In the former study, the objectives for optimising the classifiers are to minimise the aggregated error of each of the classes and to maximise the diversity amongst them. This is unclear in the paper, but was clarified via correspondence with Ahmadian (2008). Since the error on each class is not treated as separate objectives, this is similar to a general error measure such as MSE, which will have the same issues as the implementation of Parrott *et al.* (2005), being biased towards the major class(es).

In the second phase of the approach proposed by Ahmadian *et al.* (2007a, 2007b), the objectives are to minimise the size of the ensemble and maximise the accuracy. Consequently, the drawback of their approach is similar to the rest of the current body of research in this area, to create a single best solution based on general performance metrics.

6.4.2 Aims of the investigation

This final empirical investigation has five aims, as discussed below.

Aim 3.1: Develop a method that learns from imbalanced data and is capable of producing a set of classifiers that exhibit different classification trade-offs

The ENN from the previous chapter provides a successful method for learning from imbalanced data, but there is no control over the classification trade-off offered by the evolved MLPs. Occasionally, the ENN would, therefore, produce solutions with unacceptable performance as a result of obtaining a trade-off exhibiting too great a FPR. The first phase of MABLE extends the GA employed in the ENN to consider multiple objectives, and evolve a population that exhibits different trade-offs among the objectives that are adopted. Using the classification rate on each class as separate objectives, different trade-offs in classification performance can be obtained.

Aim 3.2: Extend the approach developed for aim 3.1 to create classifier ensembles

It can be seen that the first phase of MABLE provides a pool of *diverse* base classifiers, which can be exploited to form classifier ensembles. As discussed in Section 6.2.7 on page 110, the premise of classifier combination is that the base classifiers are diverse. This is achieved *implicitly* in MABLE since the MLPs are evolved to exhibit different classification trade-offs, which implies that they make errors on different instances.

Similarly to phase one of MABLE, a multi-objective approach is proposed for the second phase to evolve the ensembles so that these too exhibit different classification trade-offs. Although several researchers have adopted multi-objective algorithms to create ensembles, they all strive to obtain a single 'best' ensemble, based on metrics such as accuracy, diversity and complexity (size). Consequently, there is no control over the classification trade-offs offered by the resultant ensembles. The aim of phase two, therefore, is to obtain a new Pareto front of solutions (classifier ensembles) that improve on the Pareto front of the base classifiers.

Aim 3.3: Investigate how to select base classifiers to obtain a desired trade-off for an ensemble

Similarly to the drawbacks of the ENN, the existing combination approaches based on combining an arbitrary, uniformly selected set of classifiers results in a single ensemble that may offer an unfruitful classification trade-off. Therefore, a base classifier may provide a better solution if the trade-off is better. Since MABLE evolves a Pareto front of ensembles that exhibit different classification trade-offs, this provides an opportunity to determine whether there is a pattern to how one can select base classifiers to obtain a desired trade-off. The Pareto analysis provides a new perspective on the selection problem, which has not been considered previously in the literature.

Aim 3.4: Analyse the properties of the proposed approach

First, it is desirable to determine the extent to which MABLE is successful at evolving MLPs and ensembles thereof for imbalanced data, producing a range of solutions with different classification trade-offs. There are also practical considerations that are interesting to investigate, such as the performance of ensembles of varying sizes and how well the MLPs and ensembles generalise.

Aim 3.5: Comparison with other classifier combination approaches

The final aim of this investigation is to determine how well the MLPs and ensembles evolved with MABLE perform compared with other classifier combination methods.

6.4.3 Proposed method

The approach proposed here, MABLE, incorporates mechanisms for evolving classifiers (MLPs) and ensembles thereof in two separate phases. Both phases are multi-objective, offering the user a close approximation of the Pareto front of non-dominated solutions of both base classifiers and ensembles. The classification rates on the classes are adopted as separate objectives in both phases.

There are three main challenges that need to be addressed when creating an ensemble with evolutionary algorithms (Abbass 2006, Yao and Islam 2008):

1. Obtain diversity among classifiers.
2. Determine which classifiers to combine (selection).
3. Determine the size of the ensemble.

Diversity among the base classifiers is achieved *implicitly* in MABLE since the MLPs are evolved to exhibit different classification trade-offs, which implies that they make errors on different instances.

MABLE optimises the selection of classifiers to obtain a Pareto front of ensembles that exhibit different classification trade-offs. Alternative approaches to selecting and combining classifiers are also examined: combining all classifiers in the Pareto front, smaller subsets according to clustering, and smaller subsets according to a filtering process based on classification thresholds. Further details are provided in Section 6.4.5.

Several approaches to determining the ensemble size are considered here. The maximum ensemble size is determined by the the number of non-dominated classifiers in the archive from phase one of MABLE. In the case of combining all classifiers, the ensemble size can become quite large. This is the main motivation for performing clustering, to combine a smaller set of uniformly selected classifiers (Yao and Islam 2008). k-means clustering (MacQueen 1967) is adopted here, in which the value of k can be altered to determine the size of the ensemble. Further details are provided in Section 6.5.

In the case of evolving ensembles, the MOGA implementation adopted here allows for fixed sized and variable sized ensembles to be evolved. The former may be desirable if a user can determine a fixed size *a priori*, which may be necessary due to constraints on computational costs. However, if this is not an issue, evolving variable sized ensembles may obtain a greater coverage of the Pareto front. That is, an ensemble of size 10 may obtain a particular classification trade-off that an ensemble of size 5 cannot. In the case of evolving variable sized ensembles, a maximum size can be determined to allow for some flexibility whilst preventing too large ensembles from being evolved.

Although not examined here, the ensemble size may be included as an additional objective, which would allow the user to view the trade-offs that emerge in complexity as well as classification rates.

6.4.4 MABLE implementation

As this study is an extension of the previous work with the ENN, the implementation of the MOGAs used in MABLE is an extension of the GA used in that work. However, it should be noted that the approach may be implemented with any suitable multi-objective optimisation algorithm.

6.4.4.1 Base classifier and evaluation function

As a base classifier, the MLP remains the same as in the ENN, except that only three layers are considered here since the previous study showed no significant benefits from adopting four layers. Furthermore, a smaller number of neurons in the hidden layer are also considered, to examine whether ensembles consisting of smaller base classifiers remain successful, thereby promoting modularisation. The following number of neurons in the hidden layer are examined: {10, 30, 50, 70}.

The evaluation function adopted here is an extension of Eval5 (from the previous study) to multiple objectives: maximising the classification rate of each class as separate objectives. This evaluation function is used in both phases of MABLE. However, the approach is not constrained to these objectives. Examples of other objectives include classification error for each objective, TPR and FPR, and may include additional objectives such as ensemble size to view the trade-offs that emerge in complexity.

6.4.4.2 MOGA for phase one

The MOGA used in phase one, to evolve base classifiers (evolving the weights of MLPs), is almost a direct extension of the GA used in the previous investigation. The MOGA is real-coded, uses the same crossover and mutation operators, tournament selection and an elitist replacement strategy. To enable optimisation of multiple objectives, non-dominated sorting and fitness sharing are adopted, similar to those used in NSGA (Srinivas and Deb 1994). Additionally, an archive function is adopted similar to that of NSGA-II (Deb *et al.* 2000) and SPEA (Zitzler and Thiele 1999). This maintains a set of all non-dominated solutions found from the start of the evolutionary process.

Unlike the other algorithms that adopt an archive mechanism, the archive in this MOGA does not participate in the selection process, nor fitness calculation. However, there is a significant benefit of incorporating such an archive function. The archive may save many non-dominated solutions that have been lost in the population, due to the particular direction of the search. This is particularly beneficial with increasing number of objectives, since the number of non-dominated solutions required to sample the Pareto front increases significantly. For example, results presented later show that evolving MLPs for a two class problem gave an archive of approximately 30 non-dominated solutions, compared with more than 300 with three classes. The experiments used a population size of 200, in which the Pareto front of the population consisted of 132 non-dominated solutions. An archive of 347 non-dominated solutions, therefore, gave a much more complete Pareto front.

Most of the MOGAs proposed in the literature do not consider replacement, since they are generational. Compared with the replacement strategy adopted in the ENN, some modifications are required here due to the consideration of multiple objectives. First, a set of individuals that the offspring dominates is populated. From this set, the offspring replaces one at random. However, if the offspring does not dominate any individuals, it is discarded. A different elitist replacement strategy, based on replacing the lowest ranking individual, was examined during preliminary experiments. However, this was not fruitful as it lead to premature convergence.

Fitness sharing is not utilised in replacement since this would require non-dominated sorting to be performed after each offspring had been replaced to calculate new fitness values. However, this is calculated after replacement and fitness sharing is utilised in the selection process. Parents that undergo tournament selection are chosen based on their fitness values, which biases selection to less crowded regions of the search space.

Further modifications were made for pragmatic reasons since the MOGA is significantly more time consuming than the single objective GA used in the previous study. This is due mainly to the non-dominated sorting that takes place after each evaluation (when replacement takes place). Therefore, the first modification addressing this is implementing a semi-generational structure, in which a pool of offspring is populated and replacement performed in a block, followed by non-dominated sorting. The size of the offspring pool becomes an additional parameter, which was examined in preliminary studies. The second modification further improved run time by parallelisation, which is considered in Section 6.4.4.4.

The change to a semi-generational structure made a significant impact on the speed of the algorithm. A preliminary experiment was conducted to demonstrate this: whilst the single objective GA from the previous phase too 1 hour and 56 minutes to complete 20,000 evaluations, the MOGA with a steady state structure took 13 hours and 33 minutes. The MOGA with a semi-generational structure took 5 hours and 25 minutes to complete the same number of evaluations, with the offspring pool size set to half the population size.

Despite the speedup, there was some loss in search performance when changing to a semi-generational population structure. Therefore, an additional elitist mechanism was incorporated to improve the convergence speed of the algorithm. For this, the archive is used. At the time of replacement, a small percentage of the archive is reintroduced into the population (selected at random). The replacement is conducted in the same manner as with offspring, as described above. An additional benefit of this is the reintroduction of genetic material that may have been lost due to genetic drift (leading to an incomplete sampling of the Pareto front). With this modification, the MOGA was able to find solutions of the same quality as the single objective GA.

6.4.4.3 MOGA for phase two

Other studies that have evolved base classifiers for classifier ensembles have combined the Pareto front of non-dominated solutions instead of the entire population (Abbass 2003a, Chandra and Yao 2004; 2006b, Ishibuchi and Nojima 2006). Similarly, the entire population is considered superfluous here, as we are only interested in those solutions with non-dominated trade-offs in performance. Therefore, phase two of MABLE takes the archive of non-dominated solutions from phase one and evolves classifier ensembles. A fixed combiner is assumed; the majority vote combiner, which was chosen due to its success in the literature (Ruta and Gabrys 2001a;b; 2005, Zhou *et al.* 2002, Yao and Islam 2008). In case of a tie, the winner is randomly chosen.

The same modifications to the MOGA in phase one applies to the one in phase two. Furthermore, an integer-coded chromosome representation is adopted to better accommodate for the needs of the selection process in a more intuitive manner. Other representations have been considered. For example, a real valued chromosome representation can be adapted to perform selection in two ways:

1. Using a chromosome size equal to the number of base classifiers, each gene represents a base classifier. The gene values are within the range $[0,1] = \{x \in \mathbb{R} | 0 \leq x \leq 1\}$, where a value ≥ 0.5 signifies that the classifier is used in the ensemble, and a value < 0.5 signifies that it is not used.

2. Using a chromosome size equal to the desired ensemble size. Each gene represents a classifier, but the gene value is scaled within the range $[1,N] = \{x \in \mathbb{Z} | 1 \leq x \leq N\}$, where N is the number of base classifiers. Thus, the gene value represents the index/ID of a base classifier.

In the first case, it would be more intuitive to use a binary encoding. However, in either case, to accommodate fixed sized ensembles, additional mechanisms are required to control the genetic operators. A simple crossover mechanism would not guarantee that the ensemble size stay fixed; on the contrary, it is likely to change. Similarly, for a normal mutation operator, it *will* change.

In the second case, it is more straightforward to allow for fixed sized ensembles, which is not complicated by the genetic operators. Crossover and mutation will lead to different classifiers being used, but the number of them will not be altered. However, the issue with this solution is that the operators do not guarantee that the same classifier is not used more than once in an ensemble. Given that a majority vote combiner is used in this study, this is not desirable. The effect of duplicate classifier members is similar to giving more weight to one (or more) classifier(s) over others in the ensemble, which may be fruitful. However, in comparison to using a weighted combiner, duplicate members induce a computational overhead.

Following the discussion above, an integer coded MOGA is adopted. The implementation of this MOGA is similar to that in (Wellington and Vincent 2002). The chromosome size corresponds to the size of the ensemble, and each gene value corresponds to a classifier from the first phase of MABLE. The order of the genes is irrelevant, which gives flexibility in the representation. In particular, this makes it convenient to add or remove a classifier if variable sized ensembles are evolved.

The chromosome is implemented as a set, and the recombination of offspring is consistent with set theory. Selection of two parents is made according to tournament selection. Thereafter, the union of the parent genes is created, which forms a pool of genetic material from which the offspring is generated. The gene values for the offspring are randomly selected from this set, according to its size. An example for a fixed sized implementation follows:

```
Parent 1   <1,6,3,8,2>
Parent 2   <3,2,5,9,6>
Union      <1,2,3,5,6,8,9>

Offspring  <2,5,3,6,8>
```

In a fixed sized implementation, the size of the offspring will be the same as the parents (as in the example above) and the union of parents will be at least as large as the offspring. For variable sized ensembles, a mechanism is needed to determine the size of the offspring. This could be calculated from the range of sizes present in the selected parents. However, to encourage diversity, and allow the population to adapt successfully the size of the ensemble, the offspring size here is chosen randomly within a range extended by +/-1 (the range being constrained to have a minimum size of 2 and maximum size of the number of base classifiers), much like the extended blend crossover adopted for the real coded MOGA of phase one, without which there is a tendency to converge to the population mean (Eshelman and Schaffer 1993). If the offspring size is larger than the union of classifiers formed from the parents, an additional classifier is added to the offspring at random. An example of offspring generation for variable sized ensembles follows:

```
Parent 1    <3,5,2>
Parent 2    <1,8,3,6,2>
Union       <1,2,3,5,6,8>

Parent size range [3,5]
Offspring size range [2,6]
Randomly chosen offsing size: 4

Offspring  <1,3,6,2>
```

The mutation operator does not use a mutation strength, as in the real-coded MOGA. Instead, it operates only with a given probability of mutating an offspring (rather than a probability of mutating each gene, as in the real-coded MOGA, since this was found to be too disruptive). For fixed sized ensembles, the mutation randomly changes the value of a gene to the index of another classifier not currently used in the ensemble. For variable sized ensembles, there are three operations that mutation can perform:

1. Change a gene value at random to the index of another classifier.
2. Add a gene, which is randomly set to an available classifier (incrementing the chromosome size).
3. Remove a gene at random (decrementing the chromosome size).

For the first two genetic operations, the operator ensures that duplicate classifiers are not used in the ensemble by only randomly selecting from a set of classifiers not currently selected for that individual (ensemble).

The choice of mutation operation is done randomly. However, in the case of operator #1 and #2, if the chromosome size is already equal to the pool of classifiers, it cannot be conducted. In this case, operator #3 is executed instead. This ensures that the diversity function of the mutation operator remains constant throughout the search. Similarly, if the chromosome size is already at its minimum size (2), operator #3 will not be performed; instead operator #2 will add a random classifier to the ensemble. Thus, crossover and mutation operators maintain the strict set semantics of the chromosome, ensuring that only valid individuals are produced.

6.4.4.4 Parallel implementation

The parallel implementation chosen here follows the global model, which does not change the semantics of the MOGA. PVM (Geist *et al.* 1994) provided the necessary middleware to enable evaluations to be farmed out to slave processes running on a cluster of Linux computers. This is performed asynchronously to minimise the time that the slave processes are idle. Figure 6.4 illustrates the implementation.

To facilitate asynchronous processing of evaluations, the master thread implements the MOGA, maintaining a global population, and inserting offspring into a buffer. A separate buffer management thread (BMT) is used to schedule offspring to slave processes for evaluation, and to receive and store the performance evaluation of the offspring from the slave processes, allowing the slaves to be kept occupied during global operations on the population. When the buffer contains a minimum number of completed evaluations, determined by a *readback* parameter[3], the master thread reads available results from the buffer and performs replacement, followed by the usual process of non-dominated sorting, fitness sharing, and reproduction, filling the now empty slots in the buffer ready for evaluation. Through a 'well behaved' (in concurrent system terms) implementation of the master thread (*i.e.*, one that provides appropriate yield points, allowing the buffer management thread to run), and by careful selection of parameters – buffer size

[3] The readback parameter effectively determines the population structure of the MOGA, as discussed in Section 6.4.4.2.

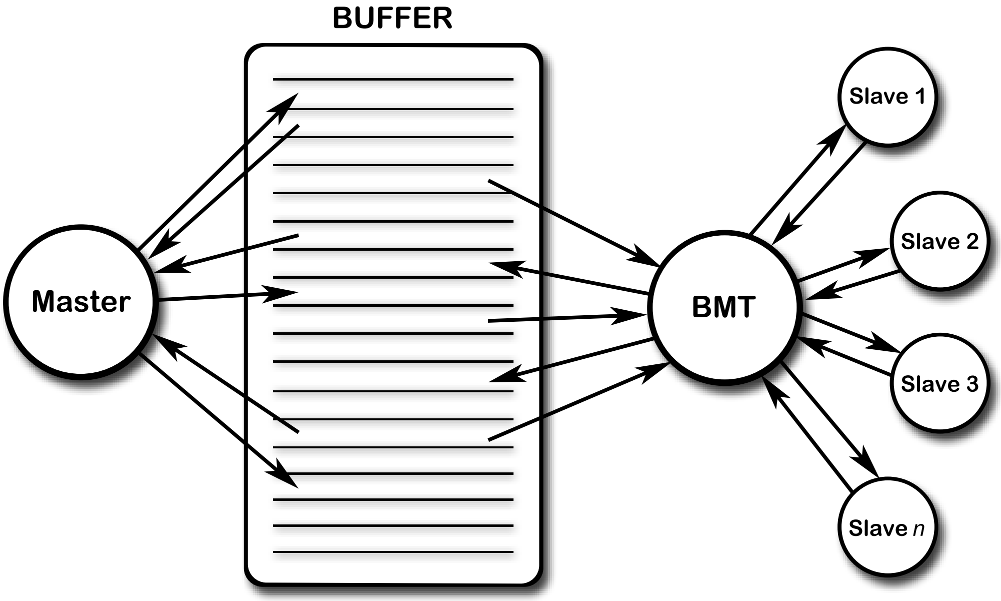


Figure 6.4: Parallel implementation of the MOGA.

and readback size – for a given number of slave processes, this asynchronous implementation is able to maximise the load on the slaves and minimise the time spent by the master in performing the time consuming non-dominated sorting and sharing operations. Note also that this implementation is tolerant to the failure of slave processes.

6.4.4.5 Configurations

The configurations of the MOGAs used in phase one and two of MABLE are given in Table 6.1. Using the configurations of the GA in the previous study as a starting point, the configuration of the MOGAs were refined according to preliminary experiments to ensure a good coverage of the Pareto front with an acceptable solution quality.

Table 6.2 provides an overview of the settings related to the parallel framework. The number of slaves was constrained by the number of hosts available. According to this, the buffer size and readback threshold were adjusted to minimise the idle time of the slaves.

Table 6.1: Configuration details for the MOGAs.

Parameter	Phase one	Phase two
Population size	200	200
Over init.	100	100
Tournament size	5	5
Mutation rate	0.1	0.2
Mutation strength	0.3	N/A
σ_{share}	0.1	0.1
Evaluations	{5000, 7,000, 10,000}	5,000
Elitism	Re-introduce 5% of archive every 500 evaluations	

Table 6.2: Configuration details for the parallel framework.

Parameter	Value
Buffer size	60
Readback	30
Number of slaves	35

6.4.5 Other combination techniques

In addition to using the MOGA in phase two of MABLE, different methods to combine the pool of MLP classifiers obtained in phase one are examined:

- Combine all non-dominated MLPs in the archive.
- Combine smaller subsets according to clustering.
- Combine reduced sets of MLPs that exceed specific thresholds on performance:

The clustering approach is discussed in Section 6.4.5.1, and the threshold approach in Section 6.4.5.2. Additionally, the following classifier and classifier combination approaches are also included for comparison of results: DT, Random Forest (RF) and AdaBoost. The same DT specifications as in the previous investigations in this thesis are adopted here. Section 6.4.5.3 provides details about RF and AdaBoost.

6.4.5.1 Clustering approach

Yao and Islam (2008) suggest that clustering is a good way to reduce the ensemble size at an acceptable loss of accuracy. A *k*-means (MacQueen 1967) clustering algorithm is adopted here, as suggested in (Yao and Islam 2008), which is a well known clustering algorithm that is ideal for this purpose. The parameter *k* determines the number of clusters, which here equates to the ensemble size. Depending on the number of base classifiers in the archive, the following values of *k* are examined: {5, 10, 20, 30, 40, 50}. Clustering is performed until a convergence threshold of 10^{-6} is reached[4].

Typically, the cluster centroids are initialised randomly. However, this was not fruitful for this application, as most of these centroids would be initialised far from the Pareto front, as seen in Figure 6.5a. Consequently, some of these centroids would never manage to establish a cluster on the Pareto front, as seen in Figure 6.5b (8 out of 10 clusters have been established). Therefore, a guided random initialisation process was adopted, which initialises the centroids along the Pareto front. This gives a much better starting point, as illustrated in Figure 6.5c, which does not have the issue of clusters not being established.

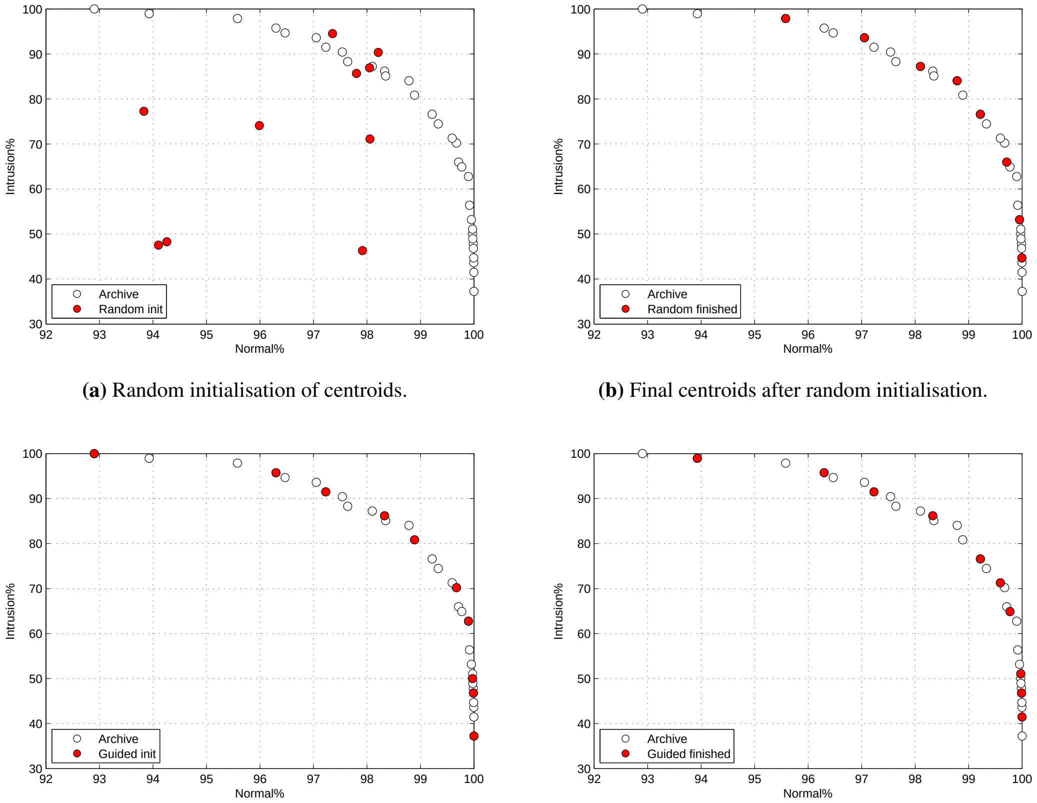
The guided initialisation is conducted by initialising the cluster centroids according to the classifiers in the archive. First, the archive is sorted according to the first class (*Normal* in this case). Thereafter, the classifiers are selected uniformly according to their index values, in a way that guarantees that the first and last classifier in the archive are selected as centroids. The pseudo code in Algorithm 6.1 specifies this process, where *N* refers to the number of classifiers in the archive:

The clustering algorithm does not guarantee that cluster centroids are formed at the boundary of the performance space, which can be seen in Figure 6.5d. Consequently, classifiers that obtain the highest performance for each class may not be included. These classifiers may be important, and are, therefore, included in a new ensemble for comparison.

[4] *k*-means is an iterative clustering algorithm that will continue to adjust the cluster centroids until the maximum adjustment (in one iteration) falls below a defined threshold for convergence.

Algorithm 6.1 Initialisation of clusters for *k*-means in this thesis.

```
sort archive
distance = 1 / (k−1)
acc = 0
while (acc < 1)
   idx = acc ∗ (N−1)
   initialise cluster according to classifier at index idx
   acc += distance
end while
```



(a) Random initialisation of centroids.

(b) Final centroids after random initialisation.

(c) Guided initialisation of centroids.

(d) Final centroids after guided initialisation.

Figure 6.5: Initialisation approaches for *k*-means clustering.

6.4.5.2 Threshold approach

One of the assumptions of classifier combination, according to Kuncheva (2004), is that a successful ensemble is composed of good classifiers that excel in different ways (makes errors on different data). Therefore, it can be interpreted that an ensemble should not include classifiers that perform poorly in some manner. However, this is a vague statement, because there is little information about this in the literature. It is only clear that the classifiers need to be diverse, as discussed in Section 6.2.7 on page 109.

One possible assumption one can make from the statement above, is that each classifier should perform better than a random classifier, which implies obtaining more than 50% accuracy according to the *Condorcent Jury Theorem* (Valentini and Masulli 2002)[5]. However, the question remains as to whether this

[5] The Condorcent Jury Theorem states that *"the judgment of a committee is superior to those of individuals, provided the individuals have reasonable competence (that is, a probability of being correct higher than 0.5)"* (Valentini and Masulli 2002).

should require above 50% correct classification on all classes. Since this is a novel problem statement, it is investigated empirically here, examining different threshold values. First, it is arguably desirable to exclude classifiers that classify everything as one class. From there, different classification rate thresholds are examined in increments of 10%: {0, 10, 20, 30, 40, 50}.

It is hypothesised here that this threshold approach will give the resultant ensembles different performance trade-offs. This is further examined by removing classifiers that do not exceed a threshold for one particular class, such as *Normal*. This may help bias the resultant ensembles to avoid unacceptable FPRs. Similarly, if a uniform selection of classifiers results in a too low TPR, thresholds may be applied that select more classifiers that obtain a higher TPR. The following *class thresholds* are examined (%): {50, 60, 70, 80, 90, 95, 97}.

6.4.5.3 Random forest and AdaBoost

RF and AdaBoost are popular classifier combination approaches, which have been adopted in the literature to learn from imbalanced data, as reviewed in Section 5.1.4 on page 80. Similar to the other classifiers adopted in this thesis, the Weka implementations are used. However, due to bugs in the 'book version' of Weka (version 3.4.11) affecting the RF algorithm, a newer version of Weka was adopted for these experiments (version 3.6.1).

The parameters of RF are the number of DTs and random features. The number of trees are investigated in a systematic manner, similar to the number of ensemble members investigated for MABLE: {5, 10, 20, 30, 40, 50}. All features are used for all trees, since this study does not consider feature selection.

The parameters of AdaBoost are the number of iterations, weight threshold for weight pruning, and whether resampling is used (instead of reweighting). Better performance was obtained with resampling, and is, therefore, enabled. The number of iterations and the weight thresholds are investigated in a systematic manner; iterations $\in \{10, 30, 50\}$ and thresholds $\in \{10, 50, 100\}$. The default values for AdaBoost in Weka are 10 iterations and a weight threshold of 100.

Both combination algorithms use a DT as a base classifier. There is no control of the configuration of the DT for the RF implementation in Weka. For AdaBoost, the default configuration is used, which employs 'normal' pruning with a confidence factor of 0.25. This configuration also yielded good performance in the previous experiments in this thesis.

It is worth noting that RF produces a diverse set of DTs in an implicit manner, similar to MABLE, whilst AdaBoost does this in an explicit, iterative, manner. Furthermore, AdaBoost may have an advantage over MABLE and RF since it makes use of a weighted combination of classifiers.

6.4.6 Data set

The KDD Cup '99 data set is adopted for this investigation, as used previously in this thesis. However, different subsets are employed for two different parts of this investigation. A two class subset, comprised of *Normal* and *U2R* data, is adopted for an initial investigation of MABLE, which allows for a thorough analysis of the algorithm. This is mainly due to pragmatic reasons, such as being able to use graphical representations to better show how the algorithm behaves, and a much reduced run time, allowing many experiments to be performed.

The *U2R* class was chosen instead of *R2L* since it is more challenging to learn. Furthermore, the data subset adopted here is comprised of *U2R* instances from both the official training and test sets, whilst the Normal instances were selected only from the 10% training set. This was done according to the findings in the first part of this thesis, which demonstrated that the results of holdout validation are very sensitive to the partitioning, especially for the minor classes. Therefore, the additional 70 instances from the test set make a significant difference, which also helps to reduce the absolute and relative rarity of this class, allowing a better analysis of the generalisation ability of the algorithm. The data set remains, however, extremely imbalanced; the *U2R* instances make up merely 0.13% of all instances. Furthermore, related to the small disjuncts problem discussed in Section 5.1 on page 77, there are still very few instances of most of the individual attacks that make up the *U2R* class. Table 6.3 provides an overview of the instances of each attack in the training and test partitions.

Table 6.3: Number of *U2R* attack instances.

Attack	Training	Testing
buffer overflow	41	11
loadmodule	8	3
perl	4	1
ps	12	4
rootkit	18	5
sqlattack	1	1
xterm	10	3

The second part of this investigation focuses more on performance and scalability. For this, a three class subset is adopted, based on the official 10% training set and test set. As in the previous investigation, the three classes are *Normal*, *U2R* and *R2L*. Furthermore, according to the findings of the first investigation, duplicates and *Normal* instances identical to intrusions are removed to avoid the methodological issues related to *R2L*, as discussed in Section 4.5 on page 71, which detract from the focus of this investigation. The class distribution of this subset is presented in Table 6.4 below.

Table 6.4: Proportions of instances in the classes of the training and test sets employed in the second part of the MABLE investigation.

Class	Training	Test
Normal	87,831 (98.82%)	47,843 (93.86%)
U2R	52 (0.06%)	70 (0.14%)
R2L	999 (1.12%)	3,058 (6%)

The preprocessing and partitioning for holdout validation is done in the same manner as previously in this thesis: all 41 features of the data set are adopted; nominal features are enumerated and then all feature values are scaled within the range [0,1]. An 80/20 split for holdout validation is adopted for the two class data set, for which the selection of training and test data is performed chronologically. This is done according to individual attacks, whilst ensuring that attacks with as few as 2 instances obtain at least one instance in the test set.

6.4.7 Metrics

Performance data for all techniques is collected on both the training and test sets. This includes every individual in population, Pareto front of the population, and archive of the MOGAs, all of which represent

MLPs and ensembles. Confusion matrices are used, as in the previous investigations. From these, the following metrics are derived: accuracy, TPR, FPR and classification rates for each class.

In a practical application, there are pragmatic constraints that determine how many false positives can be accepted. There are few guidelines in the literature about this for intrusion detection, but Hu *et al.* (2003) and Maxion and Townsend (2002) consider a 1% FPR acceptable. This is adopted here as a constraint when comparing the performance of the classifiers employed in this study.

To enable a comprehensive analysis of MABLE, the following data is also collected.

- The entire population in the MOGAs is saved at regular intervals. This is useful to confirm that the algorithm successfully evolves the population and helps to determine a suitable stopping criterion.
- Size information about the archive and Pareto front of population, and variable sized ensembles.
- The number and identity of the base classifiers each ensemble dominates.
- Which base classifiers are used in each ensemble, and which were used, or not used at all, in any ensemble.

Except for a specific study on generalisation, each algorithm is run three times. From this, the best trial is chosen, as in the previous investigations in this thesis. The analysis of the generalisation ability of the MLPs and ensembles evolved with MABLE is based on 50 trials. The difference in training and test performance for each MLP and ensemble is calculated to give a numerical metric to assist in this analysis. This is calculated by subtracting the classification rates on each class on the test set from the classification rates on the training set. Therefore, a negative value indicates that the test performance is better.

6.4.8 Outline of empirical investigation

There are two parts to this empirical investigation: the first aims to provide a thorough analysis of MABLE on a two class data set, and the second examines scaling to a larger three class data set and compares the performance of MABLE with other classifiers.

In the first phase of MABLE, MLPs are evolved in a multi-objective manner, using the classification rates of the two classes, *Normal* and *U2R*, as separate objectives. At the end of the run, the archive of non-dominated MLPs provides a diverse set of classifiers that exhibit different classification trade-offs. Based on this archive, classifier ensembles are evolved in the second phase of MABLE. Both fixed sized and variable sized ensembles are evolved. Other, more common approaches to classifier combination are investigated for comparison: combining the entire archive or smaller subsets determined by clustering. Furthermore, a new approach is also investigated here: determining the ensemble members by filtering out classifiers that do not exceed predefined classification thresholds. This may result in ensembles of higher accuracy, as well as help bias the ensemble performance towards desirable classification trade-offs for a given application.

In the first part of this investigation, emphasis is given to the analysis of the algorithm, as easily interpretable graphs can be produced from the two class data. Not only does this part aim to demonstrate whether MABLE is successful or not, the analysis provides a novel perspective on ensemble generation, demonstrating how generating a single ensemble may result in an inferior solution. The selection of base classifiers by MABLE, clustering, and threshold filtering, is examined, which sheds light on how this process affects the performance of the resultant ensembles. This part of the investigation on the two class data also includes an analysis of generalisation, ensemble size and computational costs.

In the second part of this investigation, the performance of MABLE is put to test on a larger, three class data set. The results are compared with other classifier combination techniques: Random Forest and AdaBoost. These classifiers are widely used in the literature, which helps put the performance of MABLE in context. This part also gives indications of how well MABLE scales with more objectives.

6.5 Results

Sections 6.5.1 and 6.5.2 analyse the capability of MABLE to evolve MLPs and ensembles. Other approaches to combining the MLPs evolved in phase one of MABLE are examined in Section 6.5.3. An analysis of the selection of base classifiers is considered in Section 6.5.4. Section 6.5.5 focuses on the generalisation ability of the base classifiers and ensembles, followed by a treatment of complexity and computational costs in Section 6.5.6. Finally, performance on three class data is evaluated in Section 6.5.7, which includes a comparison with DTs, RF and AdaBoost.

6.5.1 Phase one of MABLE

Figure 6.6 illustrates how the population of MLPs in phase one are successfully evolved for 10,000 evaluations. From the random initialisation of the population, it evolves towards the Pareto optimal front, keeping the front well sampled throughout this process. The final sets of non-dominated solutions in the archives range from 22–36 MLPs. The quality of these fronts is dependent on the MLP configuration (number of neurons in the hidden layer). The best fronts (on the training set) are obtained with 50 and 70 neurons in the hidden layer, which is illustrated in Figure 6.7. This figure plots the performance of the MLPs in the archives of non-dominated solutions for each configuration.

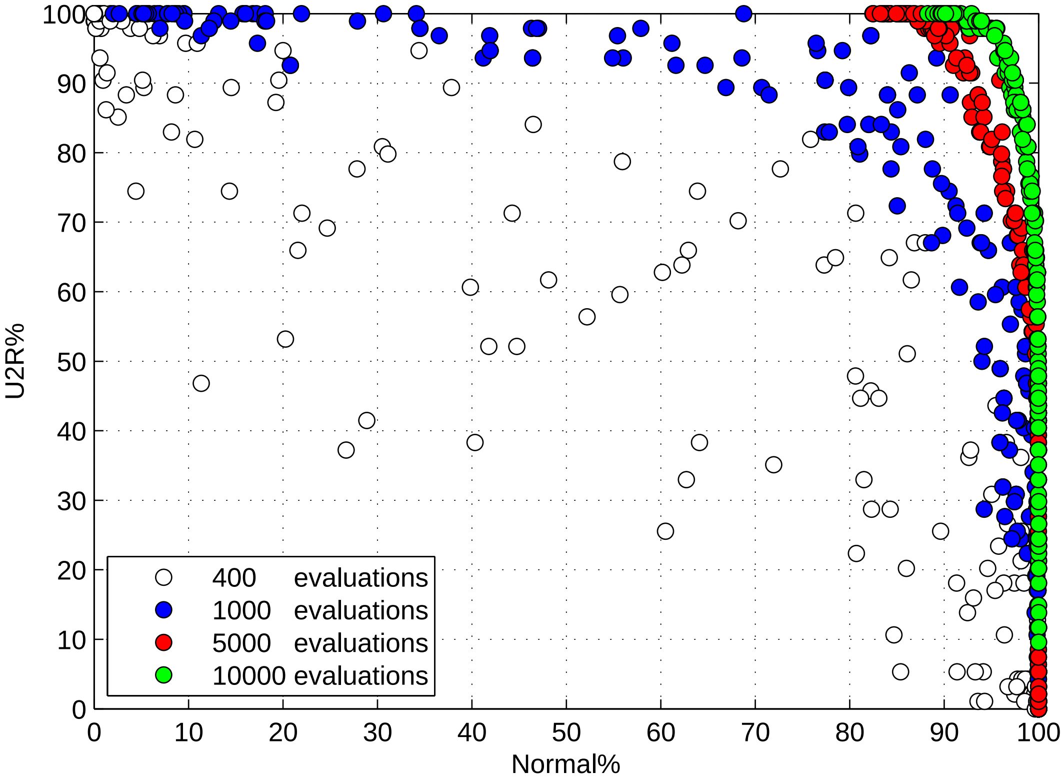
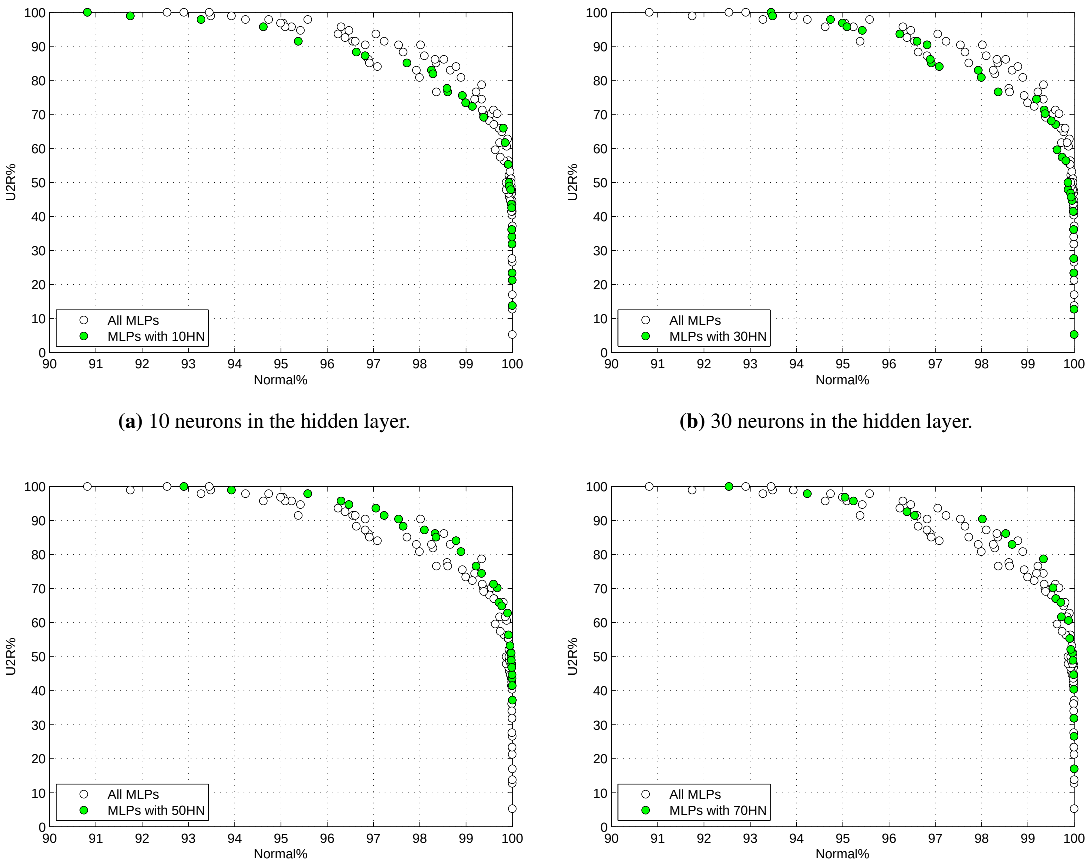


Figure 6.6: Evolution of the population in phase one of MABLE.



(a) 10 neurons in the hidden layer.

(b) 30 neurons in the hidden layer.

(c) 50 neurons in the hidden layer.

(d) 70 neurons in the hidden layer.

Figure 6.7: Best training performance for each MLP configuration according to the number of neurons in the hidden layer.

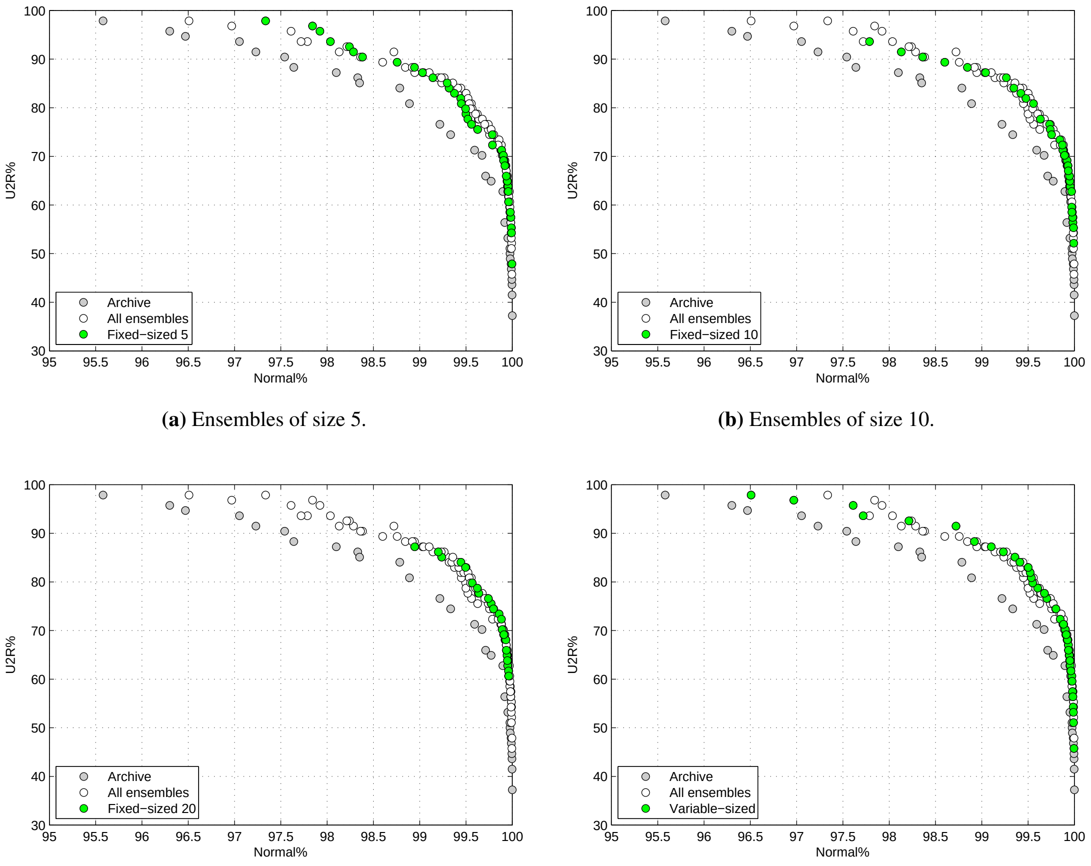
6.5.2 Phase two of MABLE

As discussed in the previous section, the archives of MLPs with 50 and 70 neurons in the hidden layer produced the best Pareto fronts. The archive of 32 MLPs with 50 neurons in the hidden layer was adopted as a pool of base classifiers for ensemble generation. Due to the size of the pool, a maximum ensemble size of 20 was set.

As seen in Figure 6.8, MABLE is able to successfully evolve a new Pareto front of variable and fixed sized ensembles that improve on the performance of the base classifiers in the archive.

For all ensembles evolved by MABLE, both the variable sized and fixed sized, at least two base classifiers are dominated. The mode of dominated classifiers is 4 (the median is also 4, except for size 20, which yielded a median of 3). Both approaches to ensemble generation with MABLE make good use of the archive. For variable sized ensembles (and fixed sized ensembles of size 20), all base classifiers are used in at least one ensemble. For smaller ensemble sizes, fewer base classifiers are used. However, still a high number, 25/32 for size 5 and 31/32 for size 10.

A small fixed sized ensemble size, and variable sized ensembles, are able to obtain the largest Pareto fronts. Based on the trials with each configuration, the Pareto front of the variable sized ensembles ranged from 37–38, and 36–40 for size 5. The larger ensembles, of size 20, gave a set ranging from 21–25. This observation is logical since the larger the ensemble size, the less scope there is to make diverse combinations of the base classifiers. Furthermore, evolving variable sized ensembles gives greater scope for diversity. A



(a) Ensembles of size 5.

(b) Ensembles of size 10.

(c) Ensembles of size 20.

(d) Variable sized ensembles.

Figure 6.8: Performance of fixed sized and variable sized ensembles evolved with MABLE.

larger Pareto front is obtained when evolving variable sized ensembles, although the mode ensemble size is 14 (the median is 12, the smallest size is 3, and the largest size is 19).

Although the Pareto front of ensembles of size 20 is smaller than that obtained with ensembles of size 5, these perform better in a particular region that is illustrated in Figure 6.9 (circled)[6]. Without consideration of the additional computational costs of larger ensembles[7], size 20 may be more desirable if the ideal trade-off is in this region, such as a 0.5% FPR (99.5% *Normal*). However, if a larger FPR is acceptable, ensembles of size 5 offer a larger range of better solutions.

6.5.3 Other combination techniques

Other approaches to combining the classifiers are considered: combining the entire archive, reduced subsets based on clustering, and a new approach based on only combining classifiers that exceed particular performance thresholds.

Table 6.5 provides an overview of the results obtained when combining the entire archive, and reduced subsets according to clustering. As illustrated in Figure 6.10, all but one of these solutions dominate base classifiers. However, only for a very small region. Furthermore, all of these ensembles are themselves dominated by ensembles obtained in the second phase of MABLE.

[6] According to 50 trials, the two single trials illustrated in Figure 6.9 are representative: ensembles of size 5 consistently underperformed in the region highlighted in this figure.

[7] An ensemble of size 5 processes all instances in the test set in 4 seconds compared with 14 seconds with an ensemble size of 20.

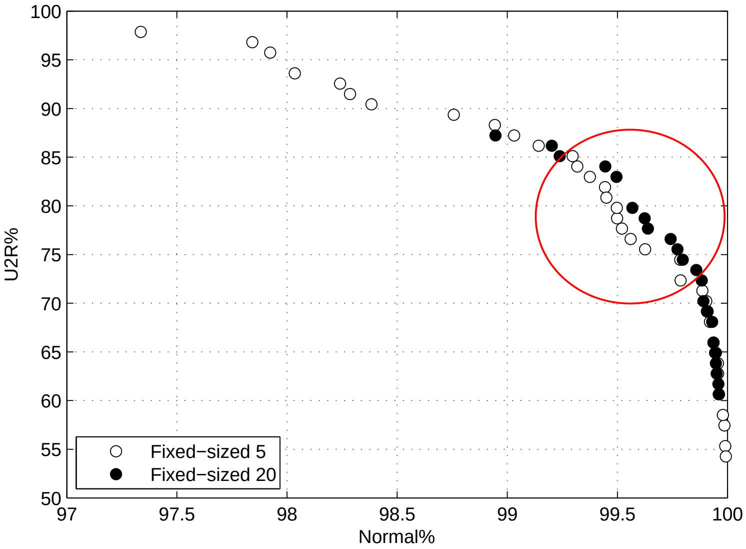


Figure 6.9: Comparison of Pareto fronts obtained with ensembles of size 5 and 20.

Table 6.5: Combination of the entire archive and reduced subsets according to clustering.

k	Normal%	U2R%	TPR%	FPR%
5	99.71	72.34	72.34	0.29
10	99.59	71.28	71.28	0.42
20	99.69	73.40	73.40	0.31
30	99.72	70.21	70.21	0.28
all	99.78	72.34	72.34	0.22

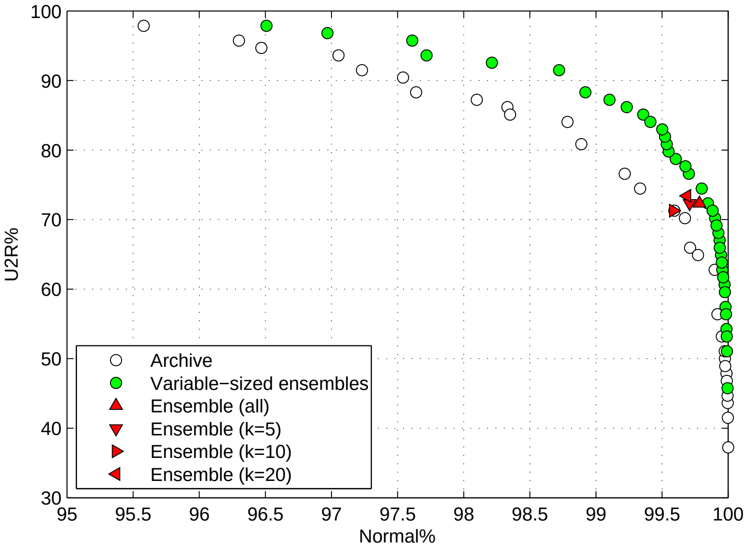


Figure 6.10: Combination of the entire archive and reduced subsets according to clustering.

Including the boundary classifiers when performing clustering does not result in any significantly different performance, as seen in Table 6.6. There is, however, a larger difference in the results obtained when filtering out classifiers according to performance thresholds. Table 6.7 provides an overview of these results. Note that all base classifiers obtained more than 30% detection on both classes, hence, threshold values below 40% are not included.

Table 6.6: Comparison of results with clustering. An asterisk signifies that boundary classifiers are included.

	$k=5$	$k=5^*$	$k=10$	$k=10^*$	$k=20$	$k=20^*$
Normal%	99.71	99.71	99.86	99.58	99.69	99.79
U2R%	72.34	72.34	71.28	73.40	73.40	70.21

Table 6.7: Ensemble performance after filtering out classifiers according to classification thresholds for both classes.

Threshold%	Normal%	U2R%	TPR%	FPR%	Size
40	99.73	73.40	73.40	0.27	31
50	99.05	85.11	85.11	0.95	24

Applying a threshold of 50% results in an ensemble that detects more than 10% more *U2R* instances than an ensemble comprised of MLPs subject to a threshold of 40%. However, the performance decreases on *Normal*, giving a different classification trade-off. This is illustrated in Figure 6.11. The following section sheds more light on this, explaining why the thresholds affect the performance in this way.

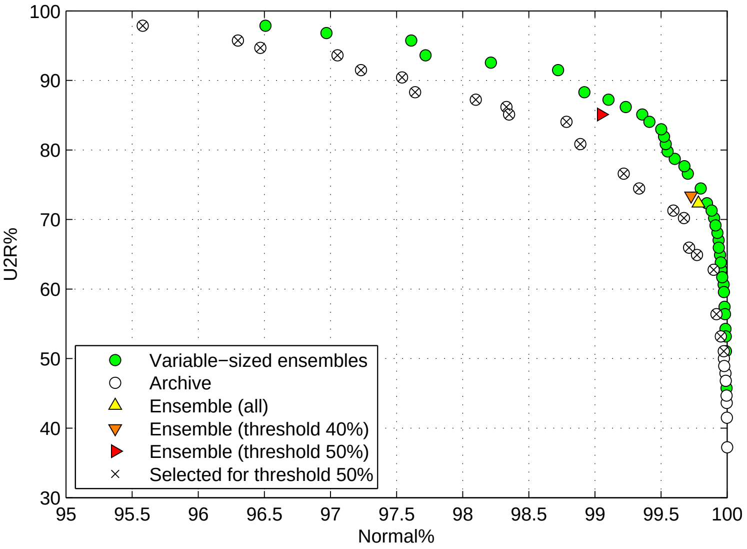
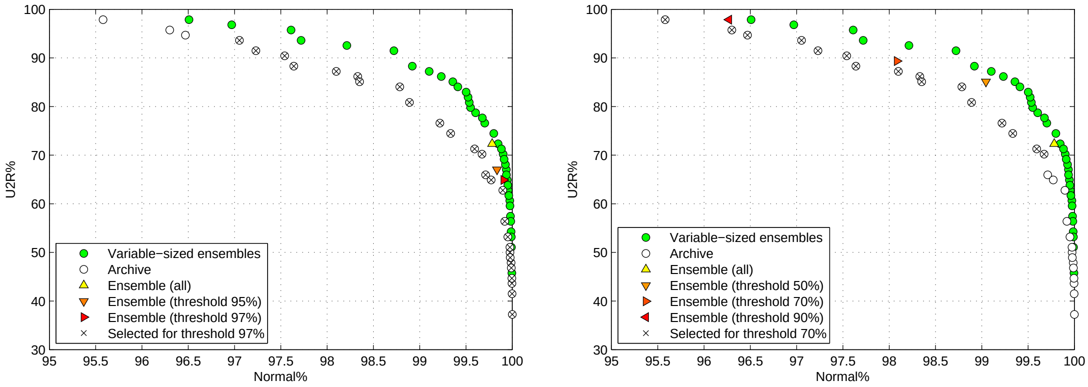


Figure 6.11: Performance of ensembles obtained after filtering out classifiers according to classification thresholds on both classes.

6.5.4 Selection of classifiers

The threshold filtering process biases the performance of the ensembles. The consequence of removing classifiers that obtain less than 50% correct classification on each class is that several classifiers that perform poorly on *U2R* are removed. Therefore, the resultant ensemble is composed of a higher proportion of classifiers that perform better on *U2R*. However, the trade-off is that the classifiers that perform better on *U2R* perform worse on *Normal*, which is reflected in the resultant ensemble. Similarly, thresholds can be applied to a single class to help bias this trade-off, which is illustrated in Figure 6.12.



(a) Based on *Normal*.

(b) Based on *U2R*.

Figure 6.12: Performance of ensembles obtained after filtering out classifiers according to classification thresholds on one class.

Although it is possible to bias the classification trade-off of the ensemble by filtering out classifiers that do not exceed specific classification thresholds, there are two issues with this:

1. Although a classifier performs worse than a defined threshold, it may perform well on instances that the other classifiers make errors on, and, thus, may be a valuable part of an ensemble.

2. Such a process presumes that the bias is induced by a uniform, local set of classifiers.

Figure 6.13 plots the performance of four ensembles (size 5) evolved by MABLE and the base classifiers that they are comprised of. This demonstrates that the selection of classifiers is not uniform, nor local, nor entirely diverse. Therefore, the development of an ensemble exhibiting a desired trade-off is non-trivial, even if that trade-off is known *a priori*. It is also evident that, if an ensemble does not meet the desired trade-off, it may be dominated by one or more base classifiers that offer a better trade-off for the application. Furthermore, with reference to point one, above, it is clear that a simple threshold based approach is insufficient, as a classifier with a lower performance may be valuable. This is particularly clear for ensemble 2 in Figure 6.13b. As can be seen in Figure 6.12, above, the ensembles created from the threshold process are all dominated by the ensembles generated by MABLE.

6.5.5 Generalisation

General observations are discussed in Section 6.5.5.1. Section 6.5.5.2 considers two approaches to improving the generalisation, with a particular focus on determining whether overfitting is the cause of a large decrease in *U2R* performance observed on the test set. This is followed by a detailed analysis of performance on the individual *U2R* attacks in Section 6.5.5.3. This analysis demonstrates that particular attack

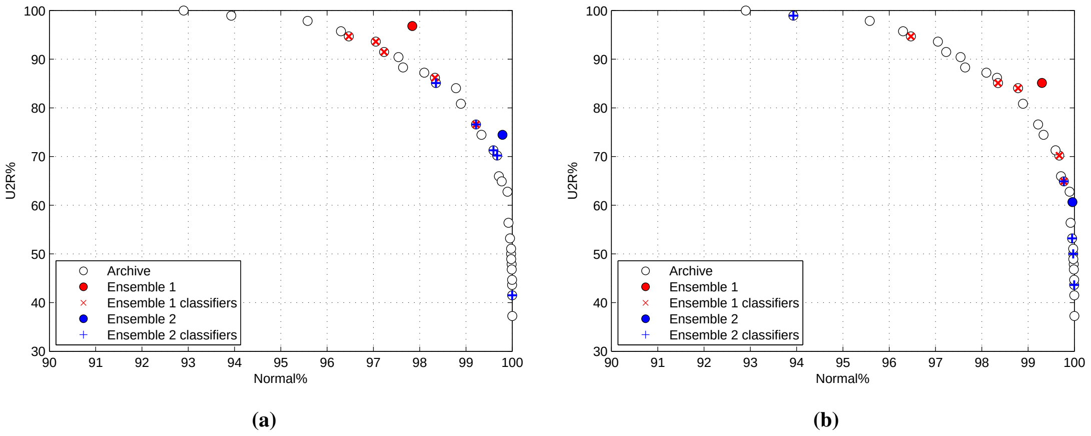
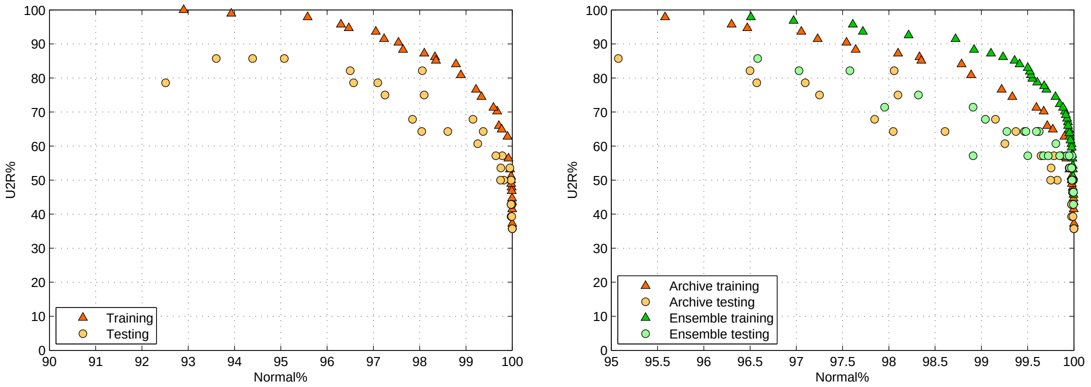


Figure 6.13: Selection of base classifiers in phase two of MABLE.

instances are consistently misclassified. Section 6.5.5.4 delves deeper still, analysing the feature values of the attack instances, to provide indications as to why they are consistently misclassified.

6.5.5.1 General observations

As with any machine learning classifier, the performance on the test set is expected to be poorer than on the training set. Moreover, each classifier is likely to generalise differently, *i.e.*, some may perform better on the test set, whilst others may perform worse. The MLPs (with 50 nodes in the hidden layer) from phase one of MABLE generalised well on *Normal*, but the performance is poorer on *U2R* during testing. The mean difference between the training and test performance of the MLPs is 0.13% on *Normal* and 11.15% on *U2R*. However, despite this larger reduction in *U2R* detection, the performance on the test set follows the performance curve on the training set well, as seen in Figure 6.14a.



(a) Archive of MLPs from phase one.

(b) Variable sized ensembles compared with the archive.

Figure 6.14: Comparison of training and test performance of MLPs (50HN) and ensembles.

The variable sized ensembles exhibit the same classifier behaviour on the test set, performing well on *Normal*, but suffer a decreased detection rate on *U2R*. Figure 6.14b depicts the performance on the training and test sets for both the base classifiers and the ensembles. The mean difference between the training and test performance of the ensembles is -0.041% on *Normal* and 14.2% on *U2R*. The negative difference on

Normal signifies that the test performance is better than the training performance.

According to the single trials discussed above, the ensembles appear to generalise better on *Normal*, but worse on *U2R* than the MLPs. However, as mentioned initially, the classifiers are expected to generalise differently, and there are trials of the algorithms when the performance is significantly better on the test set for some classifiers, whilst the performance is significantly worse for others. Therefore, averages were calculated from 50 trials, which are presented in Table 6.8. Figure 6.17b plots the performance of all base classifiers and ensembles for all trials.

Table 6.8: Average differences between training and test performance, based on 50 trials with MLPs with 50 neurons in the hidden layer.

(a) Archive of MLPs (50HN).

	Mean%	Median%
Normal	1.823	-0.003
U2R	10.511	10.258

(b) Variable sized ensembles.

	Mean%	Median%
Normal	-0.034	-0.026
U2R	14.853	15.198

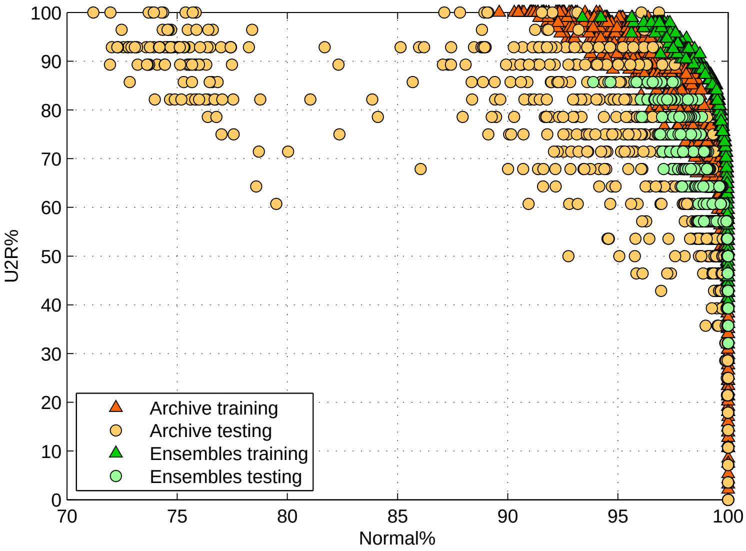


Figure 6.15: Training and test performance from 50 trials, based on an archive of MLPs with 50 neurons in the hidden layer.

Both the MLPs and the ensembles generalise well on *Normal*; the median measure indicates that the test performance is actually better than the training performance[8]. Furthermore, as indicated from the single trials discussed above, the ensembles do generalise better than the base classifiers on *Normal*. However, the median performance on *U2R* is nearly 5% worse for the ensembles (15.198% compared with 10.258%). Despite this, the average performance measures indicate that the ensembles are more robust, whilst the base classifiers have more cases of significantly worse performance on the test set (on *Normal*). This is illustrated well in Figure 6.17b, showing these as outliers, which is reflected in the smaller median of -0.003% compared to the mean difference of 1.823% on *Normal* for the MLPs.

[8] The reason the median measure indicates better performance than the mean is that the majority of the MLPs obtain good performance on the test set, whilst a small proportion detects more than 20% less *Normal* instances correctly. This is illustrated well in Figure 6.17b.

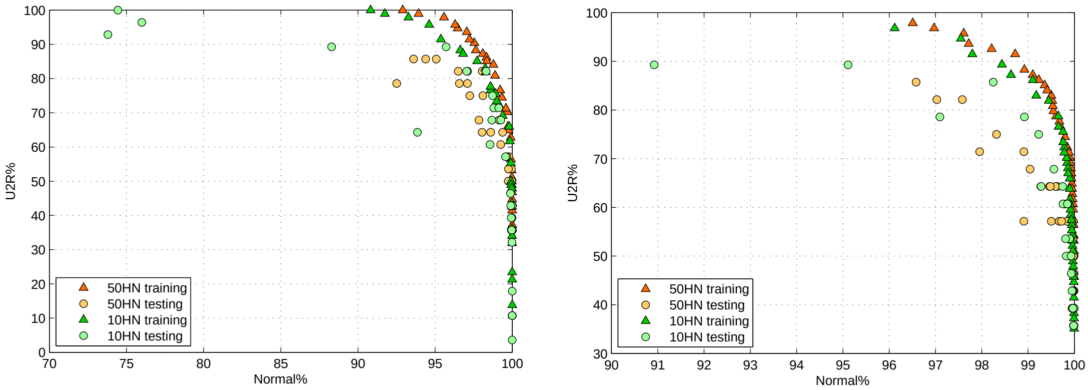
6.5.5.2 Overfitting

The reduced performance on *U2R* during testing may be due to overfitting the training data. However, the results of the ensembles are inconsistent since they generalise better than the base classifiers on *Normal*, but worse on *U2R*. The observations on *Normal* are in accordance with those in the literature, in which several studies have found that creating ensembles is an effective way to improve generalisation (Hansen and Salamon 1990, Zhou *et al.* 2001; 2002). However, the findings on *U2R* are contradictory. Alternatively, the performance reduction may be caused by issues related to the *U2R* class, which is considered further at the end of this section. First, the hypothesis of overfitting is considered, examining the following:

- Shorter training time for the base classifiers.
- Employing smaller networks (fewer neurons in the hidden layer).

Since the training process here is not an exact error based method such as backpropagation, for which overfitting can be directly influenced by the training time, shorter training here may not necessarily reduce overfitting. It may simply produce classifiers that perform worse, but maintain the same generalisation properties. With certainty, however, smaller MLPs will not be capable of as complex fitting of the data, and, therefore, may generalise better. In this case, one would expect to obtain worse training performance, but the difference in test performance may be smaller; indicating better generalisation. Furthermore, it is interesting to examine how well the Pareto front of ensembles composed of smaller base classifiers compares with the Pareto front obtained with 50HN. For these experiments, MLPs with 10HN are considered.

As seen in Figure 6.16a, MLPs with 10HN tend to generalise better than MLPs with 50HN. Although the training performance is slightly worse with 10HN, the test performance is better than that achieved with 50HN. However, there is still an issue with some MLPs performing significantly worse during testing, particularly those that detect more than 90% *U2R* during training.



(a) Archive of MLPs from phase one.

(b) Variable sized ensembles from phase two.

Figure 6.16: Training and test performance of MLPs with 50HN and 10HN, and associated ensembles.

The Pareto front of ensembles using 10HN MLPs as base classifiers is almost as good as that obtained with 50HN MLPs, which is depicted in Figure 6.16b. Furthermore, as with the base classifiers, the testing performance is generally better. Although higher detection of *U2R* is achieved during testing with smaller base classifiers, there are more outliers than when using 50HN MLPs as base classifiers.

Due to an increase in outliers, the mean difference is greater for the MLPs with 10HN, as seen in Table 6.9. However, the median difference on *U2R* is lower: 8.890 compared with 10.258. Although the median difference on *U2R* for the ensembles is almost identical with 10HN MLPs (15.426) compared with 50HN

MLPs (15.198), the performance is not as robust. Nevertheless, as depicted in Figure 6.17a the ensembles based on 10HN MLPs are capable of detecting more *U2R* intrusions. However, these are the outliers, which is why the median difference remains a negative value whilst the mean is now positive (worse performance on the test set).

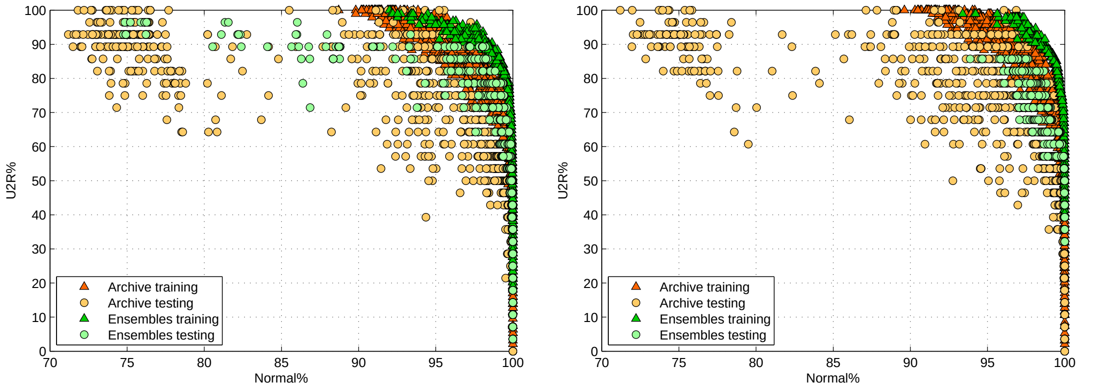
Table 6.9: Average differences between training and test performance, based on 50 trials with MLPs with 10 neurons in the hidden layer.

(a) Archive of MLPs (10HN).

	Mean%	Median%
Normal	2.970	-0.003
U2R	9.626	8.890

(b) Variable sized ensembles.

	Mean%	Median%
Normal	0.281	-0.046
U2R	14.648	15.426



(a) 10HN MLPs.

(b) 50HN MLPs.

Figure 6.17: Training and test performance of MLPs with 50HN and 10HN, and associated ensembles, over 50 trials.

The results discussed above indicate that the smaller MLPs generalise better, although the robustness is compromised. The other factor that may affect generalisation is training time. It should be noted that the training time for the MOGA is already short in comparison to the 20,000 evaluations used in the previous investigation. Furthermore, since that was for single objective optimisation, it would be reasonable to expect that the MOGA would require longer training time since it solves a more complex task. Nevertheless, the MOGA has converged well after 10,000 evaluations, as illustrated in Figure 6.6 on page 129.

Two sets of experiments were conducted, setting the training time to 5,000 and 7,000 evaluations. An overview of the average differences in performance between training and testing is presented in Table 6.10. The median difference in training and test performance on *Normal* remains nearly identical as the training time is reduced, but the mean is increased with 7000 evaluations. The mean difference on *Normal* from 7,000 to 5,000 evaluations is then slightly lower, however. Both the median and mean difference on *U2R* decrease by approximately 1% with each reduction in training time. Although a decrease can indicate improved generalisation, it is unlikely that this is the case here since the observations on the two classes are contradictory. With shorter training, MABLE is not able to obtain as good a Pareto front on the training set as with 10,000 evaluations. However, as seen in Figure 6.18, the test performance is almost the same. Therefore, it is logical that the average difference decreases, since the majority of runs obtain a worse training performance whilst the test performance remains very similar. Furthermore, although the median difference on *U2R* decreases, the mean difference remains very similar, which indicates an increase in outliers. These are all indications of insufficient time to learn and issues with the *U2R* class (which are discussed further below), rather than issues with overfitting.

Table 6.10: Average differences between training and test performance, dependent on training time. Based on 50 trials with MLPs with 50 neurons in the hidden layer.

(a) 10000 evaluations.

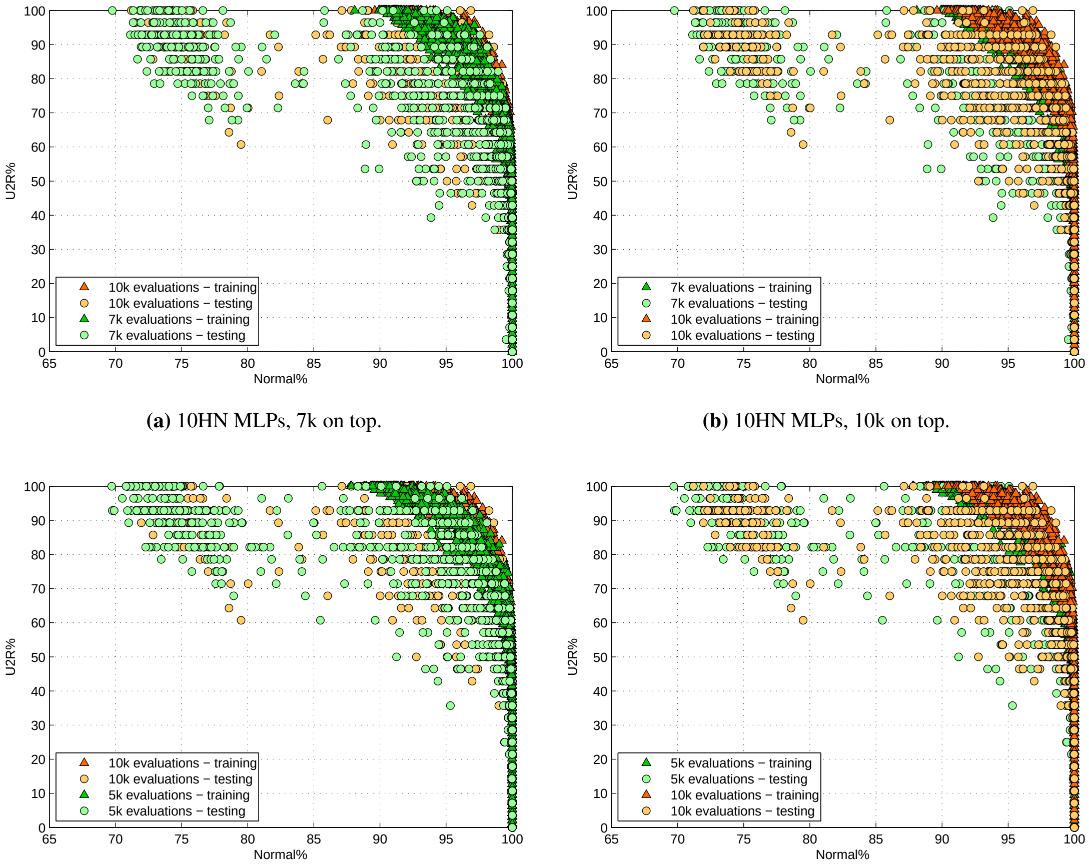
	Mean	Median
Normal	1.823	-0.003
U2R	10.511	10.258

(b) 7000 evaluations.

	Mean	Median
Normal	3.730	0
U2R	9.737	8.967

(c) 5000 evaluations.

	Mean	Median
Normal	3.477	0
U2R	8.651	7.751



(a) 10HN MLPs, 7k on top.

(b) 10HN MLPs, 10k on top.

(c) 50HN MLPs, 5k on top.

(d) 50HN MLPs, 10k on top.

Figure 6.18: Generalisation performance for 50 trials with different number of evaluations.

6.5.5.3 Analysis of U2R attacks

Although there are some changes in the performance when altering the training time and number of neurons in the hidden layers of the MLPs, the difference in test performance on *U2R* remains high. The *U2R* class consists of seven different types of attacks. Three of these attacks have less than 10 instances to learn from; one attack merely provides a single instance to learn from and one to test on. This is a very low number of instances, which becomes an issue if not all seven attacks are similar in their feature values. If they are not, one can expect that the smaller attack classes will not be learned properly. Since a single misclassification on the (small) test set has a larger impact on the classification rate than on the training set, consistently misclassifying a few specific attacks, or attack instances, may be the cause of the large performance decrease on the test set.

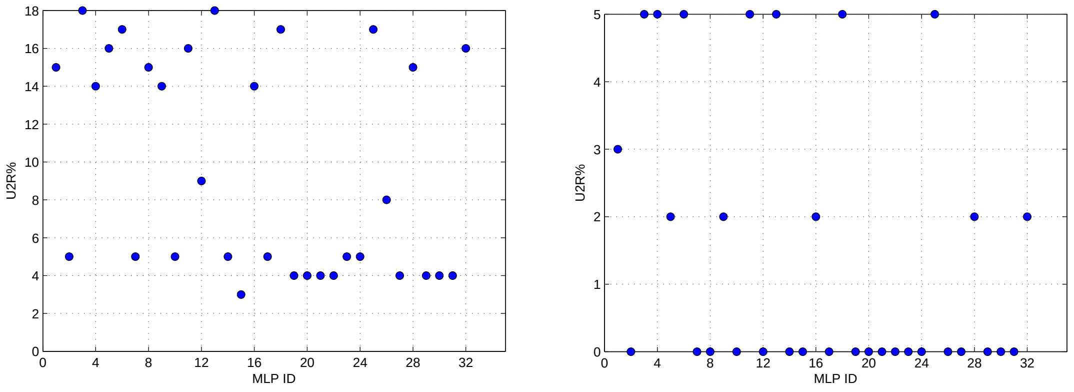
Additional experiments were conducted to obtain classification rates on each of the individual attacks. This was done by evaluating the training and test performance of the existing MLPs and ensembles on subsets of the data set that only contain a single attack type, labelled as *U2R* (as the classifiers have been trained for). For this investigation, ensembles of size 5 were chosen to avoid issues related to potential ties between the ensemble members; since the ties are resolved randomly, two trials of the same ensemble may lead to different classification rates. For reference, an overview of the average differences between the training and test performance for the ensembles are presented in Table 6.11. Since the result listings for these experiments are quite extensive, they are not presented here. Instead, refer to Appendix B.3.1.3 on page 217.

Table 6.11: Average differences between training and test performance of ensembles of size 5. Based on 50 trials with MLPs with 50 neurons in the hidden layer.

	Mean%	Median%
Normal	-0.056	-0.027
U2R	14.335	14.666

The additional experiments with the MLPs reveal that *sqlattack*, which has merely one instance to learn from and one to test on, was always correctly classified. The second attack with only one instance in the test set, *perl*, was only detected by 9 out of 32 MLPs, despite having more instances to learn from (4). Furthermore, these four instances were all learned correctly during training by all but one MLP. Two other attacks were also found to affect the decreased detection during testing: (1) the *ps* attack was never fully detected, with one instance consistently misclassified, and (2) with the exception of one MLP, three instances of *buffer overflow* were consistently misclassified. These four instances account for 14.29% of the instances in the test set, which explains the consistent performance reduction on the test set. Other attacks, however, compensate for this to a degree as they are detected with a higher accuracy, which lowers the median difference (to approximately 10% for the MLPs, as listed in Table 6.8 on page 136).

The *rootkit* attack was also found to have a significant impact on the difference between training and test performance. This attack is largely undetected despite being learned during the training phase. Interestingly, when the attack is detected, generally all instances are then detected, giving ‘extreme’, almost binary, classification behaviour. However, the attack is only detected when at least 14 instances have been correctly learned (77.78% of all *rootkit* instances in the training set). In the case of learning 14 instances from the training set, this accounts for 14.89% of all *U2R* instances in the training set. If all 5 instances are detected during testing, this accounts for 17.85% of the total *U2R* detection. Hence, this gives a difference of approximately 3%. In another typical case, when 14 (or more) instances are learned from the training set, only 2 instances are detected during testing, which leads to a difference of approximately 7%. The extreme classification behaviour on the test set is also seen during training, in which the number of learned instances can be broadly clustered into two groups: (1) learning merely 3–5 instances correctly, or (2) 14–18. Since the classification behaviour is almost binary, it causes some abrupt changes in the test performance, having a significant positive or negative impact on the difference between the training and test performance. Figure 6.19 illustrates the scattered groups on both the training and test sets.



(a) Training. **(b)** Testing.

Figure 6.19: Performance of the MLPs on the *rootkit* attack.

As with the base classifiers, the ensembles exhibit the same classification behaviour on the attacks discussed above. However, the ensembles perform better on the *rootkit* attack during training, leading to a larger difference in performance on the test set; 1.5% based on the trials examined here. Similarly, the ensembles perform better on the *buffer overflow* attack during training, giving an additional 1% larger mean difference. The mean difference on the *U2R* class as a whole, for the trials of the MLPs and ensembles examined here, is approximately 3.5%, as seen in Table 6.12, below. The remaining 1% comes from the *perl* attack. Similarly to the MLPs, the ensembles perform very well on this attack during training; every ensemble correctly learned the four instances in the training set. However, only one ensemble was able to detect the single attack instance in the test set.

Table 6.12: Differences between training and test performances of MLPs and ensembles.

(a) Archive of MLPs.

	Mean%	Median%
Normal	1.823	-0.003
U2R	10.511	10.258

(b) Ensembles (size 5).

	Mean%	Median%
Normal	-0.056	-0.027
U2R	14.335	14.666

The observations discussed above give explanations as to what caused the large difference between the training and test sets on the *U2R* class compared with the *Normal* class. There is an additional factor that affects this difference: the proportion of instances of each attack in the training and test sets. Due to the limited number of instances of most *U2R* attacks, it is not possible to partition the data to assign exactly 80% of the instances to the training set and the remaining 20% to the test set. For example, *sqlattack* has one instance in the training set and one instance in the test set, which makes up 1.06% of all *U2R* training instances and 3.57% of all *U2R* test instances. Therefore, even when the both the training and test instances are detected, *i.e.*, 100% detection of this attack, this gives a difference of 2.51%. Table 6.13 provides an overview of the proportions of each *U2R* attack in the training and test sets.

Table 6.13: Proportion of each *U2R* attack in the training and test sets.

	buffer overflow	loadmodule	perl	ps	rootkit	sqlattack	xterm
Training%	43.62	8.51	4.26	12.77	19.15	1.06	10.64
Test%	39.29	10.71	3.57	14.29	17.86	3.57	10.71

6.5.5.4 Analysis of feature values of the U2R attacks

An analysis of the specific instances being misclassified provides some indication as to why three *buffer overflow* attacks and one *ps* attack are consistently misclassified. Considering the features of the three *buffer overflow* attacks, three of these are particularly suggestive as to why they are misclassified: (1) the *duration* is greater than the other instances, both in the training and test sets; (2) the *number of files created* is greater, 4–6, whilst it is mostly 0 or 1 for other instances; (3) the value for *hot* is lower for the misclassified instances compared with other instances in the test set (0–2 compared with 3–10). For *hot*, there are several training instances with similar values, however, so this may or may not be a contributing factor.

The *ps* instance that is misclassified is nearly identical to one instance in the training set for many features. However, the values of four features are significantly higher: *destination host count* (255 compared with 29), *destination host server count* (172 compared with 3), *destination host same server* (0.67 compared with 0.1); *destination host server error* (0.94 compared with 0). For the latter feature, there are other instances in the training and test sets that have similar values, but very few; only two instances did not have a 0 value. There are instances with similar feature values for *destination host same server*, but the values of other features are different.

Since the single *perl* instance in the test set is not consistently misclassified, it is likely that its classification is dependent on correctly learning other attack instances that are similar. However, there are some feature values that stand out as different within the instances of the *perl* attack: *root shell*, *num root*, and *destination host count*. The latter feature was also found to reveal some differences between the training and test instances for the *ps* attack, as discussed above. In this case, the *destination host count* is much lower for the test instance compared with those in the training set. The two former features, *root shell* and *num root*, are both zero/false for the test instance for the *perl* attack, whilst all training instances are based on *root shell* being true and a *num root* count of 2.

As discussed above, the *rootkit* attack also exhibits quite 'extreme' classifier behaviour on the training set; either learning correctly most instances, or very few. In the cases where three of the five test instances are misclassified, it is largely three specific instances. Most strikingly, these instances target the *ftp_data* service and have a very low *duration*. The two other test instances target the *ftp* service. Furthermore, the three instances have low *source* and *destination byte* values, and a high *destination host server count*, compared with the other instances.

The *rootkit* attack targets several protocols and services. Three instances from the training set that are commonly not learned target the *udp* protocol, whilst all other instances target the *tcp* protocol. Furthermore, the service these three instances target is defined as *'other'*, which no other rootkit instances target. Another instance that is often missed from the training set targets the *ftp_data* service, which is the same service that is targeted by the three instances in the test set that are misclassified the most.

The analysis of the feature values of the attack instances only provides possible indications of what may have caused the misclassifications. Since the MLPs are generally treated as 'black boxes', it is difficult to confirm that these observations are indeed the contributing factors in the misclassifications. Although there are seemingly evident differences between feature values of certain training and test instances, some of these features may not even be used in the classification process; *i.e.*, the weights associated with the features may yield approximately the same output regardless of input value.

6.5.6 Complexity and speed

The findings discussed in the previous section demonstrate that ensembles composed of smaller MLPs (fewer neurons in the hidden layer) can achieve the same performance as ensembles composed of larger MLPs. In some cases, they even appear to provide better performance as more *U2R* instances were detected by some ensembles. There is an additional advantage of employing an ensemble of smaller MLPs: it reduces the time required to process the data. This is important to factor to consider in an IDS, which aims to achieve real-time intrusion detection. Although this is not the specific focus of this investigation, it is nonetheless interesting to compare the computation times. As shown in Table 6.14, below, the computational cost of 50HN MLPs compared with 10HN MLPs is approximately three times as high[9]. Although larger ensemble sizes imply higher computational costs, the majority vote combiner adopted here allows for convenient parallelisation, in which each base classifier could be executed independently, passing on the vote to a master decision module.

Table 6.14: Training and test times for the MLPs and ensembles.

	MLPs		Ensembles size 5		Ensembles size 20	
	Training	Testing	Training	Testing	Training	Testing
10HN	< 1s	< 1s	4s	1s	18s	4s
50HN	3s	1s	14s	3s	56s	11s

Related to generalisation, it is interesting to determine whether the ensemble size has an impact on the performance and generalisation ability of the ensembles. It was not straightforward to determine this according to the findings with the variable sized ensembles from the previous section; in some cases, smaller ensembles lead to large differences in classification rates, whilst in other cases, this was caused by larger ensembles. An additional experiment was therefore conducted to measure the average performance of ensembles of size 5, 10 and 20, using 10HN and 50HN MLPs as base classifiers, respectively. Averages are presented in Table 6.15 for ensembles with 10HN MLPs and Table 6.16 for ensembles with 50HN MLPs. The performance of each ensemble configuration is depicted in Figure 6.20.

Table 6.15: Average differences between training and test performance, based on 50 trials with MLPs with 10 neurons in the hidden layer.

(a) Ensemble size 5.

	Mean	Median
Normal	0.629	-0.022
U2R	13.847	14.894

(b) Ensemble size 10.

	Mean	Median
Normal	-0.040	-0.041
U2R	15.735	16.033

(c) Ensemble size 20.

	Mean	Median
Normal	-0.074	-0.048
U2R	15.478	15.881

[9] It should be noted that the computation times quoted here are indicative only. The code has not undergone any specific investigation to identify ineffective code ‘hot spots’ that may render these times unrepresentative.

Table 6.16: Average differences between training and test performance, based on 50 trials with MLPs with 50 neurons in the hidden layer.

(a) Ensemble size 5.

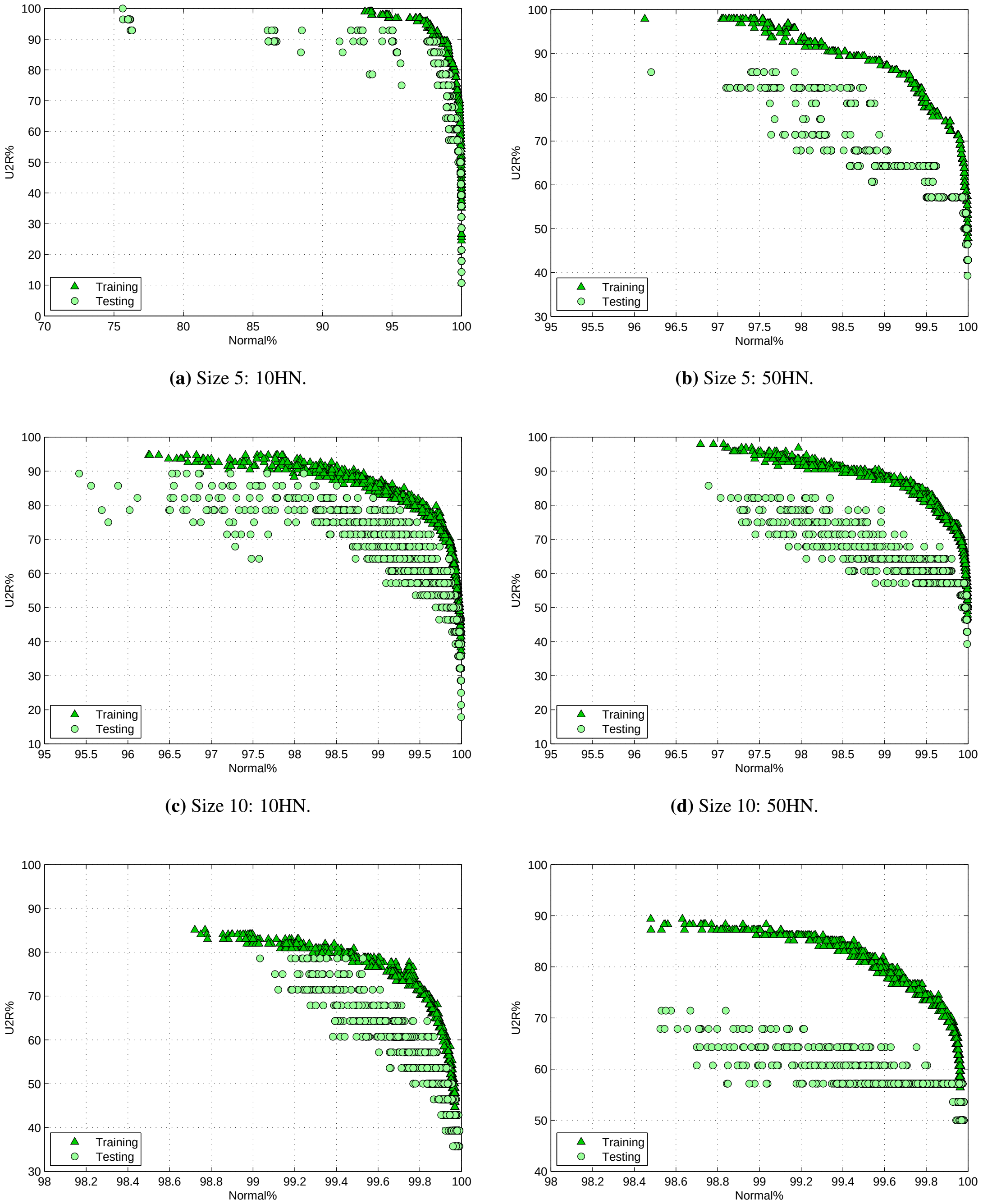
	Mean	Median
Normal	-0.056	-0.027
U2R	14.335	14.666

(b) Ensemble size 10.

	Mean	Median
Normal	-0.024	-0.020
U2R	15.416	15.578

(c) Ensemble size 20.

	Mean	Median
Normal	-0.035	-0.027
U2R	16.623	16.261



(a) Size 5: 10HN.

(b) Size 5: 50HN.

(c) Size 10: 10HN.

(d) Size 10: 50HN.

(e) Size 20: 10HN.

(f) Size 20: 50HN.

Figure 6.20: Generalisation performance of fixed size ensembles for 50 trials.

All ensemble configurations but one (size 5 with 50HN MLPs) obtained better performance on the test set on *Normal*. There are some differences in the results obtained with the ensembles comprised of 10HN MLPs compared with those comprised of 50HN MLPs. As demonstrated earlier, the smaller ensembles (size 5) comprised of 10HN MLPs produce several solutions that appear as outliers with significantly worse performance on *Normal*, resulting in a worse mean measure of difference between the training and test performance (up to approximately 20% worse on the test set). However, the largest ensembles (size 20) comprised of 10HN MLPs actually led to better test performance than ensembles of the same size comprised of 50HN MLPs.

A general trend that can be observed is that the detection of *U2R* decreases as the ensemble size is increased; particularly with 50HN MLPs. The ensembles comprised of the smaller MLPs (10HN) maintain a lower difference between the training and test performance. Although large ensembles of the larger MLPs (50HN) produce a better Pareto front on the training set (than with 10HN MLPs), the test performance is worse than with smaller base classifiers, which is depicted clearly in Figures 6.20e and 6.20f. Therefore, there is a trade-off between the ideal ensemble configuration in terms of size and complexity of the base classifiers, and how this affects the computational costs and performance.

As mentioned above, in terms of performance, large ensembles of small MLPs, or small ensembles of larger MLPs, are fruitful. If parallel implementations are possible, the former solution may be desirable, as it would require less computational costs to process the data by each classifier. However, the latter achieves the highest classification rates and obtains a larger Pareto front.

6.5.7 Performance on three class data

The purpose of this investigation is to compare the performance of the MLPs and ensembles evolved with MABLE with other popular classifiers; namely a Decision Tree (DT), Random Forest (RF) and AdaBoost. The results of these tree classifiers are discussed first, in Section 6.5.7.1. Thereafter, the findings from phase one and two of MABLE are discussed in Sections 6.5.7.2 and 6.5.7.3, respectively. This investigation also gives indications of how well MABLE scales with an additional class/objective.

6.5.7.1 Tree classifiers

The performance of the DT varies insignificantly when examining the confidence factors for pruning. However, the RF and AdaBoost performance do vary with different configurations, and more significantly from trial to trial. Both RF and AdaBoost improve on the performance of the DT, particularly in detecting the intrusions. However, the results obtained with these classifiers are biased towards very high classification of *Normal*. Table 6.17 provides an overview of the results obtained with two RFs and the best DT.

Table 6.17: Results of the DT and RF.

	Accuracy%	TPR%	FPR%	Normal%	U2R%	R2L%
DT	94.47	10.97	0.04	99.96	18.57	10.24
RF1	94.57	12.18	0.02	99.98	30.00	11.38
RF2	94.59	12.76	0.04	99.96	28.57	12.07

There is no clear trend in the performance of the RFs dependent on the number of trees. There are slight performance differences from trial to trial and with different number of trees. For example, RF1 in Table

6.17 obtained the best performance on *U2R*, which was achieved with 30 trees. RF2, performing best on all aspects but FPR, was comprised of 5 trees. This gave the highest FPR of all RFs.

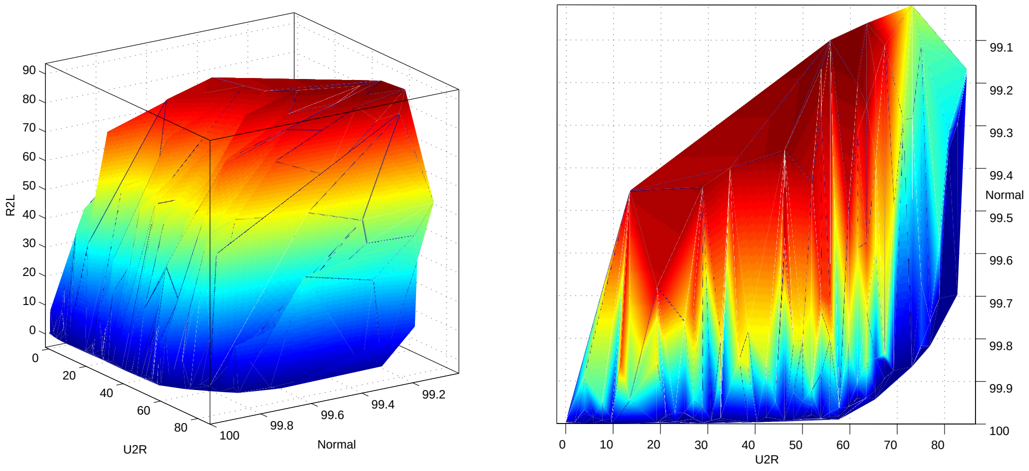
Similarly to the RF, the less complex AdaBoost solutions provide the highest TPR and accuracy. AdaBoost is able to detect more *R2L* intrusions than RF, which is the main contributor to a higher TPR; see AdaBoost1[10] in Table 6.18. There is more variation in the classification rates from trial to trial with AdaBoost. However, this is less significant with a weight threshold of 100, which produces very large trees[11] that obtain the lowest FPR, but perform consistently worse on the intrusion classes, indicating overfitting; see AdaBoost3[12] in Table 6.18. As with RF, all AdaBoost solutions obtain merely 0.02–0.04% false positives.

Table 6.18: Different classification trade-offs obtained with AdaBoost.

	Accuracy%	TPR%	FPR%	Normal%	U2R%	R2L%
AdaBoost1	95.28	24.36	0.03	99.97	14.29	23.74
AdaBoost2	94.74	15.41	0.03	99.97	37.14	14.32
AdaBoost3	94.85	16.94	0.02	99.98	24.29	16.19

6.5.7.2 MABLE phase one

MABLE successfully evolves the MLPs and ensembles thereof on this three class data. Although the results are not as easy to analyse as for the two class data, Figure 6.21 shows a 3D plot of the performance of MLPs with 70HN, which have minimally 1% false positives. A contour view is presented in Figure 6.21b, with axes for *Normal* and *U2R* detection, whilst the highest detection rate of *R2L* is depicted in red, following the same colour grading as in Figure 6.21a.



(a)

(b)

Figure 6.21: Training performance of MLPs from phase one of MABLE.

The MLPs evolved in the first phase of MABLE offer a large range of solutions with different trade-

[10] AdaBoost1 is configured with a weight threshold of 10, and 10 iterations.

[11] The size of the trees range from 125–169, with 63–85 leaf nodes. In comparison, the size of the trees obtained with a weight threshold of 10 range from 15–19, with 8–10 leaf nodes.

[12] AdaBoost 3 is configured with a weight threshold of 100, and 10 iterations.

offs. With the additional objective, the number of solutions in the Pareto front increases significantly. The archive size is between 300 and 400, which is up to twice as large as the population size adopted here. Table 6.19 presents a selection of MLPs, ordered by FPR.

Table 6.19: Test performance of a range of MLPs evolved in phase one of MABLE.

Accuracy%	TPR%	FPR%	Normal%	U2R%	R2L%
94.62	14.55	0.12	99.88	32.86	13.67
95.01	36.19	0.90	99.10	35.71	32.44
92.49	41.05	2.87	97.13	80	20.14
89.42	60.23	8.11	91.89	65.71	51.28
87.70	52.85	8.68	91.32	78.57	31.43
85.02	50.90	11.47	88.53	81.43	30.18
82.06	70.05	16.78	83.22	50	64.65
66.53	93.64	34.21	65.79	62.86	78.09
40.94	99.81	62.79	37.21	38.57	99.31
32.71	100	70.65	29.35	48.57	84.93
16.98	99.81	87.76	12.24	57.14	90.22
11.60	99.78	93.57	6.43	71.43	91.20

The tree classifiers discussed in the previous section are biased towards the major class, giving a high classification of *Normal* and a low TPR. Assuming that a 1% FPR is acceptable (as discussed in Section 6.4.7 on page 127) this trade-off is not ideal. As seen in Table 6.19, above, a higher TPR is possible to achieve with the evolved MLPs at an acceptable increase in FPR. For example, at approximately 1% FPR (99.10% detection of *Normal*), it is possible to detect more than 30% *U2R* and *R2L* intrusions, leading to a TPR of 36.19%. This is more than 20% higher TPR than the DT and RF, as seen in Table 6.20, below. Moreover, this MLP obtains a 95.01% accuracy, compared with 94.59% for the best RF. One of the AdaBoost solutions obtains a slightly higher accuracy (0.27% higher) and lower FPR, but the TPR is more than 10% lower. Another AdaBoost solution offers a different trade-off, detecting more *U2R* intrusions, but less *R2L*, which leads to a significantly lower TPR. Therefore, despite AdaBoost having a potential advantage of weighted combination of DT classifiers, the solutions are inferior to some of the MLPs evolved by MABLE.

Table 6.20: Comparison of results from the tree classifiers and an MLP evolved in phase one of MABLE.

	Accuracy%	TPR%	FPR%	Normal%	U2R%	R2L%
DT	94.47	10.97	0.04	99.96	18.57	10.24
RF1	94.57	12.18	0.02	99.98	30.00	11.38
RF2	94.59	12.76	0.04	99.96	28.57	12.07
AdaBoost1	95.28	24.36	0.03	99.97	14.29	23.74
AdaBoost2	94.74	15.41	0.03	99.97	37.14	14.32
MABLE1	95.01	36.19	0.90	99.10	35.71	32.44

6.5.7.3 MABLE phase two

The base classifiers (MLPs) evolved in the first phase of MABLE already perform well compared with the tree classifiers, producing solutions with more fruitful classification trade-offs. The archive of MLPs with 70 neurons in the hidden layer performed best, giving the largest number of 'good' solutions, and were used as base classifiers for phase two. Due to the large archive sizes on this three class problem, a filtering process was applied to help focus on the solutions that are likely to be competitive with those already discussed. Therefore, according to the findings discussed above, the following thresholds were employed: 99% on *Normal*, 30% on *U2R* and 10% on *R2L*. It should be noted that these thresholds were not employed in MABLE when evolving the ensembles; these thresholds were simply utilized to help filter out ensembles with unfruitful classification trade-offs *a posteriori*.

A larger proportion of the archive of solutions evolved in phase two fulfil these threshold compared with those in phase one: 179 of 324 variable sized ensembles (of maximum size 20) pass the filter, compared with 30 of the 347 MLPs. This is a good indication that the solution quality of the ensembles is indeed better than the base classifiers. Furthermore, the larger the ensemble size, the more solutions pass the filter. This can be explained by the analysis on the two class data in Section 6.5.2 on page 130; demonstrating that the Pareto front of larger ensembles is smaller and more 'central'. The number of ensembles in the Pareto front is 292 with size 20, compared with 388 with size 5.

The median number of base classifiers that the variable sized ensembles dominate is 22 (mode 11, minimum 0 and maximum 53). The ensemble sizes range from 3 to 20. A maximum size 20 was set for these experiments, but size 50 has also been examined, which is discussed further below. Evolving small fixed sized ensembles dominate fewer base classifiers; the median is 12 for size 5. However, at size 10 and 20, the median number of dominated base classifiers is nearly the same as for the variable sized; 21 and 23, respectively. Although an ensemble size of 5 was sufficient for the two class data set, these findings indicate that it is not for this three class data set. The median ensemble size is 17 when evolving variable sized ensembles with a maximum size of 20. Table 6.21 gives an overview of the performance of a selection of these variable sized ensembles, giving different classification trade-offs.

Table 6.21: A selection of variable sized ensembles from phase two of MABLE. The best individual rates are highlighted in bold.

Accuracy%	TPR%	FPR%	Normal%	U2R%	R2L%
94.60	14.16	**0.10**	**99.90**	31.43	13.21
94.88	20.08	0.20	99.80	32.86	19.20
94.83	22.90	0.40	99.60	32.86	29.82
95.19	31.07	0.50	99.50	41.43	28.91
95.48	37.18	0.66	99.34	34.29	36.43
94.64	27.62	0.73	99.27	**48.57**	23.35
95.46	**39.58**	0.81	99.19	38.57	**38.42**
95.22	38.43	0.94	99.06	42.86	36.33

The maximum ensemble size was constrained to 20 for the results discussed above, since small ensembles are desirable for this application domain to enable real time intrusion detection. Although this aspect is not the focus of this investigation, it is generally interesting to demonstrate how MABLE performs with such constraints. In comparison, experiments were also conducted with a maximum size of 50, to determine whether performance gains are possible.

The archive size is reduced when setting the maximum ensemble size to 50, remaining below 250, which is nearly 100 less than when a maximum size of 20 was set. However, the quality of these ensembles is similar to the larger fixed sized ensembles, giving a larger proportion of solutions passing the threshold. There are fewer solutions ‘at the boundaries’, which can be observed as lower maximum individual classification rates obtained in the entire archive. Although an ensemble size of 50 was allowed, the maximum evolved ensemble size is 44, with a median size of 33. The median number of base classifiers the ensembles dominate is 26, which is greater than those discussed above (with a maximum size 20).

The performance of the other combination approaches was somewhat more successful on this data set compared with those on the two class data, producing several ensembles with good trade-offs among the classification rates. Results from combining all 347 classifiers and reduced sets according to clustering are presented in Table 6.22. The highest accuracy is obtained when combining all classifiers. However, this is merely 0.03% higher than combining a subset of 20 classifiers, which arguably obtains a better trade-off for this application; exhibiting a slightly higher FPR (0.83% compared with 0.73%), but detecting more intrusions.

Table 6.22: Combination of the entire archive and reduced subsets according to clustering.

k	Accuracy%	TPR%	FPR%	Normal%	U2R%	R2L%
5	93.62	14.61	1.08	98.92	37.14	11.87
10	94.28	31.68	1.51	98.49	31.43	29.86
20	95.08	36	0.83	99.17	44.29	32.28
30	94.84	32.10	0.90	99.10	44.29	29.33
40	95	30.37	0.69	99.31	37.14	28.88
50	94.87	27.33	0.63	99.37	41.43	25.67
all (347)	95.11	33.22	0.73	99.27	38.57	31.43

Results for the other approaches to combining the archive, based on clustering and threshold filtering, are similar to those obtained on the two class data. Therefore, these results are only discussed in brief here, whilst all results are available in Appendix B.3.2 on page 221. The ensembles obtained with clustering when including the boundary solutions maintain a lower FPR (all below 1%). Most of these solutions merely give a slightly different classification trade-off, except for the smallest ensemble sizes (size 5 and 10). This is most prominent for size 5, which, when including boundary solutions, gives an ensemble size of 11. This is more than doubling the ensemble size, and, as discussed above, ensembles of size 10 and above do perform better.

Removing classifiers that do not obtain more than 0% correct classification on all classes reduced the pool of classifiers from 347 to 333. The results are, as expected, very similar. With increasing threshold values, the FPR and TPR increase, as was observed on the two class data. Furthermore, filtering out classifiers based only on the *Normal* class does bias the results towards lower FPR, however, at the expense of the TPR. For example, a class threshold of 97% on *Normal* yields a pool of 162 classifiers, achieving the lowest FPR of 0.07%. However, the TPR is merely 1.37% since only 0.49% *R2L* instances were classified correctly.

Although combining the entire archive, or subsets based on clustering, led to some ensembles with good classification trade-offs (close to 1% FPR), they do not perform as well as ensembles evolved by MABLE. This is due to the limitation of performing uniform selection, as discussed in Section 6.5.4 on page 134. An overview of results obtained with all classifiers and ensemble approaches is presented in Table 6.23[13].

Table 6.23: Comparison of the results of all classifiers.

	Accuracy%	TPR%	FPR%	Normal%	U2R%	R2L%
DT	94.47	10.97	0.04	99.96	18.57	10.24
RF #1	94.57	12.18	**0.02**	**99.98**	30.00	11.38
RF #2	94.59	12.76	0.04	99.96	28.57	12.07
AdaBoost #1	95.28	24.36	0.03	99.97	14.29	23.74
AdaBoost #2	94.74	15.41	0.03	99.97	37.14	14.32
MABLE1	95.01	36.19	0.90	99.10	35.71	32.44
MABLE2 #1	95.22	38.43	0.94	99.06	42.86	36.33
MABLE2 #2	94.85	33.34	0.87	99.13	**51.43**	28.88
MABLE2 #3	95.46	**39.58**	0.81	99.19	38.57	**38.42**
MABLE2 #4	**95.48**	37.18	0.66	99.34	34.29	36.43

6.6 Discussion

The proposed approach, MABLE, successfully learns from imbalanced data and offers the user a set of solutions that exhibit different trade-offs in classification performance. The first phase of MABLE provides an *implicit* mechanism for creating a *diverse* pool of classifiers that can be combined to form classifier ensembles. Ensembles evolved in the second phase of MABLE provide an improved Pareto front of solutions. A particular benefit of the multi-objective approach taken here is that a good approximation of the true Pareto front of non-dominated solutions is obtained in a single run.

It is clear that the classification trade-off needs to be taken into account both when creating classifiers and ensembles. The implications of not doing so were indicated in the previous chapter, observing that the ENN would occasionally obtain solutions with too great a false positive rate. Furthermore, the other approaches adopted here for comparison of results, DT, RF and AdaBoost, all offer undesirable classification trade-offs. Although the false positive rates obtained with these techniques are very low, the trade-off is a lower true positive rate than MLPs and ensembles evolved with MABLE were able to achieve. Since practical applications may allow a greater false positive rate, another solution may offer an acceptable trade-off that obtains a higher true positive rate. Following statements in the literature that a 1% false positive rate is acceptable for an IDS (Hu *et al.* 2003, Maxion and Townsend 2002), even MLPs offered better solutions than the popular classifier combination approaches adopted here for comparison (RF and AdaBoost).

The Pareto analysis conducted in this thesis offers a novel perspective on the selection process of ensemble members. Although combining all classifiers in the Pareto front, or reduced subsets according to clustering or thresholding, produced ensembles that yielded improved performance compared with a small number of base classifiers, these were consistently dominated by ensembles evolved with MABLE. An analysis of the selection performed by MABLE demonstrates that the selection of classifiers is not uniform, nor local, nor entirely diverse. Therefore, the development of an ensemble exhibiting a desired trade-off is non-trivial, even if that trade-off is known *a priori*. It is also evident that, if an ensemble does not meet the

[13] MABLE2 #1 was evolved as a variable sized ensemble (size 18).
MABLE2 #2 was evolved as a fixed sized ensemble (size 10).
MABLE2 #3 was evolved as a variable sized ensemble (size 17).
MABLE2 #4 was evolved as a variable sized ensemble (size 16).

desired trade-off, it may be dominated by one or more base classifiers that offer a better trade-off for the application.

CHAPTER 7

Conclusions

A summary of the research conducted for this thesis is provided in Section 7.1, followed by the findings and conclusions in Section 7.2. A discussion of the work conducted is provided in Section 7.3. The novelty and contributions of this thesis are outlined in Section 7.4. Potential further work is discussed in Section 7.5.

7.1 Summary

The review of Artificial Intelligence (AI) applied to intrusion detection, conducted here, uncovered discrepancies in the findings reported in the literature. Numerous studies have adopted the same machine learning techniques and evaluated them on the KDD Cup ’99 data set, but have obtained different results. In some cases, the results are contradictory. This formed the focus of the first part of this thesis, investigating empirically the underlying causes of the discrepancies.

Several methodological factors were found to affect significantly the results in the first part of the thesis, one of which was selected for further investigation in the second part of the thesis: learning from imbalanced data. An empirical investigation was conducted to demonstrate that this is indeed a critical challenge to intrusion detection, and one of the main reasons that Multi Layer Perceptrons (MLPs) have been reported in the literature to be incapable of detecting a particular class of intrusion.

In the second part of the thesis, a Genetic Algorithm (GA) was proposed to optimise the weights of MLPs to better learn from imbalanced data. This approach, referred to as an Evolutionary Neural Network (ENN), was found to be successful, being able to detect the previously undetectable class of intrusion. However, one limitation of the approach was identified. Since single objective optimisation is performed, there is no control of the classification trade-off that the resultant MLP obtains. Furthermore, it has been recognised here that this is a limitation in all approaches that strive to obtain a single ‘best’ solution, typically based on a single measure of accuracy.

To address the limitations of the ENN, the approach was extended to perform multi-objective optimisation. Treating the classification rate on each class as a separate objective, a Multi-Objective GA (MOGA) is capable of evolving a set of non-inferior MLPs that exhibit different classification trade-offs. From this, a user may select the ideal solution. This process is treated as phase one of an approach referred to as MABLE (**M**ulti-Objective Evolution of **A**rtificial Neural Network Ensem**ble**s), which, in phase two, evolves classifier ensembles based on the MLPs evolved in the first phase. A MOGA was also adopted for phase two, which provides an improved set of solutions that exhibit different classification trade-offs. Furthermore, this final part of the thesis offered a novel perspective on the selection of ensemble members, demonstrating why common approaches fail to provide fruitful solutions, since the classification trade-off is not considered.

7.2 Findings and conclusions

This section presents the findings and conclusions related to the aims of the three empirical parts of this thesis.

7.2.1 Part one – discrepancies

Aim 1.1: Determine causes of discrepancies in the literature

The initial analysis of the published work indicated that the main cause of the discrepancies is due to researchers adopting different subsets of the KDD Cup '99 data set. Some researchers adopt the original training and test sets, whilst others adopt subsets that change significantly the classification task, and, therefore, obtain better results. The importance of these findings is two-fold: first, the perils of comparing results and findings across different studies are evident; second, it is clear that there is a need for deeper consideration of methodological factors in empirical studies.

Several methodological factors that affect the results have been identified, and their impact on the findings have been investigated empirically:

Subsets: As mentioned above, the choice of data partition affects significantly the results. This is not the underlying cause; other methodological factors directly affect the results. The impact of these factors differ between the subsets, and are not present in all. For example, one factor is that the original test set contains 17 new attacks. Therefore, this is a more challenging problem than using only the training set, or other subsets that do not contain new attacks in the test data.

New attacks: The new attacks introduced in the test set cause a more challenging problem than the other subsets adopted in the literature, which is reflected in the findings. This is particularly prominent for the *Probing* class, in which one new attack (*mscan*) in the test set caused the majority of misclassifications for this class. However, it was found here that the new attacks are not the sole reason for a decreased performance on the test set for *R2L* and *U2R*, as discussed further below.

R2L class: There are two particular factors that affect the detection of this class, dependent on the subsets that are used. In the original test set, the number of *R2L* attacks double and the number of *R2L* instances is approximately sixteen times greater. However, 90.59% of all *R2L* instances in the training set (full or 10% version) belong to the *warezclient* attack, which is not present in the test set. This leads to poor *R2L* detection when the original training and test sets are used. If only the training set is adopted, *R2L* detection is very high. The second factor that affects *R2L* detection is that a large proportion of *snmpgetattack* instances are identical to *Normal* instances, which has been identified previously in the literature (Bouzida and Cuppens 2006a;b). Therefore, misclassifications between these two are inevitable, unless only the training set is adopted (or small subsets that do not include this attack).

Class imbalance: This is a factor that particularly affects *U2R* detection. There are so few instances to learn from compared with the other classes that the classifiers are prone to simply 'ignore' it as the impact on the accuracy is insignificant. Part two of this thesis has demonstrated empirically that this is the reason why MLPs trained with backpropagation have been unable to detect this class of intrusion. Therefore, data processing that affects the class balance will directly affect the results. Consequently, researchers that adopt small, balanced, subsets are able to obtain significantly better results. However, this improvement is not a valid representation of the performance of the classifier or proposed IDS prototype, if the effect of the class balance is not considered in the analysis of the results.

Duplicates: Some studies have removed duplicate instances from the data, which affects the results in two ways. First, the class balance changes, which is likely to improve the detection of the minor class(es). Second, removing duplicates from the test data removes most of the *snmpgetattack* instances identical to *Normal* instances. Therefore, misclassifications between *Normal* and *R2L* are reduced.

Validation method: It was found that holdout validation produced a more 'pessimistic' representation of the performance compared with cross validation, particularly for the minor classes. Cross validation makes more rigorous use of the data for testing, which is less sensitive to the chosen partition of a training and test set. Therefore, studies that perform cross validation on the training set, merged data set, or other subsets that are created, are likely to suggest better performance.

Taxonomy: Typically, the individual attacks are grouped into four classes of intrusion, according to the taxonomy of Kendall (1999). As identified by McHugh (2000), this may not be appropriate since the individual attacks within a class may not be similar in terms of their feature values. The empirical findings in this thesis, both from the first and third parts, confirm that this is an issue. For example, one of the new *Probing* attacks in the test set, *mscan*, targets a service that no other *Probing* attacks target. It is mainly misclassified as either *Normal* or a *DoS* attack that also targets the same service. Furthermore, in the third part of this thesis, the MLPs and classifier ensembles evolved with MABLE appeared to generalise poorly on the *U2R* class (whilst generalising well on *Normal*). However, this was found to be due to specific instances being consistently misclassified since they were different to the other instances of that class.

Aim 1.2: Provide a benchmark to assist in interpreting the findings reported in the literature

The empirical work that has been conducted for this part of the thesis has provided benchmark results that have enabled a better interpretation of the current body of research. First, it has demonstrated the impact of the methodological factors discussed above, which has helped identify why there are such discrepancies in the literature. Second, the findings can help interpret the findings of studies adopting different subsets of the KDD Cup '99 data set, such as those adopting the very small subset, as discussed in Section 4.6 on page 73.

Aim 1.3: Determine whether the KDD Cup '99 data set is useful to current and future research

Due to the existing criticisms in the literature and the discrepancies observed in results in the literature, it is important to consider whether there is any value in continuing to use the KDD Cup '99 data set in research on intrusion detection and/or machine learning. One conclusion made here is that the data set should not be continued to be used simply to evaluate IDS prototypes. However, there is still value in adopting the data set for research on machine learning applied to this domain. The data set poses several challenges to machine learning algorithms, and both generic and domain specific challenges for intrusion detection. For example:

- Related to the learning process:

  - Dealing with high dimensional data (curse of dimensionality (Bellman 1961, Duda 2001) and memory requirements).
  - Learning from a very large data set (learning speed).
  - Learning from imbalanced data.

- Feature selection (data reduction) .
- Incremental/continuous learning.
- Detecting new intrusions.
- Simulating intrusion detection of encrypted data.

What is important, however, is that researchers do not simply adopt subsets of the data set that gives the best results. The purpose of the investigation should ultimately determine which subset of the data to use.

7.2.2 Part two – learning from imbalanced data

Aim 2.1: Determine whether class imbalance is the cause of poor detection rates of MLPs reported in the literature

There has been a specific claim in the literature that MLPs are unable to detect the *U2R* class due to a lack of instances to learn from (Bouzida and Cuppens 2006a;b). An empirical investigation conducted in this thesis has demonstrated that this is due to the extreme class imbalance in the data set, in which *U2R* is the smallest class. However, it was also found that MLPs are able to detect the *U2R* class given appropriate training. The commonly adopted backpropagation algorithm is biased towards the major class(es), which causes the MLPs to ignore the *U2R* class since errors on this class are insignificant to global measures of accuracy. This coincides with other observations in the literature on class imbalance (Jo and Japkowicz 2004, Kotsiantis *et al.* 2006, Weiss and Provost 2003, Weiss 2004).

Aim 2.2: Develop an alternative method of training MLPs for imbalanced data

GAs have been used to evolve the weights of the MLPs, which has provided a successful approach to learning from imbalanced data due to the evaluation functions that have been developed. Contrary to the backpropagation algorithm, evaluation functions have been adopted here that are not biased towards the major class(es). When the MLPs are evaluated with equal weight to each class, the minor class(es) are learned significantly better, and, as mentioned above, the MLPs are able to detect the *U2R* class.

Another aim of the proposed method was to provide a generic method that assumes no domain knowledge, and, thus, can be applied to any classification task. This is one of the benefits of using GAs, which offer great flexibility in the way the MLPs can be evaluated. Generic measures based on the classification rates and error have been investigated here, which are indeed applicable to any classification task. However, domain knowledge can be incorporated into the evaluation functions if this is available, which may further improve the results.

7.2.3 Part three – MABLE

Aim 3.1: Develop a method that learns from imbalanced data and is capable of producing a set of classifiers that exhibit different classification trade-offs

The ENN proposed in the second part of the thesis, successfully learns from imbalanced data. However, the drawback of the approach is that there is no control of the classification trade-offs the evolved MLPs obtain.

Although two MLPs may obtain the same fitness score based on their classification rates, they may exhibit completely different classification trade-offs. In some cases, the classification trade-off is unacceptable; *e.g.*, too high FPR, despite a high TPR, or not detecting an intrusion class (typically *U2R*), despite obtaining a very low FPR.

An extension of the ENN has been developed, which evolves the MLPs in a multi-objective manner, considering the classification rate on each class as a separate objective. Therefore, the MOGA adopted for this purpose is able to evolve a *Pareto front* of MLPs that exhibit different classification trade-offs. This approach otherwise offers the same properties as the ENN; providing a generic way to evolve MLPs that successfully learns from imbalanced data.

Aim 3.2: Extend the approach developed for aim 3.1 to create classifier ensembles

The approach developed for aim 3.1 has been further extended to evolve classifier ensembles based on the previously evolved MLPs. The purpose of this is to obtain a new Pareto front of classifier ensembles, which should improve on the Pareto front of the MLPs. This method is referred to as MABLE (**M**ulti-Objective Evolution of **A**rtificial Neural Network Ensem**ble**s), which, importantly, also does not compromise the properties of the previous methods; *i.e.*, learns from imbalanced data and provides a set of solutions with different classification trade-offs. MABLE provides a method that is competitive with popular, state-of-the-art, methods.

Aim 3.3: Investigate how to select base classifiers to obtain a desired trade-off for an ensemble

Analysis of the Pareto fronts of solutions has yielded some significant insights into how the selection process affects the performance of the resultant ensemble(s). First, it was found that ensembles based on uniformly selected subsets of classifiers according to clustering produce ensembles with approximately the same classification trade-off as combining all. Therefore, it can be concluded that clustering is a good way to reduce the size of the ensemble without too much loss of accuracy, as recommended by Yao and Islam (2008). However, there are two drawbacks of this: (1) such an approach may not give the desired classification trade-off, and (2) a uniform selection limits the potential performance of the ensembles.

It is possible to bias the performance of the ensembles by filtering out classifiers that do not obtain a particular threshold of performance. For example, to obtain an ensemble that performs better on one class, classifiers that perform poorly on this class may be excluded from the selection process. Although it is possible to affect the classification trade-off of the ensembles in such a manner, it is an *ad hoc* process and it is difficult to obtain the exact trade-off that is desired. Furthermore, such a process remains limited by performing a uniform selection.

The selection process that takes place in MABLE is non-uniform, and the ensembles created by MABLE consistently *dominate* the ensembles based on all classifiers, clustering and thresholds. The analysis of the selected classifiers revealed that the selection is indeed non-uniform and neither local, nor entirely diverse. It was observed that the ensemble would fall approximately at the centre of the Pareto front of the selected base classifiers, but that classifiers with a seemingly poor classification trade-off were important members of the ensemble. Therefore, selection is a non-trivial task, which is well solved by an optimisation algorithm such as the MOGA adopted in MABLE. Critically, diversity alone would be a poor basis for selection.

Aim 3.4: Analyse the properties of MABLE

Several properties of MABLE have been examined, both for phase one and two. Some aspects of this analysis serve to demonstrate that the method successfully achieves the aims of this investigation, whilst other aspects help to give guidelines for practical applications.

Evolution and trade-offs: First, it has been established that MABLE successfully evolves both MLPs (base classifiers) and ensembles thereof for imbalanced data. This has been successful for the intrusion detection application considered here. Furthermore, the approach has also demonstrated a weakness in methods that generate only a single solution. Similar to the ENN developed for the second part of this thesis, the classification trade-off is a factor that is important to consider, and not only for imbalanced data. The approach is useful for any problem that has conflicting objectives, thus, leading to different classification trade-offs, which is likely for any non-trivial classification problem (*i.e.*, one that has poor separation of classes).

Ensemble size: The most obvious impact of the ensemble size is on the computational cost; the larger the ensemble, the greater the computational cost. It has also been found that the ensemble size affects both the solution quality and generalisation ability of the ensembles. On the one hand, small ensembles may not be able to obtain as good a solution for a particular classification trade-off as a larger ensemble. That is, if both a large and small ensemble obtain a 1% FPR, the larger ensemble may obtain a higher TPR than the smaller one. On the other hand, smaller ensembles are able to give a larger Pareto front. Therefore, the smaller ensembles may obtain solutions with classification trade-offs that the larger ensembles cannot obtain. However, it was found that increasing the ensemble size improves the generalisation ability and robustness when small base classifiers are used. Robustness here refers to the difference between classification rates on the training and test sets.

Size configuration: MABLE can be configured to evolve fixed sized or variable sized ensembles. As indicated above, there are benefits of evolving variable sized ensembles. However, it is a more challenging problem for the MOGA to solve. Therefore, it was observed that evolving fixed sized ensembles could produce a better Pareto front, which was more robust from trial to trial. Evolving variable sized ensembles appears to increase the chances of the MOGA 'getting stuck' in local optima. However, this may not be an issue if the MOGA is configured differently and allowed a longer time to evolve the ensembles. This is an aspect that would be useful to investigate in future work.

Generalisation: Both the MLPs and ensembles were found to generalise well, which is most clearly demonstrated on the *Normal* class. However, on average, the performance on *U2R* was found to be approximately 10% worse during testing for the MLPs, and approximately 15% worse for the ensembles. This was investigated as a potential issue of overfitting. However, despite altering the training time and number of neurons in the hidden layer of the MLPs, the classification of *U2R* instances remained significantly worse on the test set. A different hypothesis was therefore investigated; examining whether this might be caused by not learning the smallest *U2R* attacks, as there are three attacks with less than 10 instances to learn from. Again, this was not the cause. Instead, it was found that some specific instances in the test set were consistently misclassified, which caused the consistent reduction in classification rates on the test set. This reduction is greater for the ensembles since they were found to perform better on the training set, whilst misclassifying the same instances.

Another aspect of generalisation was investigated, related to the ensemble size and complexity of the MLPs (number of neurons in the hidden layer). The smaller MLPs were found to generalise better, but were not as robust in their performance. That is, whilst the larger MLPs had a more consistent performance reduction on the test set, there were a small proportion of smaller MLPs that obtained a large performance reduction; up to approximately 20% on the *Normal* class. This is a property that was found to be inherited by the ensembles. However, increasing the ensemble size does eradicate this classification behaviour. Furthermore, the larger ensembles (size 20) of the smallest MLPs (10HN) were found to generalise best.

Scalability: MABLE successfully evolves MLPs and ensembles for both two and three class data. There is one potential issue with applying the current implementation of MABLE to higher dimensional data (more classes), related to the MOGA. The number of solutions in the archive of non-dominated solutions is approximately ten times as large with three classes compared with two. This will continue to grow with increasing dimensions in order to be able to obtain the same coverage of the Pareto front. However, mechanisms can be put into place to maintain the archive to a more reasonable size, which would be useful to investigate in further work.

It is not necessary for the archive to participate in either selection or replacement. If the archive is passive, the growth induced by increasing objectives may not be an issue. More critically, however, it may be necessary to increase significantly the population size to maintain a sufficient sample of the Pareto front. This may be problematic in terms of computation, even if parallel hardware is available. As the population size increases, the work of the master thread becomes significant. At some point, it may be come too much for effective parallelisation given a cluster size, and it is not straightforward to distribute the work of the master thread.

Aim 3.5: Comparison with other classifier combination approaches

The performance of the MLPs and ensembles evolved with MABLE have been compared with DTs and two popular classifier combination approaches, namely Random Forest (RF) and AdaBoost. The DT is the base classifier used in RF and AdaBoost (which do perform better). There are indications in the literature that RF and AdaBoost learn better from imbalanced data. However, the experiments here have demonstrated that both algorithms remain biased towards the major class (*Normal*), which leads to a very low FPR. This is somewhat wasteful in this application since we can allow a higher amount of false positives.

Hu *et al.* (2003) and Maxion and Townsend (2002) consider 1% FPR acceptable, which is adopted here as a constraint to demonstrate the drawback of approaches that produce only a single solution and do not consider the classification trade-off problem. By simply allowing a 1% FPR, there are MLPs that offer better solutions compared with RF and AdaBoost. The ensembles evolved with MABLE provide better solutions still, whilst also achieving a higher accuracy.

7.3 Discussion

This thesis has made contributions to both the intrusion detection and machine learning domains. The specific contributions, and novelty of this work, are discussed specifically in the following section. This section discusses how this work relates to current research and practical applications.

The methods proposed here (ENN and MABLE) are not sufficient as stand alone systems for intrusion detection. There are several aspects of an IDS that have not been considered here, including *inter alia* system architecture, data processing, alert aggregation, and reporting mechanisms. However, the methods proposed here have addressed a critical challenge of learning from imbalanced data and providing the user with a set of solutions from which s/he can choose one (or more) that exhibit a sought after classification trade-off. This solution, be it a single classifier or an ensemble, can then be incorporated in an IDS framework as a detection module. Furthermore, there is room for improvement of the proposed methods, concerning scalability and performance, which are discussed further in Section 7.5.

Classifier combination is a successful method for improving on the performance of a single classifier. Furthermore, classifier ensembles can provide additional security from an adversary (Biggio *et al.* 2008; 2009). Although the findings in this thesis support the observations in the literature, that classifier combination can improve on the performance of single classifiers, current approaches to creating ensembles are prone to fail to do so simply because they may yield a solution with a poor classification trade-off. It has been demonstrated in this thesis that simply allowing a 1% FPR would enable single classifiers to outperform both RF and AdaBoost. In general, this would be true of all such constraints except those within some small region about the single ensemble produced by these techniques. This is of concern to the wider machine learning domain, as such a trade-off problem is likely to exist in any non-trivial classification problem (in fact, irrespective of any class imbalance). For example, during its development, MABLE was applied to drug screening (Engen *et al.* 2009), which is also an application that is challenged by extreme class imbalance. Similarly to the intrusion detection problem, the primary goal is to obtain as high a TPR as possible (correct classification of the minor class). However, for this problem, a significantly higher FPR can be allowed, which demonstrates an even greater difference between feasible solutions evolved with MABLE and 'single' ensembles that offer a poor classification trade-off.

There are other approaches to learning from imbalanced data, namely data sampling and cost sensitive learning, as reviewed in Section 5.1 on page 77. Sampling training data is considered an infeasible approach here due to the extreme class imbalance. For example, undersampling would arguably remove too much data of the majority classes to create an equilibrium with the minor classes. Furthermore, this is not a convenient approach for addressing the trade-off problem. In this respect, cost sensitive learning is more appropriate. Weight matrices can be designed and modified to both better learn from imbalanced data and to directly bias the classification trade-off. The drawback of such an approach is similar to the *ad hoc* procedure of the threshold based filtering process that has been examined in this thesis. Such an approach is also similar to weight based approaches to dealing with multiple objectives in optimisation problems, in which multiple runs of the same algorithm with different weights are executed to obtain different solutions. Research on multi-objective optimisation has established that this is not an ideal approach, since it may not provide good coverage of the Pareto front (Deb 2001) and it would be far more computationally expensive to do so. In this respect, multi-objective optimisation is considered a better alternative, with the additional benefit that an approximation of the true Pareto front of solutions can be found in a single run of the algorithm. Nevertheless, it is interesting for further work to examine how well MABLE compares with cost sensitive learning.

There is another trade-off that should be considered when developing classifier ensembles, which is related to computational costs and performance. These factors are affected by the ensemble size and complexity of the base classifiers. For example, small ensembles of small MLPs are not as robust as small ensembles of larger, more complex, MLPs. Large ensembles of small MLPs generalise better and resolve the robustness problem. Although the smaller MLPs have less computational costs, the required (larger) ensemble size may render this unfruitful, dependent on constraints of the particular application. However, due to the combination method adopted here, the majority vote combiner, a parallel implementation may render this problem obsolete if parallel hardware is available. It is also important, in this context, to bear in mind that the larger the ensemble, the smaller the Pareto front of trade-off solutions becomes. Therefore, a particular trade-off may not be achieved if the ensemble size is too large. This is, of course, dependent on the sample density across the Pareto front.

The enhanced generalisation ability of larger ensembles of smaller MLPs may be considered directly in the evolutionary process of MABLE. If an estimation set is adopted, the performance difference between this and the training set may be incorporated into the fitness evaluation. This is mentioned here to emphasise the further potential benefits of MABLE, or, specifically, multi-objective optimisation, when utilised to create classifiers or classifier ensembles. This is discussed further in Section 7.5.

7.4 Contributions and novelty

This thesis has made several contributions to two key research areas: intrusion detection and machine learning. Some of the contributions apply specifically to the application of machine learning to intrusion detection. However, many of the findings here contribute to the wider machine learning community, on general classification problems, classifier combination and class imbalance.

Of particular interest to the intrusion detection domain, one of the main contributions of the thesis is the discovery and empirical investigation of discrepancies in the findings reported in the literature with the KDD Cup '99 data set. From this investigation, class imbalance has been identified as a significant challenge to intrusion detection, which has been shown, empirically, to cause poor detection of certain classes of intrusion (the minor classes). Learning from imbalanced data is a problem that is of interest to the wider machine learning community, and, therefore, this research is of interest to any problem with class imbalance. Furthermore, an issue not previously considered in the machine learning community has been identified, that most non-trivial classification problems will have conflicting objectives, and, therefore, there can be a range of possible solutions that exhibit different trade-offs among these objectives/classes. Consequently, the current approaches that focus on creating a single best solution based on accuracy may provide a classification trade-off that renders the solution useless. This problem has been demonstrated empirically and a new method has been proposed that takes this into account when creating both classifiers and classifier ensembles.

The specific contributions of the work in this thesis are listed below, organised according to the three empirical parts of this thesis. The contributions related to the first part of this thesis are as follows:

- Discovered wide spread discrepancies in the findings reported in the literature, despite the studies stating that they use the same data set (KDD Cup '99).
- Identified several methodological factors that have caused the discrepancy, and demonstrated empirically how they affect the results of common, well known, classifiers.
  - The comparative investigation has allowed a better interpretation of the current body of research.
  - The findings can contribute to a more accurate analysis of results in future studies adopting the KDD Cup '99 data set.
- The empirical investigation, and the findings from that, have enabled the proper consideration of the use of the data set in future work. This is an important contribution for two reasons:
  1. There are several criticisms of the data set that encourage researchers to stop using it. However, no other studies have attempted to identify how the data set can still provide valuable contributions to the research community.
  2. Due to a lack of better publicly available data sets, researchers do continue to use the data set.

Contributions related to the second part of the thesis:

- Identified and demonstrated that the extreme class imbalance in the KDD Cup '99 data set is a reason that the *U2R* class has been reported to be undetectable by MLPs in the literature.
- Demonstrated that MLPs are capable of learning from imbalanced data, given a training approach that is not biased towards the major class(es).
- Investigated the impact of various formulations of objective function in genetic algorithms used to evolve MLPs.

Contributions related to the final empirical part of the thesis:

- Identification of the classification trade-off problem.
    - The investigations here, on imbalanced data, have demonstrated clearly this problem.
    - A solution may dominate a small number of other (local) classifiers, but is also dominated by a number of other classifiers. If the solution does not provide an appropriate classification trade-off, another solution with a lower accuracy will be superior if it delivers a better trade-off.
- Proposed an approach to creating MLP classifiers that is capable of learning from imbalanced data and considers the classification trade-off problem.
    - Gives a user the choice of a wide range of solutions, from which s/he can select one (or more) with the ideal classification trade-off(s).
    - Can be applied to any classification problem for which there are conflicting objectives.
- Proposed an approach to creating classifier ensembles (MABLE), which takes into account the challenges of learning from imbalanced data and allows the user to select the ensemble with the ideal classification trade-off.
    - Gives a new set of solutions that exhibit different classification trade-offs, which yields an improved Pareto front of solutions compared with the base classifiers (MLPs).
- Demonstrated empirically that common classifier combination approaches such as RF and AdaBoost, although indicated in the literature to better learn from imbalanced data, remain biased towards the major class, therefore obtaining poor classification trade-offs, which were worse than the MLPs and ensembles evolved with MABLE.
- Provided a new perspective on the selection of base classifiers, using the concepts of dominance and Pareto optimality, which has assisted in explaining why an ensemble obtains a particular classification trade-off.
- The proposed method of classifier combination has achieved pragmatic solutions to three common challenges of evolving classifier ensembles (Abbass 2006, Yao and Islam 2008), which can be adopted in other studies:
    1. Provided an implicit method of creating diverse base classifiers, by evolving MLPs with a MOGA.
    2. Optimising the selection of classifiers with a MOGA, which produces a Pareto front of solutions that exhibit different classification trade-offs.
    3. Determining the size of the ensemble automatically, allowing the MOGA to evolve an optimal Pareto front of solutions, or specifying a fixed size, which may be desirable for applications with known computational constraints. This was found to be superior to determining the size of the ensemble(s) by clustering.
- Demonstrated how the ensemble size and complexity of the base classifiers affect the generalisation, size of the Pareto front of solutions, and the computational costs. This is, therefore, another trade-off problem to consider.

Much of the work of this thesis contains elements of novelty. Specific areas of novelty include:

- Novel use of fitness functions in a GA to evolve MLPs for imbalanced data.
- MOGA application in MABLE.
  - MOGAs have only very recently been adopted for evolving classifiers and classifier ensembles, *e.g.*, (Abbass 2003a, Chandra and Yao 2004; 2005; 2006a;b, dos Santos *et al.* 2008), and none of the existing studies consider the trade-off problem for which the MOGA has been applied to solve here.
  - Novel multi-objective fitness function, used to evolve both classifiers and classifier ensembles.
- MABLE provides a new, implicit, method of creating diverse base classifiers.
- Novel application of the concepts of dominance and Pareto optimality to analyse the selection of base classifiers for the ensembles, and to demonstrate the trade-off problem.

7.5 Further work

The further work that is discussed here mainly focuses on MABLE. Since MABLE is already an extension of the ENN, the latter is therefore not considered. The aims and objectives regarding the investigation into discrepancies in the findings with the KDD Cup '99 data set in the literature were conclusive. Recommendations for future work with this data set have been indicated, and as previously stated in this thesis, this should rather be focused on creating and maintaining a new data set for intrusion detection. Therefore, this section focuses on the further work related to MABLE.

First, in Section 7.5.1 further validation of the results is discussed. Thereafter, potential improvements of MABLE are discussed. The scalability of the algorithm is discussed in Section 7.5.2. In conclusion, Section 7.5.3 discusses further work related to the performance and pragmatic extensions.

7.5.1 Validation

The findings reported in this thesis are of interest to the wider machine learning community. Therefore, it is desirable to conduct more experiments on different data sets. This has already been examined for drug screening (Engen *et al.* 2009), which focused on learning from imbalanced data. However, it is interesting to include more data sets with different properties, not only imbalanced data.

According to the findings here, MABLE should be successful on any classification problem with conflicting objectives (classes). However, there may be classification problems that do not have a prominent conflict between the objectives. Even though there may be a class imbalance, this does not imply that the objectives are conflicting. For such cases, MABLE may not be as successful as other, existing, techniques.

7.5.2 Scalability

Perhaps most importantly, the scalability of MABLE needs to be addressed in future work. In particular, this concerns the extensive growth of the archive in the current implementation. A simple solution, which may suffice, is to perform clustering of the archive at regular intervals to maintain an appropriate size, as in the SPEA2 algorithm (Zitzler *et al.* 2002). However, the archive may be implemented as a passive mechanism, in which case this increase may not be an issue. More critically, the population size is likely to significantly increase to be able to maintain a reasonable coverage of the Pareto front.

Dependent on the application, domain knowledge may be used to determine a different, smaller, set of objectives. For example, using TPR and FPR as objectives for intrusion detection may suffice. The drawback of using TPR and FPR as objectives is the lack of control of the trade-offs between the individual classes, which may be important if this approach is adopted to learn from imbalanced data. That is, since the training would be insensitive to this, a class could be 'ignored' whilst achieving a reasonable TPR/FPR. Furthermore, this may reduce the diversity in the pool of base classifiers for ensemble generation. A potential solution may be to evolve base classifiers according to the individual classes, and then evolve the ensembles according to TPR and FPR.

A different mechanism that could be adopted to help manage the magnitude of solutions and their properties is to implement constraint handling in the MOGA. Several methods of constraint handling in GAs has been proposed; see, for example, (Coello Coello and Christiansen 1999, Deb 2000; 2001, Jiménez *et al.* 1999), which can be utilised if suitable constraints on classification rates are known *a priori*.

7.5.3 Performance and pragmatic extensions

A different aim for future work on MABLE is to enhance the performance. This has not been a specific focus of the investigation in this thesis, and better performance may be achieved by simply configuring the MOGA and the MLPs differently, or by processing the data differently. With respect to the latter, only one approach to handling nominal parameters was examined, by enumeration and scaling to single features. Alternatively, treating all nominal values as separate features may prove beneficial. Bouzida and Cuppens (2006a;b) treat the nominal values in the KDD Cup '99 data set as separate features and, thus, obtain 125 input features. Due to the increase in input features, it may be desirable to reduce the dimensionality of the problem by combining some nominal values into a single feature. For this reason, Tsang *et al.* (2007) exclude the service feature, which gives 52 remaining features. However, instead of excluding this feature, the main services could be adopted as separate features, and the rest labelled 'other'.

Different preprocessing may be beneficial, compared with scaling or normalising numerical features according to the minimum and maximum values in the data set. For example, for the features that represent source and destination bytes, greater values may be encountered when the IDS is deployed, compared with the values present in the training data. Furthermore, Depren *et al.* (2005) found it necessary to employ a clustering algorithm to perform the analysis of those values since it was otherwise not possible to properly distinguish instances based on these features if the values had been scaled or normalised in the range [0,1]. They give an example of a DoS attack to demonstrate why this is an issue: *"destination byte values have 0 bytes and source byte values have 40–50 bytes. However, in normal connections both features have 40–50 bytes."* Considering a maximum value of 5000000, 50 bytes obtains a normalised/scaled value of 10^{-5}, which was not possible to distinguish from 0 by a self organising map (Depren *et al.* 2005). Alternative processing that may be explored includes discretisation, which may incorporate domain knowledge to set appropriate ranges, or performing exponential scaling.

Different evaluation functions could also be examined. For example, as mentioned above, the TPR and FPR may be considered as objectives, which may help with potential scalability issues. Examples of other potential objectives include the MSE (Mean of Squared Errors) for each class, ensemble size, and consideration of generalisation ability. However, for the latter, this needs to be calculated based on performance on an estimation set. For problems that have very few instances to learn from, this may not be feasible. The MOGA may also be extended to evolve the structure of the MLPs as well as their weights, as in (Han and Cho 2006), which has been found to produce better classifiers.

Better performing ensembles may be evolved with different combiners, such as a weighted majority vote combiner or a cascade combination. Ensembles of one class classifiers are also interesting to investigate, in two respects: (1) to compare the performance of the current implementation of MABLE with existing one class ensemble approaches, and (2) to allow MABLE to evolve ensembles of one class classifiers. This may be an alternative that resolves the potential scalability issues of MABLE on problems with many classes. Furthermore, evolving *hierarchical* combinations of one class classifiers may provide more successful solutions for intrusion detection. An recent example of this is the system proposed by Xiang *et al.* (2008), as discussed in Section 4.6.1 on page 73, which appears to be very successful, although the FPR is arguably too large. Utilising the concepts proposed here, the classification trade-off can be controlled to obtain solutions with a more acceptable false positive rate. There are other examples of multistage classifier ensembles in the literature, *e.g.*, (Senator 2005, Yang *et al.* 2002), but evolving such combinations has not been previously considered.

References

M.S. Abadeh, J. Habibi and S. Aliari. Using a Particle Swarm Optimization Approach for Evolutionary Fuzzy Rule Learning: A Case Study of Intrusion Detection. In *Information Processing and Management of Uncertainty in Knowledge Based Systems (IPMU)*, Paris, France, July 2–7 2006.

M.S. Abadeh, J. Habibi and E. Soroush. Induction of Fuzzy Classification Systems Using Evolutionary ACO-Based Algorithms. In *AMS '07: Proceedings of the First Asia International Conference on Modelling & Simulation*, pages 346–351, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-2845-7.

H. Abbass. *Multi-Objective Machine Learning*, chapter Pareto-Optimal Approaches to Neuro-Ensemble Learning, pages 405–427. Studies in Computational Intelligence. Springer, January 2006.

H.A. Abbass. Pareto neuro-evolution: constructing ensemble of neural networks using multi-objective optimization. In *Proc. of the 2003 Conference on Evolutionary Computation*, pages 2074–2080. IEEE Press, 2003a.

H.A. Abbass. 2003b. Speeding up backpropagation using multiobjective evolutionary algorithms. *Neural Computation*, **15**, 2705–2726. ISSN 0899-7667. (doi:http://dx.doi.org/10.1162/089976603322385126)

A. Abraham, C. Grosan and C. Martin-Vide. 2007. Evolutionary Design of Intrusion Detection Programs. *International Journal of Network Security*, **4**, 328–339.

D. Abramson and J. Abela. Parallelisation of a genetic algorithm for the computation of efficienttrain schedules. In *Proceedings of the1993 Parallel Computing and Transputers Conference*, pages 139–149, 1993.

K. Ahmadian. *Re: Research question*. e-mail to Engen, V., (vengen@bournemouth.ac.uk), (ku_ahmadian@comp.iust.ac.ir), 24 Oct 2008.

K. Ahmadian, A. Golestani, M. Analoui and M.R. Jahed. Evolving Ensemble of Classifiers In Low-Dimensional Spaces Using Multi-Objective Evolutionary Approach. In *Proc. 6th IEEE/ACIS Int. Conference on Computer and Information Science*, pages 217–222, 11-13 July 2007a.

K. Ahmadian, A. Golestani, N. Mozayani and P. Kabiri. A New Multi-Objective Evolutionary Approach for Creating Ensemble of Classifiers. In *Proc. IEEE Int. Conf. on Systems, Man and Cybernetics*, pages 1031–1036, 7-10 Oct. 2007b.

E. Ahmed, K. Samad and W. Mahmood. Cluster-based Intrusion Detection (CBID) Architecture for Mobile Ad Hoc Networks. In *Proceedings of the AusCERT Asia Pacific Information Technology Security Conference*, pages 46–56, 2006.

U. Aickelin, P. Bentley, S. Cayzer, J. Kim and J. McLeod. Danger Theory: The Link Between AIS and IDS. In *Proceedings of the 2nd International Conference on Artificial Immune Systems*, pages 147–155, 2003.

U. Aickelin and S. Cayzer. The Danger Theory and Its Application to AIS. In *Proceedings of the 1st International Conference on Artificial Immune Systems*, pages 141–148, 2002.

S.O. Al-Mamory and H. Zhang. 2009. Intrusion detection alarms reduction using root cause analysis and clustering. *Comput. Commun.*, **32**, 419–430. ISSN 0140-3664.

A. Al-Shahib, R. Breitling and D. Gilbert. 2005. Feature Selection and the Class Imbalance Problem in Predicting Protein Function from Sequence. *Applied Bioinformatics*, **4**, 195–203.

M. Al-Subaie and M. Zulkernine. Efficacy of Hidden Markov Models Over Neural Networks in Anomaly Intrusion Detection. In *30th Annual International Computer Software and Applications Conference (COMPSAC'06)*, pages 325–332, 2006.

E. Alba and J.M. Troya. 1999. A survey of parallel distributed genetic algorithms. *Complexity*, **4**, 31–52. ISSN 1076-2787.

M. Albaghdadi, B. Briley, M.W. Evens, R.S.M. Petiwala and M. Hamlen. A Framework for Event Correlation in Communication Systems. In *MMNS '01: Proceedings of the 4th IFIP/IEEE International Conference on Management of Multimedia Networks and Services*, pages 271–284, London, UK, 2001. Springer-Verlag. ISBN 3-540-42786-4.

J. Albus. 1975. A new approach to manipulator control: The cerebellar model articulation controller (cmac). *Transactions ASME, Journal Dynamic Systems, Measurement, and Control*, **97**, 220–227.

S.H. Amer. *Enhancing host based intrusion detection systems with danger theory of artificial immune systems*. PhD thesis, Auburn University, Auburn, AL, USA, 2008.

G. An. 1996. The effects of adding noise during backpropagation training on a generalization performance. *Neural Computation*, **8**, 643–674. ISSN 0899-7667.

X. An, D. Jutla and N. Cercone. A Bayesian Network Approach to Detecting Privacy Intrusion. In *WI-IATW '06: Proceedings of the 2006 IEEE/WIC/ACM international conference on Web Intelligence and Intelligent Agent Technology*, pages 73–76, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2749-3.

K.G. Anagnostakis, E.P. Markatos, S. Antonatos and M. Polychronakis. $E^2 \times B$: A Domain Specific String Matching Algorithm for Intrusion Detection. In *Proceedings of the 18th IFIP International Information Security Conference (SEC2003)*, Athens, Greece, 2003.

J.P. Anderson. Computer Security Threat Monitoring and Surveillance. Technical Report 79F296400, James P. Anderson Co., Fort Washington, Pennsylvania, 1980.

S. Ando. Heuristic speciation for evolving neural network ensemble. In *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1766–1773, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-697-4.

S. Ando and E. Suzuki. An Information Theoretic Approach to Detection of Minority Subsets in Database. In *ICDM '06: Proceedings of the Sixth International Conference on Data Mining*, pages 11–20, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2701-9.

G.R. Andrews. *Foundations of Multithreaded, Parallel, and Distributed Programming*. Addison-Wesley, 2000. ISBN 0-201-35753-6.

P.J. Angeline. *Evolutionary Computation 1: Basic Algorithms and Operators*, chapter Parse trees, pages 155–159. Institute of Physics Publishing, 2000.

Anon. Weka 3: Data Mining Software in Java, 2006. Available online: http://www.cs.waikato.ac.nz/ml/weka/ [Accessed March 5th 2010].

Anon. Central loghost mini how to, 2010. Available online: http://www.campin.net/newlogcheck.html#newlogcheck/ [Accessed March 5th 2010].

S. Antonatos, M. Polychronakis, P. Akritidis, D. Kostas, K. G. Anagnostakis and E. P. Markatos. Piranha: Fast and Memory Efficient Pattern Matching for Intrusion Detection. In *Proceedings of the 20th International Information Security Conference (IFIP/SEC 2005)*, May 2005.

N.B. Anuar, H. Sallehudin, A. Gani and O. Zakari. 2008. Identifying false alarm for network intrusion detection system using hybrid data mining and decision tree. *Malaysian Journal of Computer Science*, **21**, 101–115.

M. Asaka, A. Taguchi and S. Goto. The Implementation of IDA: An Intrusion Detection Agent System. In *Proceedings of the 11th Annual FIRST Conference on Computer Security Incident Handling and Response (FIRST'99)*, 1999.

M.A. Aydin, A.H. Zaim and K.G. Ceylan. May 2009. A hybrid intrusion detection system design for computer network security. *Computers and Electrical Engineering*, **35**, 517–526. ISSN 0045-7906.

H. Bal. *Programming Distributed Systems*. Silicon Press and Prentice Hall, 1990. ISBN 0137220839. ISBN 0-13-722083-9.

B. Balajinath and S.V. Raghavan. July 2001. Intrusion Detection Through Learning Behaviour Model. *International Journal of Computer Communications*, **24**, 1202–1212.

J. Balthrop, F. Esponda, S. Forrest and M. Glickman. Coverage and Generalization in an Artificial Immune System. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2002)*, pages 3–10, New York, 2002. Morgan Kaufmann.

S. Banerjee, C. Crina Grosan and A. Abraham. IDEAS: Intrusion Detection Based on Emotional Ants for Sensors. In *5th International Conference on Intelligent Systems, Design and Applications (ISDA-05)*, Wroclaw, Poland, 2005.

Z. Banković, S. Bojanić, O. Nieto and A. Badii. Unsupervised Genetic Algorithm Deployed for Intrusion Detection. In *HAIS '08: Proceedings of the 3rd international workshop on Hybrid Artificial Intelligence Systems*, pages 132–139, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 978-3-540-87655-7.

Z. Banković, D. Stepanović, S. Bojanić and O. Nieto-Taladriz. 2007. Improving network security using genetic algorithm approach. *Comput. Electr. Eng.*, **33**, 438–451. ISSN 0045-7906.

A. Banks, J. Vincent and C. Anyakoha. 2008a. A review of particle swarm optimization. Part I: background and development. *Natural Computing*, **6**, 467–484. ISSN 1567-7818.

A. Banks, J. Vincent and C. Anyakoha. 2008b. A review of particle swarm optimization. Part II: hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications. *Natural Computing*, **7**, 109–124. ISSN 1567-7818.

D. Barbara, N. Wu and S. Jajodia. Detecting Novel Network Intrusions Using Bayes Estimators. In *Proceedings of the SIAM International Conference on Data Mining*, Chicago, USA, April 2001.

M. Barreno, B. Nelson, R. Sears, A.D. Joseph and J.D. Tygar. Can Machine Learning Be Secure? In *ASIACCS '06: Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, 2006.

E. Bauer and R. Kohavi. 1999. An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants. *Machine Learning*, **36**, 105–139. ISSN 0885-6125.

BBC. Cyber-security strategy launched, 2009a. Available online: `http://news.bbc.co.uk/1/hi/uk_politics/8118348.stm` [Accessed June 25th 2009].

BBC. Major cyber spy network uncovered, 2009b. Available online: `http://news.bbc.co.uk/1/hi/world/americas/7970471.stm` [Accessed May 8th 2009].

BBC. Spies 'infiltrate US power grid', 2009c. Available online: `http://news.bbc.co.uk/1/hi/technology/7990997.stm` [Accessed May 8th 2009].

BBC. US launches cyber security plan , 2009d. Available online: `http://news.bbc.co.uk/1/hi/world/americas/8073654.stm` [Accessed June 8th 2009].

R.E. Bellman. *Adaptive Control Processes: A Guided Tour.* Princeton University Press, 1961.

N. Ben Amor, S. Benferhat and Z. Elouedi. Naive Bayes vs Decision Trees in Intrusion Detection Systems. In *SAC '04: Proceedings of the 2004 ACM symposium on Applied computing*, pages 420–424, New York, NY, USA, 2004. ACM. ISBN 1-58113-812-1.

S. Benferhat, F. Autrel and F. Cuppens. Enhanced Correlation in an Intrusion Detection Process. In *Lecture Notes in Computer Science*, volume 2776, pages 157–170. Springer, 2003.

S. Benferhat and K. Tabia. On the combination of naive bayes and decision trees for intrusion detection. In *CIMCA '05: Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce Vol-1 (CIMCA-IAWTIC'06)*, pages 211–216, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2504-0-01.

H-G. Beyer and H-P. Schwefel. 2002. Evolution strategies - a comprehensive introduction. *Natural Computing*, **1**, 3–52.

E. Biermann, E. Cloete and L.M. Venter. 2001. A comparison of intrusion detection systems. *Computers & Security*, **20**, 676–683.

B. Biggio, G. Fumera and F. Roli. Adversarial Pattern Classification Using Multiple Classifiers and Randomisation. In *SSPR & SPR '08: Proceedings of the 2008 Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition*, pages 500–509, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 978-3-540-89688-3.

B. Biggio, G. Fumera and F. Roli. Multiple Classifier Systems for Adversarial Classification Tasks. In *MCS '09: Proceedings of the 8th International Workshop on Multiple Classifier Systems*, pages 132–141, Berlin, Heidelberg, 2009. Springer-Verlag. ISBN 978-3-642-02325-5.

L. Booker. *Genetic Algorithms and Simulated Annealing*, chapter Improving Search in Genetic Algorithms, pages 61–73. Morgan Kaufmann, 1987.

B.E Boser, I.M. Guyon and V.N. Vapnik. A training algorithm for optimal margin classifiers. In *COLT '92: Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, New York, NY, USA, 1992. ACM. ISBN 0-89791-497-X.

A. Bosin, N. Dessì and B. Pes. Intelligent Bayesian Classifiers in Network Intrusion Detection. In *Proceedings of the 18th international conference on Innovations in Applied Artificial Intelligence*, pages 445–447, London, UK, 2005. Springer-Verlag. ISBN 3-540-26551-1.

A. Boukerche, R.B. Machado, K.R.L. Jucá, J.B.M. Sobral, M. Bosco and M.S.M.A. Notare. 2007. An agent based and biological inspired real-time intrusion detection and security model for computer network operations. *Compututer Communications*, **30**, 2649–2660. ISSN 0140-3664.

Y. Bouzida. *Principal Component Analysis for Intrusion Detection and Supervised Learning for New Attack Detection*. PhD thesis, Graduate Faculty of Ecole Nationale Supérieure des Télécommunications de Bretagne, 2006.

Y. Bouzida and F. Cuppens. Detecting Known and Novel Network Intrusions. In *IFIP/SEC 2006, 21st IFIP TC-11 International Information Security Conference*, 2006a.

Y. Bouzida and F. Cuppens. Neural networks vs. decision trees for intrusion detection. In *IEEE / IST Workshop on Monitoring, Attack Detection and Mitigation (MonAM)*, 2006b.

A.P. Bradley. 1997. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Regonition*, **30**, 1145–1159.

M.F. Bramlette. Initialization, Mutation and Selection Methods in Genetic Algorithms for Function Optimization. In *Proceedings of the 4th International Conference on Genetic Algorithms*, pages 100–107. Morgan Kaufmann, July 1991.

J. Branke, H. Schmeck, K. Deb and M. Reddy. Parallelizing Multi-Objective Evolutionary Algorithms: Cone Separation. In *2004 Congress on Evolutionary Computation (CEC'2004)*, volume 2, pages 1952–1957, Portland, Oregon, USA, June 2004. IEEE Service Center.

L. Breiman. 1996. Bagging Predictors. *Machine Learning*, **24**, 123–140. ISSN 0885-6125.

L. Breiman. 1998. Arcing classifiers. *The Annals of Statistics*, **26**, 801–849.

L. Breiman. 2001. Random Forests. *Machine Learning*, **45**, 5–32. ISSN 0885-6125.

S. M. Bridges and R. B. Vaughn. Fuzzy Data Mining and Genetic Algorithms Applied to Intrusion Detection. In *Proceedings of the National Information Systems Security Conference (NISSC)*, pages 13–31, Baltimore, MD, October 2000.

G. Brown and J. Wyatt. Negative Correlation Learning and the Ambiguity Family of Ensemble Methods. In *Multiple Classifier Systems*, 2003.

G. Brown, J. Wyatt, R. Harris and X. Yao. 2005. Diversity creation methods: A survey and categorisation. *Journal of Information Fusion*, **6**, 5–20.

T. Brugger. An assessment of the DARPA IDS Evaluation Dataset using Snort. Technical Report CSE-2007-1, University of California, Department of Computer Science, 2007a.

T. Brugger. September 2007b. Kdd cup '99 dataset (network intrusion) considered harmful. *KDnuggets*, **n18**. Available online: [`http://www.kdnuggets.com/news/2007/n18/4i.html`] [Accessed November 23 2007].

P. Brutch and C. Ko. Challenges in Intrusion Detection for Wireless Ad-hoc Networks. In *SAINT-W '03: Proceedings of the 2003 Symposium on Applications and the Internet Workshops (SAINT'03 Workshops)*, pages 368–373, Washington, DC, USA, 2003. IEEE Computer Society. ISBN 0-7695-1873-7.

J. Burez and D. van den Poel. April 2009. Handling Class Imbalance in Customer Churn Prediction. *Expert Systems with Applications*, **36**, 4626–4636.

C.J.C. Burges. 1998. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, **2**, 121–167.

M. Burgess. Computer immunology. In *LISA '98: Proceedings of the 12th USENIX conference on System administration*, pages 283–298, Berkeley, CA, USA, 1998. USENIX Association.

M. Burgess. Evaluating cfengine's immunity model of site maintenance. In *Proceedings of the 2nd SANE System Administration Conference*, 2000.

D.J. Burroughs, L.F. Wilson and G.V. Cybenko. Analysis of distributed intrusion detection systems using Bayesian methods. In *21st IEEE International Conference on Performance, Computing, and Communications*, pages 329–334, Washington, DC, USA, 2002. IEEE Computer Society. ISBN 0-7803-7371-5.

J. Cannady. Artificial Neural Networks for Misuse Detection. In *Proceedings of the National Information Systems Security Conference*. NIST, Washington, DC, 1998.

J. Cannady. Next Generation Intrusion Detection: Autonomous Reinforcement Learning of Network Attacks. In *Proceedings of the 23rd National Information Systems Secuity Conference*, 2000.

E. Cantú-Paz. A survey of parallel genetic algorithms. Technical Report 95007, Illinois Genetic Algorithms Labratory, University of Illinoi, 1998.

E. Cantú-Paz. *Efficient and Accurate Parallel Genetic Algorithms*. Kluver Academic Publishers, 2001.

E. Cantú-Paz and D.E. Goldberg. Predicting speedups of idealized bounding cases of parallel genetic algorithms. In *Proceedings of the seventh International Conference on Genetic Algorithms*, pages 113–120. Morgan Kaufmann, July 1997.

A. Chandra and X. Yao. Divace: Diverse and accurate ensemble learning algorithm. In *Proceedings of the 5th International Conference on Intelligent Data Engineering and Automated Learning*, pages 619–625, August, 2004.

A. Chandra and X. Yao. Evolutionary Framework for the Construction of Diverse Hybrid Ensembles. In *Proc. of the 13th European Symposium on Artificial Neural Networks (ESANN'2005)*, pages 253–258, Bruges, Belgium, 27-29 April 2005.

A. Chandra and X. Yao. 2006a. Ensemble Learning Using Multi-Objective Evolutionary Algorithms. *Mathematical Modelling and Algorithms*, **5**, 417–445.

A. Chandra and X. Yao. 2006b. Evolving hybrid ensembles of learning machines for better generalisation. *Neurocomputing*, **69**, 686–700.

N.V. Chawla. C4.5 and Imbalanced Data sets: Investigating the effect of sampling method, probabilistic estimate, and decision tree structure. In *ICML Workshop on Learning from Imbalanced Datasets II*, 2003.

N.V. Chawla, K.W. Bowyer, L.O. Hall and W.P. Kegelmeyer. 2002. Smote: Synthetic minority oversampling technique. *Journal of Artificial Intelligence Research*, **16**, 321–357.

N.V. Chawla, N. Japkovicz and A. Kolcz, editors. *ICML Workshop on Learning from Imbalanced Data Sets II*, 2003.

N.V. Chawla, N. Japkowicz and A. Kotcz. 2004. Editorial: special issue on learning from imbalanced data sets. *SIGKDD Explorations Newsletter*, **6**, 1–6. ISSN 1931-0145.

S. Chebrolu, A. Abraham and J. P. Thomas. June 2005. Feature Deduction and Ensemble Design of Intrusion Detection Systems. *Computers and Security*, **24**, 295–307.

P. Cheeseman and J. Stutz. Bayesian Classification (AutoClass): Theory and Results. In *Advances in Knowledge Discovery and Data Mining*, pages 153–180, 1996. ISBN 1-55860-163-5.

C. Chen, A. Liaw and L. Breiman. Using random forest to learn imbalanced data. Technical Report 666, Department of Statistics, University of California, Berkeley, July 2004.

D. Chen, J. Quirein, H. Smith Jr, S. Hamid and J. Grable. October 2005a. Neural Network Ensemble Selection using a Multi-Objective Genetic Algorithm in Processing Pulsed Neutron Data. *Petrophysics*, **46**, 323–334.

H. Chen and X. Yao. Evolutionary Multiobjective Ensemble Learning Based on Bayesian Feature Selection. In *Proc. of the 2006 IEEE Congress on Evolutionary Computation (CEC'06)*, pages 971–978, Vancouver, Canada, 16-21 July 2006. IEEE Press.

H. Chen and X. Yao. Evolutionary Random Neural Ensemble Based on Negative Correlation Learning. In *Proc. of the 2007 IEEE Congress on Evolutionary Computation (CEC'07)*, pages 1468–1474, Singapore, 25-28 September 2007.

Y. Chen, A. Abraham and J. Yang. Feature Selection and Intrusion Detection Using Hybrid Flexible Neural Tree. In *International Syumposium on Neural Networks (ISNN 03)*, pages 439–444, Chongqing, China, 2005b.

Y. Chen, Y. Li, X-Q. Cheng and L. Guo. Survey and Taxonomy of Feature Selection Algorithms in Intrusion Detection System. In *Information Security and Cryptology*, pages 153–167. Springer, 2006.

D. Chess, C. Harrison and A. Kershenbaum. Mobile Agents: Are They a Good Idea? Technical Report RC 19887 (December 21, 1994 - Declassified March 16, 1995), IBM Research Report, Yorktown Heights, New York, 1994.

W. Chimphlee, A.H. Abdullah, M. Noor Md Sap, S. Srinoy and S. Chimphlee. Anomaly-Based Intrusion Detection using Fuzzy Rough Clustering. In *ICHIT '06: Proceedings of the 2006 International Conference on Hybrid Information Technology*, pages 329–334, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2674-8.

D.A. Cieslak, N.V. Chawla and A. Striegel. Combating Imbalance in Network Intrusion Datasets. In *IEEE Int. Conf. on Granual Computing*, pages 732–737, 2006.

CNet news. Georgia accuses Russia of coordinated cyberattack, 2008. Available online: `http://news.cnet.com/8301-1009_3-10014150-83.html` [Accessed May 8th 2009].

W. Cockayne and M. Zyda, editors. *Mobile Agents*. Prentice-Hall, 1998.

C. A. Coello Coello. August 1999. A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques. *Knowledge and Information Systems. An International Journal*, **1**, 269–308.

C.A. Coello Coello. June 2000. An Updated Survey of GA-Based Multiobjective Optimization Techniques. *ACM Computing Surveys*, **32**, 109–143.

C.A. Coello Coello. Recent Trends in Evolutionary Multiobjective Optimization. In Ajith Abraham, Lakhmi Jain and Robert Goldberg, editors, *Evolutionary Multiobjective Optimizations: Theoretical Advances and Applications*, pages 7–32, London, 2005. Springer-Verlag. ISBN 1-85233-787-7.

C.A. Coello Coello and A.D. Christiansen. 1999. MOSES : A Multiobjective Optimization Tool for Engineering Design. *Engineering Optimization*, **31**, 337–368.

C.A. Coello Coello and Alan D. Christiansen. May 2000. Multiobjective optimization of trusses using genetic algorithms. *Computers and Structures*, **75**, 647–660.

G. Cohen, M. Hilario, H. Sax, S. Hugonnet and A. Geissbuhler. 2006. Learning from imbalanced data in surveillance of nosocomial infection. *Artificial Intelligence in Medicine*, **37**, 7–18.

W.W. Cohen. Fast Effective Rule Induction. In *In Proceedings of the 12th International Conference on Machine Learning*, pages 115–123. Morgan Kaufmann, 1995.

J. P. Cohoon, S. U. Hegde, W. N. Martin and D. Richards. Punctuated equilibria: a parallel genetic algorithm. In *Proceedings of the Second International Conference on Genetic Algorithmson Genetic algorithms and their application*, pages 148–154, Mahwah, NJ, USA, 1987. Lawrence Erlbaum Associates, Inc. ISBN 0-8058-0158-8.

I. Corona, G. Giacinto, C. Mazzariello, F. Roli and C. Sansone. 2009. Information fusion for computer security: State of the art and open issues. *Information Fusion*, **10**, 274–284.

C. Cortes and V. Vapnik. 1995. Support-vector networks. *Machine Learning*, **20**, 273–297.

K. Crammer and Y. Singer. 2002. On the algorithmic implementation of multiclass kernel-based vector machines. *J. Mach. Learn. Res.*, **2**, 265–292. ISSN 1533-7928.

M. Crosbie and E.H. Spafford. Applying Genetic Programming to Intrusion Detection. In *Working Notes for the AAAI Symposium on Genetic Programming*, pages 1–8, MIT, Cambridge, MA, USA, 10–12 1995. AAAI.

F. Cuppens and R. Ortalo. LAMBDA: A Language to Model a Database for Detection of Attacks. In *RAID '00: Proceedings of the Third International Workshop on Recent Advances in Intrusion Detection*, pages 197–216, London, UK, 2000. Springer-Verlag. ISBN 3-540-41085-6.

D. Dal, S. Abraham, A. Abraham, S. Sanyal and M. Sanglikar. Evolution Induced Secondary Immunity: An Artificial Immune System Based Intrusion Detection System. In *CISIM '08: Proceedings of the 2008 7th Computer Information Systems and Industrial Management Applications*, pages 65–70, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-0-7695-3184-7.

N. Dalvi, P. Domingos, Mausam, S. Sanghai and D. Verma. Adversarial Classification. In *Proceedings of the 10th International Conference on Knowledge Discovery and Data Mining*, pages 99–108. ACM Press, 2004.

P.J. Darwen and X. Yao. 1997. Speciation as automatic categorical modularization. *IEEE Transactions on Evolutionary Computation*, **1**, 101–108.

D. Dasgupta, editor. *Artificial Immune Systems and Their Applications*. Springer, 1998.

D. Dasgupta. Immunity Based Intrusion Detection System: A General Framework. In *Proceedings of the 22nd National Information Systems Security Conference*, pages 147–160, 1999.

D. Dasgupta and N. Attoh-Okine. Immunity Based Systems: A Survey. In *IEEE International Conference on Systems, Man and Cybernetics*, Hyatt Orlando, Florida, 1997.

D. Dasgupta and F. González. 2002. An Immunity Based Technique to Characterize Intrusions in Computer Networks. *IEEE Transactions on Evolutionary Computation*, **6**, 1081–1088.

D. Dasgupta, Z. Ji and F. Gonzalez. Artificial Immune System (ais) Research in the Last Five Years. In *The 2003 Congress on Evolutionary Computation, CEC '03*, volume 1, pages 123–130, Canberra, Australia, 8-12 December 2003. IEEE.

L. Davis. Adapting operator probabilities in genetic algorithms. In *3rd International Conference on Genetic Algorithms*, pages 61–69. Morgan Kaufmann, 1989.

R.I.A. Davis, B.C. Lovell and T. Caelli. Improved estimation of hidden markov model parameters from multiple observation sequences. In *Proceedings of the International Conference on Pattern Recognition*, pages 168–171, Quebec City, Canada, August 2002.

L.N. de Castro and J. Timmis. *Artificial Immune Systems: A New Computational Intelligence Approach*. Springer, 2002.

K.A. De Jong. *An analysis of the behavior of a class of genetic adaptive systems*. PhD thesis, University of Michigan, 1975.

K. Deb. Evolutionary Algorithms for Multi-Criterion Optimization in Engineering Design. In Kaisa Miettinen, Marko M. Mäkelä, Pekka Neittaanmäki and Jacques Periaux, editors, *Evolutionary Algorithms in Engineering and Computer Science*, chapter 8, pages 135–161. John Wiley & Sons, Ltd, Chichester, UK, 1999.

K. Deb. 2000. An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, **186**, 311–338.

K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, Chichester, UK, 2001. ISBN 047187339X. ISBN 0-471-87339-X.

K. Deb, S. Agrawal, A. Pratab and T. Meyarivan. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. KanGAL report 200001, Indian Institute of Technology, Kanpur, India, 2000.

K. Deb and D.E. Goldberg. An Investigation of Niche and Species Formation in Genetic Function Optimization. In *Proceedings of the third international conference on Genetic algorithms*, pages 42–50, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc. ISBN 1-55860-006-3.

K. Deb, P. Zope and A. Jain. Distributed Computing of Pareto-Optimal Solutions with Evolutionary Algorithms. In Carlos M. Fonseca, Peter J. Fleming, Eckart Zitzler, KalyanmoyDeb and Lothar Thiele, editors, *Evolutionary Multi-Criterion Optimization. Second International Conference,EMO 2003*, pages 534–549, Faro, Portugal, April 2003. Springer. Lecture Notes in Computer Science. Volume 2632.

H. Debar. An Introduction to Intrusion Detection Systems. In *Connect 2000*, 2000.

H. Debar, M. Becker and D. Siboni. A Neural Network Component for an Intrusion Detection System. In *Proceedings of the 1992 IEEE Symposium on Security and Privacy*, pages 240–250, Washington, DC, USA, 1992. IEEE Computer Society. ISBN 0-8186-2825-1.

H. Debar, M. Dacier and A. Wespi. 1999. Towards a Taxonomy of Intrusion Detection Systems. *Computer Networks*, **31**, 805–822.

H. Debar and B. Dorizzi. An Application of a Recurrent Network to an Intrusion Detection System. In *Proceedings of the International Joint Conference on Neural Networks*, 1992.

K. Deeter, K. Singh, S.Wilson, L. Filipozzi and S.T. Vuong. APHIDS: A Mobile Agent Based Programmable Hybrid Intrusion Detection System. In *Mobility Aware Technologies and Applications/Mobile Agents for Telecommunication Applications*, pages 244–253, Floianópolis, Brazil, 2004.

D.E. Denning. 1987. An intrusion-detection model. *IEEE Transactions on Software Engineering*, **13**, 222–232. ISSN 0098-5589.

O. Depren, M. Topallar, E. Anarim and M.K. Ciliz. 2005. An Intelligent Intrusion Detection System for Anomaly and Misuse Detection in Computer Networks. *Expert systems with Applications*, **29**, 713–722.

E. DeRouin, J. Brown, L. Fausett and M. Schneider. Neural Network Training om Unequally Represented Classes. In *Intelligent Engineering Systems Through Artificial Neural Networks*, pages 135–140, 1991.

J.E. Dickerson and J.A. Dickerson. Fuzzy Network Profiling for Intrusion Detection. In *Proceedings of NAFIPS 19th International Conference of the North American Fuzzy Information Processing Society*, pages 301–306, July 2000.

J.E. Dickerson, J. Juslin, O. Koukousoula and J.A. Dickerson. Fuzzy intrusion detection. In *IFSA World Congress and 20th North American Fuzzy Information Processing Society (NAFIPS) International Conference*, volume 3, pages 1506–1510, July 2001.

T.G. Dietterich. 2000a. An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization. *Machine Learning*, **40**, 139–157. ISSN 0885-6125.

T.G. Dietterich. Ensemble Methods in Machine Learning. In *MCS '00: Proceedings of the 1st International Workshop on Multiple Classifier Systems*, pages 1–15, London, UK, 2000b. Springer-Verlag. ISBN 3-540-67704-6.

C. Dimopoulos. A Review of Evolutionary Multiobjective Optimization Applications in the Area of Production Research. In *2004 Congress on Evolutionary Computation (CEC'2004)*, volume 2, pages 1487–1494, Portland, Oregon, USA, June 2004. IEEE Service Center.

D. Djenouri, L. Khelladi and A.N. Badache. 2005. A survey of security issues in mobile ad hoc and sensor networks. *Communications Surveys & Tutorials*, **7**, 2–28.

P. Dokas, L. Ertoz, V. Kumar, A. Lazarevic, J. Srivastava and P. Tan. Data Mining for Network Intrusion Detection. In *Proceedings of the NSF Workshop on Next Generation Data Mining*, Baltimore, MD, November 2002.

D.J. Doorly, J. Peiró, T. Kuan and J-P. Oesterle. Optimization of Airfoils using Parallel Genetic Algorithms. In *Proceedings of the 15th AIAA International Conference on Numerical Methods in Fluid Mechanics*, 1996.

M. Dorigo, G. Di Caro and L.M. Gambardella. 1999. Ant Algorithms for Discrete Optimization. *Artificial Life*, **5**, 137 –172.

M. Dorigo and T. Stutzle. *Ant Colony Optimization*. The MIT Press, July 2004.

E.M. dos Santos, R. Sabourin and P. Maupin. Single and Multi-Objective Genetic Algorithms for the Selection of Ensemble of Classifiers. In *International Joint Conference on Neural Networks*, pages 3070–3077, 2006.

E.M. dos Santos, R. Sabourin and P. Maupin. Pareto analysis for the selection of classifier ensembles. In *GECCO '08: Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 681–688, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-130-9.

J. Dréo and P. Siarry. 2004. Continuous interacting ant colony algorithm based on dense heterarchy. *Future Generation Computer Systems*, **20**, 841–856.

H. Drucker. Improving Regressors using Boosting Techniques. In *ICML '97: Proceedings of the 14th International Conference on Machine Learning*, pages 107–115, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc. ISBN 1-55860-486-3.

C. Drummond and R.C. Holte. C4.5, Class Imbalance, and Cost Sensitivity: Why Under-Sampling beats Over-Sampling. In *Proceedings of the International Conference on Machine Learning (ICML 2003) Workshop on Learning from Imbalanced Data Sets II*, 2003.

K-B. Duan and S.S. Keerthi. Which is the Best Multiclass SVM Method? An Empirical Study. In *Multiple Classifier Systems*, pages 278–285, 2005.

R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, 1973.

R.O. Duda. *Pattern Classification*. Wiley, 2nd edition, 2001. ISBN 0471056693.

P. Duell, I. Fermin and X. Yao. Speciation Techniques in Evolved Ensembles with Negative Correlation Learning. In *Proc. of the 2006 IEEE Congress on Evolutionary Computation (CEC'06)*, pages 11086–11090, Vancouver, Canada, 16-21 July 2006. IEEE Press.

S.T. Eckmann, G. Vigna and R.A. Kemmerer. 2002. STATL: an attack language for state-based intrusion detection. *Journal of Computer Security*, **10**, 71–103. ISSN 0926-227X.

M. Egmont-Peterson, W.R.M. Dassen and J.H.C. Reiber. 1999. Sequential selection of discrete features for neural networks – A Bayesian approach to building a cascade. *Pattern Recognition Letters*, **20**, 1439–1448. ISSN 0167-8655.

M.E. Elhdhili, L.B. Azzouz and F. Kamoun. August 2008. CASAN: Clustering algorithm for security in ad hoc networks. *Computer Communications*, **31**, 2972–2980.

C. Elkan. Results of the KDD'99 Classifier Learning Contest, 1999. Available online: `http://www-cse.ucsd.edu/~elkan/clresults.html` [Accessed March 5th 2010].

C. Elkan. 2000. Results of the KDD'99 classifier learning. *SIGKDD Explorations Newsletter*, **1**, 63–64. ISSN 1931-0145.

D. Endler. Intrusion Detection: Applying Machine Learning to Solaris Audit Data. In *Proceedings of the Annual Computer Security Applications conference*, pages 267–279, 1998.

V. Engen, J. Vincent and K. Phalp. December 2008. Enhancing Network Based Intrusion Detection for Imbalanced Data. *International Journal of Knowledge-Based and Intelligent Engineering Systems (KES)*, **12**, 357–367.

V. Engen, J. Vincent and K. Phalp. 2011. Exploring Discrepancies in Findings Obtained with the KDD Cup '99 Data Set. *To appear in the International Journal of Intelligent Data Analysis*, **15**.

V. Engen, J. Vincent, A.C. Schierz and K. Phalp. Multi-Objective Evolution of the Pareto Optimal Set of Neural Network Classifier Ensembles. In *Proceedings of the International Conference on Machine Learning and Cybernetics (ICMLC)*, volume 1, pages 74–79, Baoding, China, July 2009. IEEE.

S. Ertekin, J. Huang, L. Bottou and L. Giles. Learning on the border: active learning in imbalanced data classification. In *CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 127–136, New York, NY, USA, 2007a. ACM. ISBN 978-1-59593-803-9.

S. Ertekin, J. Huang and C.L. Giles. Active learning for class imbalance problem. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 823–824, New York, NY, USA, 2007b. ACM. ISBN 978-1-59593-597-7.

L.J. Eshelman and J.D. Schaffer. Preventing Premature Convergence in Genetic Algorithms by Preventing Incest. In Richard K. Belew and Lashon B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms(ICGA'91)*, pages 115–122, San Diego, California, USA, July 1991. Morgan Kaufmann.

L.J. Eshelman and J.D. Schaffer. Real-Coded Genetic Algorithms and Interval-Schemata. In *Proceedings of the Second Workshop on Foundations of Genetic Algorithms(FOGA2)*, pages 187–202. Morgan Kaufmann, 1993. ISBN 1-55860-263-1.

E. Eskin, A. Arnold, M. Prerau, L. Portnoy and S. Stolfo. A Geometric Framework for Unsupervised Anomaly Detection: Detecting Intrusions in Unlabeled Data. In *Data Mining for Security Applications*. Kluwer, 2002.

M. Esmaili, B. Balachandran, R. Safavi-Naini and J. Pieprzyk. Case-Based Reasoning for Intrusion Detection. In *ACSAC '96: Proceedings of the 12th Annual Computer Security Applications Conference*, page 214, Washington, DC, USA, 1996. IEEE Computer Society. ISBN 0-8186-7606-X.

A. Estabrooks and N. Japkowicz. A Mixture-of-Experts Framework for Learning from Imbalanced Data Sets. In *Proceedings of the 4th International Conference on Advances in Intelligent Data Analysis*, pages 34–43, London, UK, 2001. Springer-Verlag. ISBN 3-540-42581-0.

B. S. Everitt. *Cluster Analysis*. Edward Arnold: A division of Hodder & Stoughton, New York, USA, 3 edition, 1993.

E. Fabre, A. Benveniste, S. Haar, C. Jard and A. Aghasaryan. Algorithms for Distributed Fault Management in Telecommunications Networks. In *Proceedings of the International Conference on Telecommunications*, pages 820–825. Springer, 2004.

K.M. Faraoun and A. Boukelif. 2007. Neural Networks Learning Improvement using the K-Means Clustering Algorithm to Neural Networks Learning Improvement. *International Journal of Computational Intelligence*, **3**, 161–168.

Y. Feng, J. Zhong, Z-Y. Xiong, C-X. Ye and K-G. Wu. Network Anomaly Detection Based on DSOM and ACO Clustering. In *ISNN '07: Proceedings of the 4th international symposium on Neural Networks*, pages 947–955, Berlin, Heidelberg, 2007. Springer-Verlag. ISBN 978-3-540-72392-9.

Y. Feng, J. Zhong, C-X. Ye and Z-F. Wu. Clustering based on Self-Organizing Ant Colony Networks with Application to Intrusion Detection. In *ISDA '06: Proceedings of the Sixth International Conference on Intelligent Systems Design and Applications*, pages 1077–1080, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2528-8.

G. Florez, S.M. Bridges and R.B. Vaughn. An Improved Algorithm for Fuzzy Data Mining for Intrusion Detection. In *NAFIPS. 2002 Annual Meeting of the North American Fuzzy Information Processing Society*, 2002.

L.J. Fogel. *On the Organization of Intellect*. PhD thesis, University of California, 1964.

L.J. Fogel. *Intelligence Through Simulated Evolution: forty years of evolutionary programming*. Wiley, 1999. ISBN: 047133250x.

L.J. Fogel, A.J. Owens and M. J. Walsh. *Artificial Intelligence Through Simulated Evolution*. Wiley, 1966.

C.M. Fonseca and P.J. Fleming. Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In Stephanie Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 416–423, San Mateo, California, 1993a. University of Illinois at Urbana-Champaign, Morgan Kauffman Publishers.

C.M. Fonseca and P.J. Fleming. Multiobjective Genetic Algorithms. In *IEE Colloquium on Genetic Algorithms for Control Systems Engineering*, pages 6/1–6/5. IEE, 1993b.

C.M. Fonseca and P.J. Fleming. Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms I: A Unified Formulation. Technical Report 564, University of Sheffield, Sheffield, UK, January 1995.

S. Forrest, A. Perelson, L. Allen and R. Cherukuri. Self Nonself Discrimination in a Computer. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, 1994.

I. Foster. *Designing and Building Parallel Programs - Concepts and Tools for ParallelSoftware Engineering*. Addison-Wesley, 1995. ISBN 0-201-57594-9.

Y. Freund. 1995. Boosting a weak learning algorithm by majority. *Information and Computation*, **121**, 256–285. ISSN 0890-5401.

Y. Freund and R.E. Schapire. 1997. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and Systems Sciences*, **55**, 119–139.

Y. Freund and R.E. Schapire. 1998. Discussion of the paper Ärcing Classifiersb̈y Leo Breiman. *The Annals of Statistics*, **26**, 824–832.

J.H. Friedman. 1997. On Bias, Variance, 0/1—Loss, and the Curse-of-Dimensionality. *Data Mining and Knowledge Discovery*, **1**, 55–77. ISSN 1384-5810.

T.P. Fries. A fuzzy-genetic approach to network intrusion detection. In *GECCO '08: Proceedings of the 2008 GECCO conference companion on Genetic and evolutionary computation*, pages 2141–2146, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-131-6.

I. Gadaras, L. Mikhailov and S. Lekkas. Generation of Fuzzy Classification Rules Directly from Overlapping Input Partitioning. In *IEEE Int. Fuzzy Systems Conf.*, pages 1–6, 2007.

F. Gagnon and B. Esfandiari. Using Artificial Intelligence for Intrusion Detection. In *Proceeding of the 2007 Conference on Emerging Artificial Intelligence Applications in Computer Engineering*, pages 295–306, Amsterdam, The Netherlands, The Netherlands, 2007. IOS Press. ISBN 978-1-58603-780-2.

J.C. Galeano, A. Veloza-Suan and F.A. Gonzalez. A comparative analysis of artificial immune network models. In *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionarycomputation*, pages 361–368, New York, NY, USA, 2005. ACM Press. ISBN 1-59593-010-8.

J. Gama and P. Brazdil. 2000. Cascade generalization. *Machine Learning*, **41**, 315–343. ISSN 0885-6125.

S. García and F. Herrera. 2009. Evolutionary undersampling for classification with imbalanced datasets: Proposals and taxonomy. *Evolutionary Computation*, **17**, 275–306.

N. García-Pedrajas, C. García-Osorio and C. Fyfe. 2007. Nonlinear Boosting Projections for Ensemble Construction. *Machine Learning Research*, **8**, 1–33.

N. Garcia-Pedrajas, C. Hervas-Martinez and D. Ortiz-Boyer. June 2005. Cooperative coevolution of artificial neural network ensembles for pattern classification. *IEEE Transactions on Evolutionary Computation*, **9**, 271–302. ISSN 1089-778X.

A. Geist, A. Beguelin, J. Dongarra, W. Jiang and B. Manchek andV. Sunderam. *PVM: Parallel Virtual Machine - A User's Guide and Tutorial for NetworkParallel Computing*. MIT Press, 1994.

F. Gharibian and A.A. Ghorbani. Comparative Study of Supervised Machine Learning Techniques for Intrusion Detection. In *CNSR '07: Proceedings of the Fifth Annual Conference on Communication Networks and Services Research*, pages 350–358, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-2835-X.

A. Ghosh and A. Schwartzbard. A Study in Using Neural Networks for Anomaly and Misuse Detection. In *Proceedings of the 8th USENIX Security Symposium*, 1999.

A.K. Ghosh, C. Michael and M. Schatz. A Real Time Intrusion Detection System Based on Learning Program Behaviour. In *Proceedings of the Third International Workshop on the Recent Advances in Intrusion Detection*, pages 93–109, 2000.

A.K. Ghosh, A. Schwartzbard and M. Schatz. Learning Program Behaviour Profiles for Intrusion Detection. In *Proc. Workshop on Intrusion Detection and Network Monitoring*, pages 51–62, 1999.

J. C. Giarratano and G. D. Riley. *Expert Systems: Principles and Programming*. Course Technology, Boston, MA USA, 4 edition, 1998.

A. Giordana and F. Neri. 1995. Search-intensive concept induction. *Evolutionary Computiation*, **3**, 375–416. ISSN 1063-6560.

F. Glover. 1989. Tabu search – part i. *ORSA Journal on Computing*, **1**, 190–206.

F. Glover. 1990. Tabu search – part ii. *ORSA Journal on Computing*, **2**, 4–32.

D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Co., Reading, Massachusetts, 1989. ISBN 0-201-15767-5.

D.E. Goldberg and K. Deb. A Comparative Analysis of Selection Schemes Used in Genetic Algorithms. In Gregory J. E. Rawlins, editor, *Proceedings of the First Workshop on Foundations of Genetic Algorithms (FOGA'91)*, pages 69–93, Bloomington Campus, Indiana, USA, July 15-18 1991. Morgan Kaufmann. ISBN 1-55860-170-8.

D.E. Goldberg, K. Deb and J.H. Clark and. 1992. Genetic Algorithms, Noise, and the Sizing of Populations. *Complex Systems*, **6**, 333–362.

D.E. Goldberg and J. Richardson. Genetic Algorithms with Sharing for Multimodal Function Optimization. In *Proceedings of the Second International Conference on Genetic Algorithmson Genetic algorithms and their application*, pages 41–49, Mahwah, NJ, USA, 1987. Lawrence Erlbaum Associates, Inc. ISBN 0-8058-0158-8.

D. Gollmann. *Computer Security*. Wiley, 2nd edition, 2006.

F. González and D. Dasgupta. An Immunogenetic Technique to Detect Anomalies in Network Traffic. In *Proceedings of the Genetic and Evolutionary Compuation Conference (GECCO)*, pages 1081–1088. Morgan Kaufman Publishers, 2002.

F.A. González, J. Gómez, M. Kaniganti and D. Dasgupta. An Evolutionary Approach to Generate Fuzzy Anomaly Signatures. In *IEEE Information Assurance workshop*, pages 251–259, West Point, NY, 2003.

V.S. Gordon and L.D. Whitley. Serial and Parallel Genetic Algorithms as Function Optimizers. In *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 177–183, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc. ISBN 1-55860-299-2.

R. Goss, M. Botha and R. von Solms. Utilizing fuzzy logic and neural networks for effective, preventative intrusion detection in a wireless environment. In *SAICSIT '07: Proceedings of the 2007 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries*, pages 29–35, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-775-9.

J.J. Grefenstette. Genesis: Genetic search implementation system. Available online: `http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/genetic/ga/systems/genesis/0.html` [Accessed 21 August 2009], 1990.

Y. Guan, A. Ghorbani and N. Belacel. Y-means: A Clustering Method for Intrusion Detection. In *Proceedings of the IEEE Canadian Conference on Electrical and Computer Engineering*, 2003.

H. Guo and H.L. Viktor. 2004. Learning from imbalanced data sets with boosting and data generation: the databoost-im approach. *SIGKDD Explorations Newsletter*, **6**, 30–39. ISSN 1931-0145.

D.W. Gürer, I. Khan, R. Ogler and R. Keffer. An Artificial Intelligence Approach to Network Fault Management. In *SRI International*, Menlo Park, CA, 1996.

C.R. Haag, G.B. Lamont, P.D. Williams and G.L. Peterson. An artificial immune system-inspired multiobjective evolutionary algorithm with application to the detection of distributed computer network intrusions. In *GECCO '07: Proceedings of the 2007 GECCO conference companion on Genetic and evolutionary computation*, pages 2717–2724, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-698-1.

J. M. Hall and D. A. Frincke. An Architecture for Intrusion Detection Modeled After the Human Immune System. In *Proceedings of the International Conference on Computer Communication and Control Technologies*, volume 3, pages 75–78, 2003.

K. Han, B. Ravindran and E.D. Jensen. Byzantine-Tolerant, Point-To-Point Information Propagation in Untrustworthy and Unreliable Networks. In *Proc. of the Int. Conf. on Network-Based Information Systems*, 2007.

S-J. Han and S-B. Cho. 2006. Evolutionary neural networks for anomaly detection based on the behavior of a program. *IEEE Transactions on Systems, Man, and Cybernetics*, **36**, 559–570.

J. Handl, D.B. Kell and J. Knowles. Multiobjective optimization in bioinformatics and computational biology. Technical Report TR-COMPSYSBIO-2006-04, UMIST, Manchester, UK, April 2006.

A. Hanemann. A Hybrid Rule-Based/Case-Based Reasoning Approach for Service Fault Diagnosis. In *Proceedings of the 2006 International Symposium on Frontiers in Networking with Applications (FINA 2006)*, Vienna, Austria, 2006.

J.V. Hansen, P.B. Lowry, R.D. Meservy and D.M. McDonald. 2007. Genetic programming for prevention of cyberterrorism through dynamic and evolving intrusion detection. *Decision Support Systems*, **43**, 1362–1374.

L.K. Hansen and P. Salamon. 1990. Neural Network Ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **12**, 993–1001. ISSN 0162-8828.

W.E. Hart, S. Baden, R.K. Belew and S.Kohn. Analysis of the Numerical Effects of Parallelism on a Parallel Genetic Algorithm. In *Proc. of 10th Intl Parallel Processing Symp*, pages 606–612, 1996.

S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 2 edition, 1998.

J. He, D. Long and C. Chen. An Improved Ant-based Classifier for Intrusion Detection. In *ICNC '07: Proceedings of the Third International Conference on Natural Computation (ICNC 2007)*, pages 819–823, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-2875-9.

G. Helmer, J. Wong, V. Honavar and L. Miller. Intelligent agents for intrusion detection. In *IEEE Information Technology Conference*, pages 121–124, 1998.

G. Helmer, J.S.K. Wong, V.G. Honavar and L. Miller. 2002. Automated discovery of concise predictive rules for intrusion detection. *Journal of Systems and Software*, **60**, 165–175. ISSN 0164-1212.

G. Helmer, J.S.K. Wong, V.Honavar, L. Miller and Y. Wang. 2003. Lightweight Agents for Intrusion Detection. *The Journal of Systems and Software*, **67**, 109–122.

Á. Herrero, E. Corchado, M.A. Pellicer and A. Abraham. Hybrid Multi Agent-Neural Network Intrusion Detection with Mobile Visualization. In *Innovations in Hybrid Intelligent Systems*, volume 44 of *Advances in Soft Computing*, pages 320–328. Springer, 2008.

S. Hido and H. Kashima. Roughly balanced bagging for imbalanced data. In *Proceedings of the 2008 SIAM International Conference on Data Mining*, pages 143–152, 2008.

T. Hiroyasu, M. Miki and S. Watanabe. Distributed Genetic Algorithms with a New Sharing Approach in Multiobjective Optimization Problems. In *1999 Congress on Evolutionary Computation*, pages 69–76, Washington, D.C., July 1999. IEEE Service Center.

T.K. Ho. Nearest neighbors in random subspaces. In *SSPR '98/SPR '98: Proceedings of the Joint IAPR International Workshops on Advances in Pattern Recognition*, pages 640–648, London, UK, 1998a. Springer-Verlag. ISBN 3-540-64858-5.

T.K. Ho. 1998b. The Random Subspace Method for Constructing Decision Forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **20**, 832–844. ISSN 0162-8828.

X.D. Hoang, J. Hu and P. Bertok. A multi-layer model for anomaly intrusion detection using program sequences of system calls. In *The 11th IEEE International Conference on Networks (ICON)*, pages 531–536, 2003.

X.D. Hoang, J. Hu and P. Bertok. 2009. A program-based anomaly intrusion detection scheme using multiple detection engines and fuzzy inference. *To appear in the Journal of Network and Computer Applications*.

A. Hofmann and B. Sick. Evolutionary optimization of radial basis function networks for intrusion detection. In *Proc. Int. Conf. on Neural Networks*, pages 415–420. IEEE, July 2003.

S. A. Hofmeyr and S. Forrest. Immunity by Design: An Artificial Immune System. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 1289–1296, 1999.

S. A. Hofmeyr and S. Forrest. 2000. Architecture for an Artificial Immune System. *Evolutionary Computation*, **8**, 443–473.

A.J. Höglund and K. Hätönen. Computer Network User Behaviour Visualisation Using Self Organising Maps. In *Proceedings of the 8th International Conference on Artificial Neural Networks*, pages 899–904, 1998.

J.H. Holland. *Adaptation in Natural and Artificial Systems: an introductory analysis with applications to biology, control and artificial intelligence*. The MIT Press, 2nd edition, 1992. ISBN 0262581116. ISBN 0-262-58111-6.

R. Holte, N. Japkovicz, C. Ling and S. Matwin, editors. *AAAI Workshop on Learning from Imbalanced Data Sets*, 2000. AAAI Press.

R.C. Holte, L.E. Acker and B.W. Porter. Concept learning and the problem of small disjuncts. In *Proceedings of the 11th Joint International Conference on Artificial Intelligence*, pages 813–818, 1989.

J. Horn, N. Nafpliotis and D.E. Goldberg. A Niched Pareto Genetic Algorithm for Multiobjective Optimization. In *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEEWorld Congress on Computational Intelligence*, volume 1, pages 82–87, Piscataway, New Jersey, June 1994. IEEE Service Center.

K. Hornik, M. Stinchcombe and H. White. 1989. Multilayer feedforward networks are universal approximators. *Neural Networks*, **2**, 359–366. ISSN 0893-6080.

H. Hou, J. Zhu and G. Dozier. Artificial Immunity Using Constraint Based Detectors. In *Proceedings of the 5th Biannual World Automation Congress*, pages 239–244, 2002.

W. Hu, Y. Liao and V.R. Vemuri. Robust Anomaly Detection Using Support Vector Machines. In *Proc. Int.Conf. on Machine Learning and Applications*. Morgan Kaufmann, 2003.

Y-A. Huang and W. Lee. A Cooperative Intrusion Detection System for Ad Hoc Networks. In *Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks (SASN)*, pages 135–147, 2003.

Y-M. Huang, C-M. Hung and H.C. Jiau. 2006. Evaluation of neural networks and data mining methods on a credit assessment task for class imbalance problem. *Nonlinear Analysis: Real World Applications*, **7**, 720–747.

K. Hwang and F.A. Briggs. *Computer Architecture and Parallel Processing*. McGraw-Hill, 1986. ISBN 0-07-Y66354-8.

K. Ilgun. USTAT: A Real-Time Intrusion Detection System for UNIX. In *SP '93: Proceedings of the 1993 IEEE Symposium on Security and Privacy*, page 16, Washington, DC, USA, 1993. IEEE Computer Society.

K. Ilgun, R.A. Kemmerer and P. A. Porras. 1995. State transition analysis: A rule-based intrusion detection approach. *Software Engineering*, **21**, 181–199.

H. Ishibuchi and Y. Nojima. 2006. Evolutionary multiobjective optimization for the design of fuzzy rule-based ensemble classifiers. *International Journal of Hybrid Intelligent Systems*, **3**, 129–145. ISSN 1448-5869.

K. Jajuga, A. Sokolowski and H. Bock, editors. *Classification, Clustering and Data Analysis*. Springer, 1 edition, August 15 2002.

G. Jakobson, J. Buford and L. Lewis. Towards and Architecture for Reasoning About Complex Event Based Dynamic Systems. In *Proceedings of the 3rd International Workshop on Distributed Event Based Systems*, 2004.

C. Janikow and Z. Michalewicz. An Experimental Comparison of Binary and Floating-point Representations in Genetic Algorithms. In *Proceedings of the 4th International Conference on Genetic Algorithms*. Morgan Kaufmann, 1991. ISBN 1558602089.

W. Jansen and T. Karygiannis. Applying Mobile Agents to Intrusion Detection and Response. Technical report, NIST Interim Report, October 1999.

W. Jansen, P. Mell, T. Karygiannis and D. Marks. Mobile Agents in Intrusion Detection and Response. In *12th Annual Canadian Information Technology Security Symposium*, Ottawa, Canada, 2000.

N. Japkowicz. Learning from Imbalanced Data Sets: A Comparison of Various Strategies. In *AAAI Workshop on Learning from Imbalanced Data Sets*, pages 10–15, 2000a.

N. Japkowicz. The Class Imbalance Problem: Significance and Strategies. In *Proceedings of the 2000 International Conference on Artificial Intelligence (IC-AI'2000)*, volume 1, pages 111–117, 2000b.

N. Japkowicz. Concept-Learning in the Presence of Between-Class and Within-Class Imbalances. In *AI '01: Proceedings of the 14th Biennial Conference of the Canadian Society on Computational Studies of Intelligence*, pages 67–77, London, UK, 2001. Springer-Verlag. ISBN 3-540-42144-0.

N. Japkowicz. Class Imbalances: Are we Focusing on the Right Issue? In *ICML Workshop on Learning from Imbalanced Data Sets II*, 2003.

S. Jeya and K. Ramar. 2007. Rule Based Network Intrusion Detection System Based on Crossover and Mutation. *Asian Journal of Information Technology*, **6**, 896–901.

G. Jiang and G. Cybenko. Temporal and Spatial Distributed Event Correlation for Network Security. In *Proceedings of the 2004 American Control Conference*, volume 2, pages 996–1001, 2004.

F. Jiménez, J. L. Verdegay and A.F. Gómez-Skarmeta. Evolutionary Techniques for Constrained Multiobjective Optimization Problems. In Annie S. Wu, editor, *Proceedings of the 1999 Genetic and Evolutionary Computation Conference.*, pages 115–116, Orlando, Florida, July 1999.

W. Jing-Xin, W. Zhi-Ying and D. Kui. A Network Intrusion Detection System Based on Artificial Neural Networks. In *Proceedings of the 3rd ACM International Conference on Information Security*, volume 85, pages 166–170, Shanghai, China, 2004.

T. Jo and N. Japkowicz. 2004. Class Imbalances versus Small Disjuncts. *SIGKDD Explorations Newsletter*, **6**, 40–49.

S.S. Joshi and V.V. Phoha. Investigating hidden markov models capabilities in anomaly detection. In *ACM-SE 43: Proceedings of the 43rd annual Southeast regional conference*, pages 98–103, New York, NY, USA, 2005. ACM. ISBN 1-59593-059-0.

P. Kabiri and A.A. Ghorbani. September 2005. Research on Intrusion Detection and Response: A Survey. *International Journal of Network Security*, **1**, 84–102.

O. Kachirski and R. Guba. Effective Intrusion Detection Using Multiple Sensors in Wireless Ad Hoc Networks. In *Proceedings of the 36th Annual Hawaii International Conference on System Sciences*, 2003.

B.L. Kalman and S.C. Kwasny. Why tanh: Choosing a Sigmoidal Function. In *IJCNN: International Joint Conference on Neural Networks*, volume 4, pages 578–581, 1992.

G.V. Kass. 1980. An exploratory technique for investigating large quantities of categorical data. *Applied Statistics*, **29**, 119–127.

S. Kätker and K. Geihs. 1997. A generic model for fault isolation in integrated management system. *J. Network Syst. Manage.*, **5**, 109–130.

I. Katzela and M. Schwartz. 1995. Schemes for fault identification in communication networks. *IEEE/ ACM Transactions on Networking*, **3**, 753–764.

L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. A Wiley Interscience Publication, New York, USA, 1990.

M. Kearns. A Bound on the Error of Cross Validation Using the Approximation and Estimation Rates, with Consequences for the Training-Test Split. In *Advances in Neural Information Processing Systems*, volume 8, pages 183–189. The MIT Press, 1996.

R.A. Kemmerer and G. Vigna. April 2002. Intrusion Detection: A Brief History and Overview. *Computer*, **35**, 27–30.

K. Kendall. A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems. Master's thesis, Massachusetts Institute of Technology, 1999.

J. Kennedy and R. Eberhart. Particle Swarm Optimization. In *IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948, 1995.

K. Kermanidis, M. Maragoudakis, N. Fakotakis and G. Kokkinakis. Learning Greek verb complements: addressing the class imbalance. In *COLING '04: Proceedings of the 20th international conference on Computational Linguistics*, page 1065, Morristown, NJ, USA, 2004. Association for Computational Linguistics.

L. Khan, M. Awad and B. Thuraisingham. 2007. A new intrusion detection system using support vector machines and hierarchical clustering. *The VLDB Journal*, **16**, 507–521. ISSN 1066-8888.

R. Khanna and H. Liu. System Approach to Intrusion Detection using Hidden Markov Model. In *IWCMC '06: Proceedings of the 2006 international conference on Wireless communications and mobile computing*, pages 349–354, New York, NY, USA, 2006. ACM. ISBN 1-59593-306-9.

T.M. Khoshgoftaar, M. Golawala and J. van Hulse. An Empirical Study of Learning from Imbalanced Data Using Random Forest. In *ICTAI '07: Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence - Vol.2 (ICTAI 2007)*, pages 310–317, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-3015-X.

G. H. Kim and P. J. Bentley. An Evaluation of Negative Selection in an Artificial Immune System for Network Intrusion Detection. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, 2001.

G. H. Kim and P. J. Bentley. Towards Artificial Immune Systems for Network Intrusion Detection: An Investigation of Dynamic Clonal Selection. In *Proceedings of the Congress on Evolutionary Computation*, 2002.

H-S. Kim and S-D. Cha. 2004. Efficient masquerade detection using svm based on common command frequency in sliding windows. *IEICE Transactions On Information And Systems Volume*, **E87-D**, 2446–2452.

J. Kim, P.J. Bentley, U. Aickelin, J. Greensmith, G. Tedesco and J. Twycross. December 2007. Immune system approaches to intrusion detection – a review. *Natural computing*, **4**, 413–466.

K-J. Kim and S-B. Cho. 2008. Evolutionary ensemble of diverse artificial neural networks using speciation. *Neurocomputing*, **71**, 1604–1618.

J.D. Knowles and D.W. Corne. 2000. Approximating the nondominated front using the pareto archived evolution strategy. *Evolutionary Computation*, **8**, 149–172. ISSN 1063-6560.

L. Kobyliński and A. Przepiórkowski. Definition Extraction with Balanced Random Forests. In *Advances in Natural Language Processing: Proceedings of the 6th International Conference on Natural Language Processing*, pages 237–247. Springer-Verlag, 2008.

R. Kohavi. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 1995.

U. Kohlmorgen. Parallel Genetic Algorithms. In *1st Nordic Workshop on Genetic Algorithms*, pages 135–143. University of Vaasa, Finland, January 9-12 1995.

T. Kohonen. *Self Organisation and Associative Memory*. Springer, 1989.

A. Kolcz, A. Chowdhury and J. Alspector. Data duplication: an imbalance problem? In *ICML'2003 Workshop on Learning from Imbalanced Data Sets II*, 2003.

N. Kondo, T. Hatanaka and K. Uosaki. Multiobjective Evolutionary RBF Networks and Its Application to Ensemble Learning. In *Proceedings of the International Symposium on Frontiers of Computational Science*, pages 321–325, 2007.

K.B. Korb and A.E. Nicholson. *Bayesian Artificial Intelligence*. Chapman & Hall/CRC, 2003. ISBN:1-58488-387-1.

S. Kotsiantis, D. Kanellopoulos and P. Pintelas. 2006. Handling imbalanced datasets: A review. *GESTS International Transactions on Computer Science and Engineering*, **30**, 25–36.

J.R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

J.R. Koza. *Genetic Programming 2: automatic discovery of reusable programs*. Complex adaptive systems. MIT Press, 1994.

V. Krmíček, P. Čeleda, M. Rehák and M. Pěchouček. Agent-Based Network Intrusion Detection System. In *IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, pages 528–531. IEEE, 2007.

C. Kruegel, D. Mutz, W. Robertson and F. Valeur. Bayesian Event Classification for Intrusion Detection. In *Proceedings of the 19th Annual Computer Security Applications Conference (ACSAC)*, 2003.

C. Kruegel and T. Toth. Distributed pattern detection for intrusion detection. In *Network and Distributed System Security Symposium Conference Proceedings:2002*, 1775 Wiehle Ave., Suite 102, Reston, Virginia 20190, U.S.A., 2002. Internet Society.

C. Kruegel and T. Toth. Using Decision Trees to Improve Signature-Based Intrusion Detection . In *Recent Advances in Intrusion Detection*, pages 173–191, 2003.

C. Kruegel, T. Toth and C. Kerer. Decentralized Event Correlation for Intrusion Detection. In *International Conference on Information Security and Cryptology*, 2001.

C. Kruegel, F. Valeur and G. Vigna. *Intrusion Detection and Correlation: Challenges and Solutions*. Springer-Verlag Telos, 2004.

G.P. Kumar and P. Venkataram. Network performance management using realistic abductive reasoning model. In *Integrated Network Management*, pages 187–198, 1995.

S. Kumar and E. H. Spafford. A Pattern Matching Model for Misuse Intrusion Detection. In *Proceedings of the 17th National Computer Security Conference*, 1994.

L.I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley, 2004.

L.I. Kuncheva and C.J. Whitaker. 2003. Measures of Diversity in Classifier Ensembles and Their Relationship with the Ensemble Accuracy. *Machine Learning*, **51**, 181–207. ISSN 0885-6125.

E. Lacerda, A. de Carvalho and T. Ludermir. 2001. Evolutionary optimization of rbf networks. *International Journal of Neural Systems*, **11**, 287–294.

T. Lane and C. E. Brodley. 1999. Temporal Sequence Learning and Data Reduction for Anomaly Detection. *ACM Transactions on Information and System Security*, **2**, 295–331.

T. Lane and C.E. Brodley. 2003. An Empirical Study of Two Approaches to Sequence Learning for Anomaly Detection. *Mach. Learn.*, **51**, 73–107. ISSN 0885-6125.

M.M. Lankhorst. A Genetic Algorithm for the Induction of Nondeterministic Pushdown Automata. Computing Science Report CS-R 9502, University of Groningen, 1995a.

M.M. Lankhorst. A Genetic Algorithm for the Induction of Pushdown Automata. In *IEEE International Conference on Evolutionary Computation*, pages 741–746, 1995b.

P. Laskov and R. Lippmann, editors. *NIPS 2007 Workshop on Machine Learning in Adversarial Environments for Computer Security*, 2007.

Lawrence Berkeley National Laboratory and ICSI. LBNL/ICSI Enterprise Tracing Project, 2005. Available online: `http://www.icir.org/enterprise-tracing/` [Accessed March 5th 2010].

Y. LeCun, L. Bottou, G.B. Orr and K.-R. Mueller. Efficient BackProp. In *Neural Networks: Tricks of the Trade, this book is an outgrowth of a 1996 NIPS workshop*, pages 9–50, London, UK, 1998. Springer-Verlag. ISBN 3-540-65311-2.

D-H. Lee, D-Y. Kim and J-I. Jung. Multi-Stage Intrusion Detection System Using Hidden Markov Model Algorithm. In *ICISS '08: Proceedings of the 2008 International Conference on Information Science and Security*, pages 72–77, Washington, DC, USA, 2008a. IEEE Computer Society. ISBN 0-7695-3080-X.

K. Lee, J. Kim, K.H. Kwon, Y. Han and S. Kim. 2008b. DDoS attack detection method using cluster analysis. *Expert Systems with Applications*, **34**, 1659–1665.

W. Lee and S.J. Stolfo. 2000. A Framework for Constructing Features and Models for Intrusion Detection Systems. *ACM Transactions on Information and System Security*, **3**, 227–261. ISSN 1094-9224.

W. Lee and D. Xiang. Information Theoretic Measures for Anomaly Detection. In *IEEE Symposium on Security and Privacy*, Oakland, CA, May 2001.

E. Leon, O. Nasaoui and J. Gomez. Anomaly Detection Based on Unsupervised Niche Clustering with Application to Network Intrusion Detection. In *Proceedings of the Congress of Evolutionary Computation, 2004 (CEC2004)*, 2004a.

E. Leon, O. Nasraoui and J. Gomez. Network Intrusion Detection Using Genetic Clustering. In *Genetic and Evolutionary Computation (GECCO)*, pages 1312–1313, 2004b.

K. Leung and C. Leckie. Unsupervised anomaly detection in network intrusion detection using clusters. In *ACSC '05: Proceedings of the 28th Australasian conference on Computer Science*, pages 333–342, Darlinghurst, Australia, Australia, 2005. Australian Computer Society, Inc. ISBN 1-920-68220-1.

L. M. Lewis. A Case-Based Reasoning Approach to the Resolution of Faults in Communication Networks. In *Proceedings of the IFIP TC6/WG6.6 Third International Symposium on Integrated Network Management with participation of the IEEE Communications Society CNOM and with support from the Institute for Educational Services*, pages 671–682. North-Holland, 1993.

X. Li and N. Ye. 2005. A supervised clustering algorithm for computer intrusion detection. *Knowledge and Information Systems*, **8**, 498–509. ISSN 0219-1377.

C-C. Lin and M-S. Wang. 2008. Genetic-clustering algorithm for intrusion detection system. *International Journal of Information and Computer Security*, **2**, 218–234.

M. Lin, K. Tang and X. Yao. Selective Negative Correlation Learning Algorithm for Incremental Learning. In *Proceedings of the 2008 International Joint Conference on Neural Networks (IJCNN2008)*, pages 2526–2531. IEEE Press, 2008.

U. Lindqvist and P. A. Porras. Detecting Computer and Network Misuse Through the Production-Based Expert System Toolset (P-BEST). In *IEEE Symposium on Security and Privacy*, pages 146–161. IEEE Computer Society Press, 1999.

U. Lindqvist and P. A. Porras. eXpert-BSM: A Host Based Intrusion Detection Solution for Sun Solaris. In *Proceedings of the 17th Annual computer Security Applications Conference*, pages 240–251, New Orleans, USA, Dec 2001.

R. Lippmann, D. Fried, I. Graf, J. Haines, K. Kendall, D. McClung, D. Weber, S. Webster, D. Wyschogrod, R. Cunningham and M. Zissman. Evaluating Intrusion Detection Systems: The 1998 DARPA Off-line Intrusion Detection Evaluation. In *Proceedings of the DARPA Information Survivability Conference and Exposition*, volume 2, pages 12–26. IEEE Computer Society Press, 2000a.

R. Lippmann, J.W. Haines, D.J. Fried, J. Korba and K. Das. 2000b. The 1999 DARPA off-line intrusion detection evaluation. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, **34**, 579–595. ISSN 1389-1286.

C-L. Liu. 2008. Partial discriminative training for classification of overlapping classes in document analysis. *Int. Journal on Document Analysis and Recognition*, **11**, 53–65. ISSN 1433-2833.

D-P. Liu, M-W. Zhang and T. Li. Network Traffic Analysis Using Refined Bayesian Reasoning to Detect Flooding and Port Scan Attacks. In *ICACTE '08: Proceedings of the 2008 International Conference on Advanced Computer Theory and Engineering*, pages 1000–1004, Washington, DC, USA, 2008a. IEEE Computer Society. ISBN 978-0-7695-3489-3.

J. Liu, Q. Hu and D. Yu. 2008b. A comparative study on rough set based class imbalance learning. *Knowledge-Based Systems*, **21**, 753–763.

J. Liu, Q. hu and D. Yu. 2008c. A weighted rough set based method developed for class imbalance learning. *Information Sciences*, **178**, 1235–1256.

J. Liu and L. Li. A Distributed Intrusion Detection System Based on Agents. In *PACIIA '08: Proceedings of the 2008 IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application*, pages 553–557, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-0-7695-3490-9.

Y. Liu, C. Comaniciu and H. Man. A Bayesian Game Approach for Intrusion Detection in Wireless Ad Hoc Networks. In *GameNets '06: Proceeding from the 2006 workshop on Game theory for communications and networks*, page 4, New York, NY, USA, 2006. ACM. ISBN 1-59593-507-X.

Y. Liu, Y. Li and H. Man. MAC layer anomaly detection in ad hoc networks. In *Proceedings from the Sixth Annual IEEE SMC Information Assurance Workshop*, pages 402–409, 2005.

Y. Liu and X. Yao. 1998. Negatively correlated neural networks can produce best ensembles. *Australian Journal of Intelligent Information Processing Systems*, **4**, 176–185.

Y. Liu, X. Yao and T. Higuchi. Nov 2000. Evolutionary ensembles with negative correlation learning. *IEEE Transactions on Evolutionary Computation*, **4**, 380–387. ISSN 1089-778X.

A.M. Logar, E.M. Corwin and T.M. English. Implementation of massively parallel genetic algorithms on the maspar mp-1. In *SAC '92: Proceedings of the 1992 ACM/SIGAPP symposium on Applied computing*, pages 1015–1020, New York, NY, USA, 1992. ACM Press. ISBN 0-89791-502-X.

J. Long. *A case-based framework for meta intrusion detection*. PhD thesis, Tallahassee, FL, USA, 2006. Adviser-Schwartz, Daniel G.

J. Long and D.G. Schwartz. 2008. Case-oriented alert correlation. *W. Trans. on Comp.*, **7**, 98–112. ISSN 1109-2750.

D. Lowd and C. Meek. Adversarial Learning. In *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, pages 641–647, New York, NY, USA, 2005. ACM. ISBN 1-59593-135-X.

W. Lu and I. Traore. Detecting new forms of network intrusion using genetic programming. In *The 2003 Congress on Evolutionary Computation*, volume 3, pages 2165–2172, 2003.

G. F. Luger. *Artificial Intelligence*. Addison–Wesley, 5 edition, 2002.

E.D. Lumer and B. Faieta. Diversity and Adaptation in Populations of Clustering Ants. In *SAB94: Proceedings of the third international conference on Simulation of adaptive behavior : from animals to animats 3*, pages 501–508, Cambridge, MA, USA, 1994. MIT Press. ISBN 0-262-53122-4.

W. Ma, D. Tran and D. Sharma. Negative Selection with Antigen Feedback in Intrusion Detection. In *ICARIS '08: Proceedings of the 7th international conference on Artificial Immune Systems*, pages 200–209. Springer-Verlag, 2008. ISBN 978-3-540-85071-7.

R.B. Machado, A. Boukerche, J.B.M. Sobral, K.R.L. Juca and M.S.M.A. Notare. A Hybrid Artificial Immune and Mobile Agent Intrusion Detection Based Model for Computer Network Operations. In *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Workshop 6*, pages 1–8, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2312-9.

J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proc. 5th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, 1967.

S. Madhavi and T.H. Kim. 2008. An intrusion detection system in mobile ad hoc networks. *International Journal of Security and Its Applications (IJSIA)*, **2**, 1–16.

F. Maggi, M. Matteucci and S. Zanero. 2009. Reducing false positives in anomaly detectors through fuzzy alert aggregation. *Information Fusion*, **10**, 300–311.

M.V. Mahoney and P.K. Chan. An analysis of the 1999 darpa/lincoln laboratory evaluation data for network anomaly detection. In *Proc. 6th Intl. Symp. on Recent Advances in Intrusion Detection (RAID 2003)*, volume 2820, pages 220–237, 2003.

K. Makkithaya, N.V.S. Reddy and U.D. Acharya. Improved C-Fuzzy Decision Tree for Intrusion Detection. In *Proc. World Academy of Science, Engineering and Technology*, volume 32, pages 279–283, 2008.

E.P. Markatos, S. Antonatos, M. Polychronakis and K.G. Anagnostakis. ExB: Exclusion Based signature Matching for Intrusion Detection. In *Proceedings of Communications and Computer Networks*, MIT, USA, 2002.

A. Marx. 2008. Malware vs. Anti-Malware: (How) Can We Still Survive? *Virus Bulletin*, **2**, 2.

F. Masulli and G. Valetini. Comparing decomposition methods for classification. In *Proceedings of the 4th International Conference on Knowledge-Based Intelligent Engineering Systems and Allied Technologies*, pages 788–791, 2000.

R.A. Maxion and T.N. Townsend. Masquerade Detection Using Truncated Command Lines. In *DSN '02: Proceedings of the 2002 International Conference on Dependable Systems and Networks*, pages 219–228, Washington, DC, USA, 2002. IEEE Computer Society. ISBN 0-7695-1597-5.

M.A. Mazurowski, P.A. Habas, J.M. Zurada, J.Y. Lo, J.A. Baker and G.D. Tourassi. 2008. Training neural network classifiers for medical decision making: The effects of imbalanced datasets on classification performance. *Neural Networks*, **21**, 427–436.

J. McHugh. 2000. Testing intrusion detection systems: A critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. *ACM Transactions on Information and System Security*, **3**, 262–294. ISSN 1094-9224.

L. Mena and J.A. Gonzalez. Machine learning for imbalanced datasets: Application in medical diagnostic. In *Proceedings of the 19th International FLAIRS Conference*, 2006.

A. Micarelli and G. Sansonetti. A Case-Based Approach to Anomaly Intrusion Detection. In *MLDM '07: Proceedings of the 5th international conference on Machine Learning and Data Mining in Pattern Recognition*, pages 434–448, Berlin, Heidelberg, 2007. Springer-Verlag. ISBN 978-3-540-73498-7.

E. Michailidis, S.K. Katsikas and E. Georgopoulos. Intrusion Detection Using Evolutionary Neural Networks. In *PCI '08: Proceedings of the 2008 Panhellenic Conference on Informatics*, pages 8–12, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-0-7695-3323-0.

Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, Berlin, Germany, 3rd edition, 1999. ISBN 3540606769. ISBN 3-540-60676-9.

C. Michel and L. Mé. ADeLe: an attack description language for knowledge-based intrustion detection. In *Sec '01: Proceedings of the 16th international conference on Informationsecurity: Trusted information*, pages 353–368, 2001.

F.L. Minku, H. Inoue and X. Yao. 2009. Negative correlation in incremental learning. *Natural Computing*, **8**, 289–320.

B. Mirkin. *Mathematical Classification and Clustering*, volume 11 of *Nonconvex Optimization and its Applications*. Kluwer Academic Publishers, 3300 AA Dordrecht, The Netherlands, 1996.

MIT Lincoln Laboratory. DARPA Intrusion Detection Evaluation Data Sets, 2000. Available online: `http://www.ll.mit.edu/IST/ideval/data/data_index.html` [Accessed March 5th 2010].

M. Mitchell. *An Introduction to Genetic Algorithms* . The MIT Press, Cambridge, Massachussets, reprint edition edition, February 6 1998.

T.M. Mitchell. *Machine Learning*. McGraw–Hill, 1997. ISBN: 0-07-115467-1.

D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford and N. Weaver. 2003. Inside the Slammer Worm. *IEEE Security and Privacy*, **1**, 33–39.

M. Moradi and M. Zulkernine. A Neural Network Based System for Intrusion Detection and Classification of Attacks. In *Proceedings of 2004 IEEE International Conference on Advances in Intelligent Systems - Theory and Applications*, 2004.

B. Morin, L. Mé, H. Debar and M. Ducassé. October 2009. A logic-based model to support alert correlation in intrusion detection. *Information Fusion*, **10**, 285–299.

E. Mosqueira-Rey, A. Alonso-Betanzos, B. Guijarro-Berdinas, D. Alonso-Ríos and J.L. Piñeiro. 2009. A Snort-based agent for a JADE multi-agent intrusion detection system. *International Journal of Intelligent Information and Database Systems*, **3**, 107–121.

E. Mosqueira-Rey, A. Alonso-Betanzos, B.B. Río and J.L. Pineiro. A Misuse Detection Agent for Intrusion Detection in a Multi-agent Architecture. In *KES-AMSTA '07: Proceedings of the 1st KES International Symposium on Agent and Multi-Agent Systems*, pages 466–475, Berlin, Heidelberg, 2007. Springer-Verlag. ISBN 978-3-540-72829-0.

S. Mukkamala, G. Janoski and A. Sung. Intrusion Detection Using Neural Networks and Support Vector Machines. In *Proceedings of the 2002 International Joint Conference on Neural Networks*, pages 1702–1707, 2002.

S. Mukkamala and A.H. Sung. Identifying Key Features for Intrusion Detection Using Neural Networks. In *ICCC '02: Proceedings of the 15th international conference on Computer communication*, pages 1132–1138, Washington, DC, USA, 2002. International Council for Computer Communication. ISBN 1-891365-08-8.

S. Mukkamala and A.H. Sung. Artificial Intelligent Techniques for Intrusion Detection. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, 2003.

R. Munroe. Network, 2007. Available online: `http://www.xkcd.com/350/` [Accessed December 17 2007].

D. Mutz, F. Valeur, G. Vigna and C. Kruegel. 2006. Anomalous system call detection. *ACM Trans. Inf. Syst. Secur.*, **9**, 61–93. ISSN 1094-9224.

H.S. Nagesh, S. Goil and A.N. Choudhary. A Scalable Parallel Subspace Clustering Algorithm for Massive Data Sets. In *International Conference on Parallel Processing*, pages 477–484, 2000.

O. Nasraoui and R. Krishnapuram. A Robust Estimator Based on Density and Scale Optimization and its Application to Clustering. In *Proceedings of the Fifth IEEE International Conference on Fuzzy Systems*, volume 2, pages 1031 – 1035, 1999.

National Statistics. Internet Access, 2008. Available online: `http://www.statistics.gov.uk/cci/nugget.asp?id=8` [Accessed May 8th 2009].

M. Negnevitsky. *Artificial Intelligence A Guide to Intelligent Systems*. Addison Wesley, 2nd edition, 2005.

F. Neri. Mining TCP/IP Traffic for Network Intrusion Detection by Using a Distributed Genetic Algorithm. In *Proceedings of the European Conference on Machine Learning*, pages 313–322, Barcelona, Spain, 2000.

P.G. Neumann and P.A.. Porras. Experience with EMERALD to Date. In *First USENIX Workshop on Intrusion Detection and Network Monitoring*, pages 73–80, Santa Clara, California, apr 1999.

A. Nickerson, N. Japkowicz and E. Milios. Using Unsupervised Learning to Guide Re-Sampling in Imbalanced Data Sets. In *Proceedings of the 8th International Workshop on AI and Statistics*, pages 261–265, 2001.

C. Nikolopoulos. *Expert Systems: Introduction to First and Second Generation and Hybrid Knowledge Based Systems*. 0824799275. Marcel Dekker Inc, New York, January 10 1997.

S. Ohta, R. Kurebayashi and K. Kobayashi. Dec 2008. Minimizing false positives of a decision tree classifier for intrusion detection on the internet. *J. Netw. Syst. Manage.*, **16**, 399–419. ISSN 1064-7570.

L.S. Oliveira, M. Morita, R. Sabourin and F. Bortolozzi. Multi-objective Genetic Algorithms to Create Ensemble of Classifiers. In *Evolutionary Multi-Criterion Optimization*, pages 592–606, 2005.

I. Ono and S. Kobayash. A Real-coded Genetic Algorithm for Function Optimization Using Unimodal Normal Distribution Crossover. In *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 246–253. Morgan Kaufmann, 1997.

D.W. Opitz and J.W. Shavlik. A Genetic Algorithm Approach for Creating Neural Network Ensembles. In *Combining Artificial Neural Nets*, pages 79–99. Springer-Verlag, 1999.

A. Orfila, J.M. Estevez-Tapiador and A. Ribagorda. Evolving High-Speed, Easy-to-Understand Network Intrusion Detection Rules with Genetic Programming. In *EvoWorkshops '09: Proceedings of the EvoWorkshops 2009 on Applications of Evolutionary Computing*, pages 93–98, Berlin, Heidelberg, 2009. Springer-Verlag. ISBN 978-3-642-01128-3.

M. Ostaszewski, P. Bouvry and F. Seredynski. Denial of service detection and analysis using idiotypic networks paradigm. In *GECCO '08: Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 79–86, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-130-9.

M. Ostaszewski, F. Seredynski and P. Bouvry. Immune anomaly detection enhanced with evolutionary paradigms. In *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 119–126, New York, NY, USA, 2006. ACM. ISBN 1-59593-186-4.

C-M. Ou and C.R. Ou. Multi-Agent Artificial Immune Systems (MAAIS) for Intrusion Detection: Abstraction from Danger Theory. In *KES-AMSTA '09: Proceedings of the Third KES International Symposium on Agent and Multi-Agent Systems: Technologies and Applications*, pages 11–19, Berlin, Heidelberg, 2009. Springer-Verlag. ISBN 978-3-642-01664-6.

D. Ourston, S. Matzner, W. Stump and B. Hopkins. Applications of Hidden Markov Models to Detecting Multi-Stage Network Attacks. In *HICSS '03: Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS'03) - Track 9*, Washington, DC, USA, 2003. IEEE Computer Society. ISBN 0-7695-1874-5.

S.F. Owens and R.R. Levary. 2006. An adaptive expert system approach for intrusion detection. *Int. J. Secur. Netw.*, **1**, 206–217. ISSN 1747-8405.

N. Öztürk. 2003. Use of genetic algorithm to design optimal neural network structure. *Engineering Computations: International Journal for Computer-Aided Engineering*, **20**, 979–997.

Z.S. Pan, S.C. Chen, G.B Hu and D.Q. Zhang. Hybrid Neural Network and C4.5 for Misuse Detection. In *Machine Learning and Cybernetics*, pages 2463–2467. Xi'an, 2003.

M. Panda and M.R. Patra. 2007. Network intrusion detection using naive bayes. *IJCSNS International Journal of Computer Science and Network Security*, **7**, 258–263.

M. Panda and M.R. Patra. Ensemble of classifiers for detecting network intrusion. In *ICAC3 '09: Proceedings of the International Conference on Advances in Computing, Communication and Control*, pages 510–515, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-351-8.

V. Pareto. *Manual of Political Economy*. The Macmillan Press Ltd, London, 2nd edition, 1972. ISBN 0-333-13545-8.

D. Parrott, X. Li and V. Ciesielski. Multi-objective Techniques in Genetic Programming for Evolving Classifiers. In *The 2005 IEEE Congress on Evolutionary Computation*, volume 2, pages 1141–1148, September 2005.

A. Patcha and J-M. Park. 2007. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Comput. Netw.*, **51**, 3448–3470. ISSN 1389-1286.

S. Peddabachigari, A. Abraham, C. Grosan and J. Thomas. January 2007. Modeling Intrusion Detection Systems Using Hybrid Intelligent Systems. *Journal of Network and Computer Applications*, **30**, 114–132.

M.P. Perrone and L.N. Cooper. When networks disagree: Ensemble methods for hybrid neural networks. In *Artificial Neural Networks for Speech and Vision*, pages 126–142, 1993.

B. Pfahringer. 2000. Winning the kdd99 classification cup: bagged boosting. *SIGKDD Explorations Newsletter*, **1**, 65–66. ISSN 1931-0145.

C. Phua, D. Alahakoon and V. Lee. 2004. Minority report in fraud detection: classification of skewed data. *SIGKDD Explorations Newsletter*, **6**, 50–59. ISSN 1931-0145.

S. Pierre and R. Glitho, editors. *Mobile Agents for Telecommunication Applications*. 3540424601. Springer, Third International Workshop, MATA 2001, 1 edition, August 14-16 2001.

Y. Ping, Y. Yan, H. Yafei, Z. Yiping and Z. Shiyong. Securing ad hoc Networks Through Mobile Agents. In *Proceedings of the 3rd ACM International Conference on Information Security*, pages 125–129, Shanghai, China, 2004.

A. Pinkus. 1999. Approximation Theory of the MLP Model in Neural Networks. *Acta Numerica*, **8**, 143–196.

P.A. Porras and P.G.. Neumann. EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances. In *1997 National Information Systems Security Conference*, oct 1997.

L. Portnoy, E. Eskin and S. Stolfo. Intrusion Detection With Unlabeled Data Using Clustering. In *Proceedings of the ACM Workshop on Data Mining Applied to Security*, 2001.

M.A. Potter and K.A. De Jong. 2000. Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation*, **8**, 1–29. ISSN 1063-6560.

S.T. Powers and J. He. 2008. A hybrid artificial immune system and self organising map for network intrusion detection. *Information Sciences*, **178**, 3024–3042.

J.R. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993. ISBN 1-55860-238-0.

J.R. Quinlan. Bagging, boosting, and c4.5. In *In Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 725–730. AAAI Press, 1996.

R.J. Quinlan. 1991. Improved estimates for the accuracy of small disjuncts. *Machine Learning*, **6**, 93–98.

L.R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. In *Proceedings of the IEEE*, volume 77, pages 257–286, 1989. ISBN 1-55860-124-4.

G. Ramachandran and D. Hart. A P2P intrusion detection system based on mobile agents. In *ACM-SE 42: Proceedings of the 42nd annual Southeast regional conference*, pages 185–190, New York, NY, USA, 2004. ACM. ISBN 1-58113-870-9.

V. Ramos and A. Abraham. ANTIDS: Self-Organized Ant-based Clustering Model for Intrusion Detection System, 2004.

S. Raudys. Multiple Classification Systems in the Context of Feature Extraction and Selection. In *MCS '02: Proceedings of the Third International Workshop on Multiple Classifier Systems*, pages 27–41, London, UK, 2002. Springer-Verlag. ISBN 3-540-43818-1.

M. Rehak, M. Pechoucek, P. Celeda, V. Krmicek, M. Grill and K. Bartos. Multi-agent approach to network intrusion detection. In *AAMAS '08: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems*, pages 1695–1696, Richland, SC, 2008. International Foundation for Autonomous Agents and Multiagent Systems.

C. K. Riesbeck and R. C. Schank. Inside Case-based Reasoning. In *Technical Report*. Lawrence Erlbaum Association, Cambridge, 1989.

I. Rish, M. Brodie, S. Ma, N. Odintsova, A. Beygelzimer, G. Grabarnik and K. Hernandez. 2005. Adaptive Diagnosis in Distributed Systems. *IEEE Transactions on Neural Networks*, **16**, 1088–1109.

F. Roli, G. Giacinto and G. Vernazza. Methods for Designing Multiple Classifier Systems. In *MCS '01: Proceedings of the Second International Workshop on Multiple Classifier Systems*, pages 78–87, London, UK, 2001. Springer-Verlag. ISBN 3-540-42284-6.

B.E. Rosen. 1996. Ensemble learning using decorrelated neural networks. *Connection Science*, **8**, 373–383.

J. Rowe, K. Vinsen and N. Marvin. Parallel GAs for Multiobjective Functions. In Jarmo T. Alander, editor, *Proceedings of the Second Nordic Workshop on Genetic Algorithms and TheirApplications (2NWGA)*, pages 61–70, Vaasa, Finland, August 1996. University of Vaasa.

Y. Rubner, C. Tomasi and L.J. Guibas. 2000. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, **28**, 99–121.

S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, 1995.

D. Ruta and B. Gabrys. Analysis of the Correlation Between Majority Voting Error and the Diversity Measures in Multiple Classifier Systems. In *Proceedings of the 4th International Symposium on Soft Computing*, 2001a.

D. Ruta and B. Gabrys. Application of the Evolutionary Algorithms for Classifier Selection in Multiple Classifier Systems with Majority Voting. In *MCS '01: Proceedings of the Second International Workshop on Multiple Classifier Systems*, pages 399–408, London, UK, 2001b. Springer-Verlag. ISBN 3-540-42284-6.

D. Ruta and B. Gabrys. 2005. Classifier Selection for Majority Voting. *Information Fusion*, **6**, 63–81.

J. Ryan, M.-J. Lin and R. Miikkulainen. Intrusion Detection with Neural Networks. In *Advances in Neural Information Processing Systems 10*, pages 943–949, 1998.

M. Sabhnani and G. Serpen. Application of Machine Learning Algorithms to KDD Intrusion Detection Dataset within Misuse Detection Context. In *Proceedings of the International Conference on Machine Learning, Models, Technologies and Applications (MLMTA 2003)*, volume 1, pages 209–215, 2003.

M. Sabhnani and G. Serpen. 2004. Why machine learning algorithms fail in misuse detection on kdd intrusion detection data set. *Intelligent Data Analysis*, **8**, 403–415.

R. Sadoddin and A. Ghorbani. Alert correlation survey: framework and techniques. In *PST '06: Proceedings of the 2006 International Conference on Privacy, Security and Trust*, pages 1–10, New York, NY, USA, 2006. ACM. ISBN 1-59593-604-1.

K. Samad, E. Ahmed and W. Mahmood. Simplified Clustering Scheme for Intrusion Detection in Mobile Ad Hoc Networks. In *Proceedings of the 13th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, 2005.

C. Sánches, S. Sankaranarayanan, H. Sipma, T.Z.D. Dill and Z. Manna. Event Correlation: Language and Semantics. In *Proceedings of the 3rd International Conference on Embedded Software (EMSOFT)*, pages 323–339. Springer-Verlag, 2003. ISBN 978-3-540-20223-3.

K. Sastry, H. A. Abbass and D. E. Goldberg. Sub-Structural Niching in Non–Stationary Environments. In *Australian Artificial Intelligence Conference*, pages 873–885, 2005.

J.D. Schaffer. Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. In *Proceedings of the 1st International Conference on Genetic Algorithms*, pages 93–100, Mahwah, NJ, USA, 1985. Lawrence Erlbaum Associates, Inc. ISBN 0-8058-0426-9.

R.E. Schapire. June 1990. The strength of weak learnability. *Machine Learning*, **5**, 197–227.

B. Schölkopf, J.C. Platt, J.C. Shawe-Taylor, A.J. Smola and R.C. Williamson. 2001. Estimating the support of a high-dimensional distribution. *Neural Comput.*, **13**, 1443–1471. ISSN 0899-7667.

M. Schonlau, W. DuMouchel, W. Ju, A. Karr, M. Theus and Y. Vardi. 2001. Computer Intrusion: Detecting Masquerades. *Statistical Science*, **16**, 58–74.

S.L. Scott. 2004. A bayesian paradigm for designing intrusion detection systems. *Computational Statistics & Data Analysis*, **45**, 69–83.

S. Selvakani and R.S. Rajesh. 2007. Genetic algorithm for framing rules for intrusion detection. *IJCSNS International Journal of Computer Science and Network Security*, **7**, 285–290.

T.E. Senator. Multi-Stage Classification. In *Proceedings of the 5th IEEE Int. Conf. on Data Mining*, pages 67–74, 2005.

J. Seo and S. Cha. Masquerade Detection Based on SVM and Sequence-Based User Commands Profile. In *ASIACCS '07: Proceedings of the 2nd ACM symposium on Information, computer and communications security*, pages 398–400, New York, NY, USA, 2007. ACM. ISBN 1-59593-574-6.

K. Shafi, T. Kovacs, H.A. Abbass and W. Zhu. 2009. Intrusion detection with evolutionary learning classifier systems. *Natural Computing: an international journal*, **8**, 3–27. ISSN 1567-7818.

H. Shah, J.L. Undercoffer and A. Joshi. Fuzzy Clustering for Intrusion Detection. In *The 12th IEEE International Conference on Fuzzy Systems*, volume 2, pages 1274–1278, 2003.

K. Sheers. 1996. HP OpenView Event Correlation Services. *Hewlett–Packard Journal*, **11**, 31–42.

J. Shum and H.A. Malki. Network Intrusion Detection System Using Neural Networks. In *ICNC '08: Proceedings of the 2008 Fourth International Conference on Natural Computation*, pages 242–246, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-0-7695-3304-9.

W. Sifei and X. Jiayi. An Artificial Immune Clustering Approach to Unsupervised Network Intrusion Detection. In *ISDPE '07: Proceedings of the The First International Symposium on Data, Privacy, and E-Commerce*, pages 511–513, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-3016-8.

C. Sinclair, L. Pierce and S. Matzner. An Application of Machine Learning to Network Intrusion Detection. In *Proceedings of the 15th Annual Computer Security Applications Conference*, pages 371–377, Phoenix, 1999.

R.K. Sivagaminathan and S. Ramakrishnan. 2007. A hybrid approach for feature subset selection using neural networks and ant colony optimization. *Expert Syst. Appl.*, **33**, 49–60. ISSN 0957-4174.

K. Socha and M. Dorigo. 2006. Ant colony optimization for continuous domains. *European Journal of Operational Research*, **185**, 1155–1173.

J. Song, K. Ohira, H. Takakura, Y. Okabe and Y. Kwon. 2008. A Clustering Method for Improving Performance of Anomaly-Based Intrusion Detection System. *IEICE Transactions on Information and Systems*, **E91-D**, 1282–1291. ISSN 0916-8532.

Q. Song, W. Hu and W. Xie. Nov 2002. Robust support vector machine with bullet hole image classification. *IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews*, **32**, 440–448.

E.D. Sontag. November 1992. Feedback stabilization using two-hidden-layer nets. *IEEE Transactions on Neural Networks*, **3**, 981–990.

Sourcefire Inc. Snort, 1999. Available online: `http://www.snort.org` [Accessed March 5th 2010].

E.J. Spinosa, A.P. de Leon F. de Carvalho and J. Gama. Cluster-based novel concept detection in data streams applied to intrusion detection in computer networks. In *SAC '08: Proceedings of the 2008 ACM symposium on Applied computing*, pages 976–980, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-753-7.

N. Srinivas and Kalyanmoy Deb. Fall 1994. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, **2**, 221–248.

E. Stamatatos. 2008. Author identification: Using text sampling to handle the class imbalance problem. *Information Processing and Management*, **44**, 790–799.

K.O. Stanley and R. Miikkulainen. Efficient Evolution of Neural Network Topologies. In *Proceedings of the 2002 Congress on Evolutionary Computation (IEEE)*, pages 1757–1762. IEEE, May 2002.

G. Stein, B. Chen, A.S. Wu and K.A. Hua. Decision Tree Classifier for Network Intrusion Detection With GA-Based Feature Selection. In *ACM-SE 43: Proceedings of the 43rd annual Southeast regional conference*, pages 136–141, New York, NY, USA, 2005. ACM. ISBN 1-59593-059-0.

T. Stibor, P. Mohr, J. Timmis and C. Eckert. Is Negative Selection Appropriate for Anomaly Detection? In *GECCO '05: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation*, pages 321–328, New York, NY, USA, 2005. ACM Press. ISBN 1-59593-010-8.

Y. Sun, M.S. Kamel, A.K.C. Wong and Y. Wang. 2007. Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition*, **40**, 3358–3378.

A.H. Sung and S. Mukkamala. Identifying Important Features for Intrusion Detection Using Support Vector Machines and Neural Networks. In *SAINT '03: Proceedings of the 2003 Symposium on Applications and the Internet*, pages 209–216, Washington, DC, USA, 2003. IEEE Computer Society. ISBN 0-7695-1872-9.

G. Syswerda. Uniform Crossover in Genetic Algorithms. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 2–9, San Mateo, CA, 1989. Morgan Kaufman.

G. Syswerda. A study of reproduction in generational and steady state genetic algorithms. In Gregory J. E. Rawlins, editor, *Proceedings of the First Workshop on Foundations of Genetic Algorithms*, pages 94–101, Indiana, USA, 1991. Morgan Kaufman.

A. Tajbakhsh, M. Rahmati and A. Mirzaei. 2009. Intrusion detection using fuzzy association rules. *Appl. Soft Comput.*, **9**, 462–469. ISSN 1568-4946.

E-G. Talbi, S. Mostaghim, T. Okabe, H. Ishibuchi, G. Rudolph and C.A. Coello Coello. *Multiobjective Optimization: Interactive and Evolutionary Approaches*, chapter Parallel Approaches for Multiobjective Optimization, pages 349–372. Springer-Verlag, Berlin, Heidelberg, 2008. ISBN 978-3-540-88907-6.

A.C. Tan and D. Gilbert. 2003. Ensemble machine learning on gene expression data for cancer classification. *Applied Bioinformatics*, **2**, 75–83.

A.C. Tan, D. Gilbert and Y. Deville. Multi-Class Protein Fold Classification Using a New Ensemble Machine Learning Approach. In *Proceedings of the 14th International Conference on Genome Informatics*, pages 206–217, 2003.

E.K. Tang, P.N. Suganthan and X. Yao. 2006. An Analysis of Diversity Measures. *Machine Learning*, **65**, 247–271.

K. Tang, M. Lin, F.L. Minku and X. Yao. Aug 2009. Selective negative correlation learning approach to incremental learning. *Neurocomputing*, **72**, 2796–2805.

Y. Tang, E. S. Al-Shaer and R. Boutaba. Active Integrated Fault Localization in Communication Networks. In *IM '05: 9th IFIP/IEEE International Symposium on Integrated Network Management*, pages 543–556, May 2005. ISBN 0-7803-9087-3.

S.J. Templeton and K. Levitt. A requires/provides model for computer attacks. In *NSPW '00: Proceedings of the 2000 workshop on New security paradigms*, pages 31–38, New York, NY, USA, 2000. ACM. ISBN 1-58113-260-3.

J.L. Thames, R. Abler and A. Saad. Hybrid intelligent systems for network security. In *ACM-SE 44: Proceedings of the 44th annual Southeast regional conference*, pages 286–289, New York, NY, USA, 2006. ACM Press. ISBN 1-59593-315-8.

The UCI KDD Archive. Kdd cup 1999 data, 1999. Available online: `http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html` [Accessed March 5th 2010].

J. Tian and M. Gao. Network Intrusion Detection Method Based on High Speed and Precise Genetic Algorithm Neural Network. In *NSWCTC '09: Proceedings of the 2009 International Conference on Networks Security, Wireless Communications and Trusted Computing*, pages 619–622, Washington, DC, USA, 2009. IEEE Computer Society. ISBN 978-0-7695-3610-1.

P. Tillapart, T. Thumthawatworn and P. Santiprabhob. Oct 2002. Fuzzy Intrusion Detection System. *AU J.T.*, **6**, 109–114.

J. Timmis. 2007. Artificial immune systems – today and tomorrow. *Natural Computing*, **6**, 1–18.

M. Tomassini. Parallel and Distributed Evolutionary Algorithms: A Review. In *Evolutionary Algorithms in Engineering and Computer Science*, pages 113–133. Wiley, 1999.

C-H. Tsang and S. Kwong. Multi Agent Intrusion Detection System in Industrial Networks Using Ant Colony Clustering Approach and Unsupervised Feature Extraction. In *Proceedings of the IEEE International Conference on Industrial Technology*, pages 51–56, 2005a.

C-H. Tsang, S. Kwong and H. Wang. 2007. Genetic-fuzzy rule mining approach and evaluation of feature selection techniques for anomaly intrusion detection. *Pattern Recognition*, **40**, 2373–2391. ISSN 0031-3203.

W. Tsang and S. Kwong. Unsupervised Anomaly Intrusion Detection Using Ant Colony Clustering Model. In *Proceedings of the 4th IEEE International Workshop on Soft Computing as Transdiciplinary Science and Technology*, volume 29, pages 223–232, 2005b.

University of New Mexico. Computer Immune Systems Data Sets - Synthetic sendmail, 1996. Available online: `http://www.cs.unm.edu/~immsec/data/synth-sm.html` [Accessed March 5th 2010].

R. Vaarandi. SEC - A Lightweight Event Correlation Tool. In *Proceedings of the IEEE Workshop on IP Operations & Management*, pages 111–115, 2002.

A. Valdes and K. Skinner. Adaptive, model-based monitoring for cyber attack detection. In H. Debar, L. Me and F. Wu, editors, *Recent Advances in Intrusion Detection (RAID 2000)*, number 1907 in Lecture Notes in Computer Science, pages 80–92, Toulouse, France, October 2000. Springer-Verlag.

G. Valentini and F. Masulli. Ensembles of Learning Machines. In *WIRN VIETRI 2002: Proceedings of the 13th Italian Workshop on Neural Nets-Revised Papers*, pages 3–22, London, UK, 2002. Springer-Verlag. ISBN 3-540-44265-0.

J.D. van Hulse, T.M. Khoshgoftaar and A. Napolitano. Experimental Perspectives on Learning from Imbalanced Data. In *The 24th International Conference on Machine Learning*, 2007.

D.A. Van Veldhuizen and G.B. Lamont. 2000. Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art. *Evolutionary Computation*, **8**, 125–147.

D.A. Van Veldhuizen, J.B. Zydallis and G.B. Lamont. Issues in Parallelizing Multiobjective Evolutionary Algorithms for Real World Applications. In *Proceedings of the 17th ACM Symposium on Applied Computing*, pages 595–602, Madrid, Spain, 2002. ACM Press.

R. Verwoerd and R. Hunt. September 2002. Intrusion Detection Techniques and Approaches. *Computer Communications*, **25**, 1356–1365.

J. Vincent, V. Engen, S. Langenberg, K. Phalp, R. Mintram and C. Anyakoha. A Notation and Representation for Describing and Evolving Correlation Patterns. In *IASTED International Conference on Artificial Intelligence and Soft Computing (ASC 2007)*, 2007.

D.G. Wang, T. Li, S.J. Liu, G. Liang and K. Zhao. An Immune Multi-agent System for Network Intrusion Detection. In *ISICA '08: Proceedings of the 3rd International Symposium on Advances in Computation and Intelligence*, pages 436–445, Berlin, Heidelberg, 2008a. Springer-Verlag. ISBN 978-3-540-92136-3.

H-B. Wang, Z. Yuan and C-D. Wang. Intrusion Detection for Wireless Sensor Networks Based on Multi-agent and Refined Clustering. In *CMC '09: Proceedings of the 2009 WRI International Conference on Communications and Mobile Computing*, pages 450–454, Washington, DC, USA, 2009. IEEE Computer Society. ISBN 978-0-7695-3501-2.

W. Wang, X. Guan and X. Zhang. 2008b. Processing of massive audit data streams for real-time anomaly intrusion detection. *Comput. Commun.*, **31**, 58–72. ISSN 0140-3664.

X. Wang, J. Zheng, K. Xiao, X. Xue and C.K. Toh. A Mobile Agent-based P2P Model for Autonomous Security Hole Discovery. In *Proceedings of the The Fifth International Conference on Computer and Information Technology*, pages 723–727, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2432-X.

C. Warrender, S. Forrest and B. Pearlmutter. Detecting Intrusions Using System Calls: Alternative Data Models. In *Proceedings of the IEEE Symposium on Security and Privacy*, 1999.

R.A. Wasniowski. Multi-sensor agent-based intrusion detection system. In *InfoSecCD '05: Proceedings of the 2nd annual conference on Information security curriculum development*, pages 100–103, New York, NY, USA, 2005. ACM. ISBN 1-59593-261-5.

Web Application Security Consortium. WHID annual report for 2007, 2008. Available online: `http://www.webappsec.org/projects/whid/statistics.shtml` [Accessed May 8th 2009].

G.I. Webb. 2000. Multiboosting: A technique for combining boosting and wagging. *Machine Learning*, **40**, 159–196. ISSN 0885-6125.

G.M. Weiss. Learning with Rare Cases and Small Disjuncts. In *Proceedings of the 12th Internaional Conference on Machine Learning*, pages 558–565. Morgan Kaufmann, 1995.

G.M. Weiss. 2004. Mining with rarity: A unifying framework. *SIGKDD Explorations Newsletter*, **6**, 7–19. ISSN 1931-0145.

G.M. Weiss and F. Provost. 2003. Learning when training data are costly: The effect of class distribution on tree induction. *Journal of Artificial Intelligence Research*, **19**, 315–354.

S.J. Wellington and J.D. Vincent. Design optimisation of the Nadaraya-Watson fuser using a genetic algorithm. In *Proceedings of the 5th International Conference on Information Fusion*, volume 1, pages 327–332. IEEE, 2002.

D. Whitley. The GENITOR Algorithm and Selection Pressure: Why Rank-Based Allocation of Reproductive Trials is Best. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, San Mateo, CA, 1989. Morgan Kaufman.

L.D. Whitley. *Evolutionary Computation 1: Basic Algorithms and Operators*, chapter Permutations, pages 139–150. Institute of Physics Publishing, 2000.

I.H. Witten and E. Frank. *Data Mining: Practical machine Learning Tools and Techniques*. Morgan Kaufmann, 2nd edition, 2005.

D.H. Wolpert and W.G. Macready. April 1997. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, **1**, 67–82.

A.H. Wright. Genetic Algorithms for Real Parameter Optimization. In Gregory J. Rawlins, editor, *Foundations of genetic algorithms*, pages 205–218, San Mateo, CA, 1991. Morgan Kaufmann.

C. Wright, F. Monrose and G.M. Masson. HMM Profiles for Network Traffic Classification. In *VizSEC/DMSEC '04: Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, pages 9–15, New York, NY, USA, 2004. ACM. ISBN 1-58113-974-8.

S-Y. Wu and E. Yen. 2009. Data mining-based intrusion detectors. *Expert Syst. Appl.*, **36**, 5605–5612. ISSN 0957-4174.

C. Xiang, P.C. Yong and L.S. Meng. 2008. Design of multiple-level hybrid classifier for intrusion detection system using Bayesian clustering and decision trees. *Pattern Recogn. Lett.*, **29**, 918–924. ISSN 0167-8655.

Y. Xie, X. Li, E.W.T. Ngai and W. Ying. 2009. Customer Churn Prediction using Improved Balanced Random Forests. *Expert Systems with Applications*, **36**, 5445–5449.

C. Xu, X-S. Chen, H. Zhao, Y-M. Jiang, N. Liu and T-F. Wang. Research of DoS Intrusion Real-time Detection Based on Danger Theory. In *ISDPE '07: Proceedings of the The First International Symposium on Data, Privacy, and E-Commerce*, pages 209–211, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-3016-8.

J. Yang, X. Liu, T. Li, G. Liang and S. Liu. 2009. Distributed agents model for intrusion detection based on ais. *Knowledge-Based Systems*, **22**, 115–119. ISSN 0950-7051.

S. Yang, A. Browne and P.D. Picton. Multistage Neural Network Ensembles. In *Multiple Classifier Systems*, pages 91–97, 2002.

X. Yao. 1993. A review of evolutionary artificial neural networks. *International Journal of Intelligent Systems*, **8**, 539–567.

X. Yao. 1999. Evolving Artificial Neural Networks. *Proceedings of the IEEE*, **87**, 1423–1447.

X. Yao and Md.M. Islam. 2008. Evolving artificial neural network ensembles. *IEEE Computational Intelligence Magazine*, **3**, 31–42.

X. Yao and Y. Liu. Ensemble structure of evolutionary artificial neural networks. In *Proc. of the 1996 IEEE Int'l Conf. on Evolutionary Computation (ICEC'96)*, pages 659–664, 1996.

D. Ye, Q. Bai, M. Zhang and Z. Ye. P2P Distributed Intrusion Detections by Using Mobile Agents. In *ICIS '08: Proceedings of the Seventh IEEE/ACIS International Conference on Computer and Information Science (icis 2008)*, pages 259–265, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-0-7695-3131-1.

N. Ye and X. Li. A scalable clustering technique for intrusion signature recognition. In *The 2001 IEEE Workshop on Information Assurance and Security*, United States Military Academy, West Point, NY, June 2001. IEEE Publications.

S.A. Yemini, S. Kliger, E. Mozes, Y. Yemini and D. Ohsie. 1996. High Speed and Robust Event Correlation. *IEEE Communications Magazine*, **34**, 82–90.

Y. Yuan and D. Guanzhong. 2007. An Intrusion Detection Expert System with Fact-Base. *Asian Journal of Information Technology*, **6**, 614–617.

K.H. Yung. *Update algorithms for masquerade detection*. PhD thesis, Stanford University, Stanford, CA, USA, 2004.

L.A. Zadeh. 1965. Fuzzy sets. *Information and Control*, **8**, 338–353.

G. Zenobi and P. Cunningham. Using Diversity in Preparing Ensembles of Classifiers Based on Different Feature Subsets to Minimize Generalization Error. In *EMCL '01: Proceedings of the 12th European Conference on Machine Learning*, pages 576–587, London, UK, 2001. Springer-Verlag. ISBN 3-540-42536-5.

J. Zhang and Y. Liang. A novel intrusion detection model based on danger theory. In *IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application*, volume 2, pages 867–871, 2008.

J. Zhang and M. Zulkernine. A Hybrid Network Intrusion Detection Technique Using Random Forests. In *ARES '06: Proceedings of the First International Conference on Availability, Reliability and Security*, pages 262–269, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2567-9.

Y. Zhang and W. Lee. Intrusion detection in wireless ad-hoc networks. In *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 275–283, New York, NY, USA, 2000. ACM. ISBN 1-58113-197-6.

Y-F. Zhang, Z-Y. Xiong and X-Q. Wang. Distributed intrusion detection based on clustering. In *Proceedings of 2005 International Conference on Machine Learning and Cybernetics*, volume 4, pages 2379–2383, 2005.

S. Zhicai, J. Zhenzhou and H. Mingzeng. A Novel Distributed Intrusion Detection Model Based on Mobile Agent. In *Proceedings of the International Conference on Information Security*, Shanghai, China, 2004.

C.V. Zhou, C. Leckie and S. Karunasekera. 2010. A survey of coordinated attacks and collaborative intrusion detection. *Computers & Security*, **29**, 124–140.

J. Zhou, M. Heckman, B. Reynolds, A. Carlson and M. Bishop. 2007. Modeling network intrusion detection alerts for correlation. *ACM Trans. Inf. Syst. Secur.*, **10**, 4. ISSN 1094-9224.

Z-H. Zhou and X-Y. Liu. 2006. Training cost-sensitive neural networks with methods addressing the class imbalance problem. *IEEE Transactions on Knowledge and Data Engineering*, **18**, 63–77.

Z-H. Zhou, J. Wu and W. Tang. 2002. Ensembling neural networks: many could be better than all. *Artificial Intelligence*, **137**, 239–263. ISSN 0004-3702.

Z-H. Zhou, J-X. Wu, Y. Jiang and S-F. Chen. Genetic Algorithm based Selective Neural Network Ensemble. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, pages 797–802, 2001.

D. Zhu and A.S. Sethi. SEL, a New Event Pattern Specification Language for Event Correlation. In *Proceedings of the 10th International Conference on Computer Communicationsand Networks*, pages 586–589. IEEE, 2001.

E. Zio, P. Baraldi and G. Gola. 2008. Feature-based classifier ensembles for diagnosing multiple faults in rotating machinery. *Applied Soft Computing*, **8**, 1365–1380.

E. Zitzler, K. Deb and L. Thiele. Summer 2000. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, **8**, 173–195.

E. Zitzler, M. Laumanns and L. Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. In *Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems*, pages 95–100, 2002.

E. Zitzler and L. Thiele. Multiobjective Optimization Using Evolutionary Algorithms—A Comparative Study. In A. E. Eiben, editor, *Parallel Problem Solving from Nature V*, pages 292–301, Amsterdam, September 1998. Springer-Verlag.

E. Zitzler and L. Thiele. November 1999. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, **3**, 257–271.

A. Zomorodian. Context-Free Language Induction by Evolution of Deterministic Push-Down Automata Using Genetic Programming. In *Working Notes for the AAAI Symposium on Genetic Programming*, pages 127–133, 1996.

APPENDIX A

KDD Cup '99 data set

The KDD Cup '99 data set stems from data gathered at MIT Lincoln Laboratory (2000) under sponsorship of the Defense Advanced Research Projects Agency (DARPA) to evaluate Intrusion Detection Systems (IDSs) in 1998 (Lippmann *et al.* 2000a) and 1999 (Lippmann *et al.* 2000b). These two data sets are referred to as DARPA98 and DARPA99, which consist of raw *tcpdump* data from a simulated medium sized US air force base. The KDD Cup '99 data set was provided by Stolfo and Lee for the Knowledge Discovery and Data Mining Tools competition (and associated conference) in 1999 (Elkan 2000). This is a transformed version of the DARPA *tcpdump* data, consisting of a set of features considered suitable for classification with machine learning algorithms. The data set consists of 41 features, some of which are intrinsic to the network connections, whilst other are created using domain knowledge. Refer to Lee and Stolfo (2000) for further details.

There are three partitions of the KDD Cup '99 data available online (The UCI KDD Archive 1999): a full training set (4,898,431 instances), a 10% version of this training set, and a test set (311,029 instances). The test set includes 17 new attacks. The intrusions are commonly grouped into 4 classes, according to the taxonomy of Kendall (1999): *Probing/Surveillance*, *Denial of Service* (*DoS*), *User to Root* (*U2R*) and *Remote to Local* (*R2L*). Some intrusions in the KDD Cup '99 data set are not described by Kendall (1999), but are grouped here according to that of the KDD contest (Elkan 1999), with two exceptions due to inconsistencies. According to the KDD classification, three attacks were present in two categories: *httptunnel* and *multihop* were present in *U2R* and *R2L*, but are kept only as *R2L* here; *warezmaster* was classified as *R2L* for training, but as *DoS* during testing, but is consistently kept as *R2L* here. An overview of the intrusions, grouped according to the these classes, is provided in Table A.1. The number of instances for each of the attack types are listed in Table A.2, followed by the proportions of each attack class in Table A.3.

Table A.1: Attack types grouped to their respective classes. Legend: normal font for attacks in the training sets; *italic* for attacks also present in the test set, with new attacks in ***bold***.

Probing (1)	DoS (2)	U2R (3)	R2L (4)
ipsweep	***apache2***	*buffer_overflow*	*ftp_write*
mscan	*back*	*loadmodule*	*guess_passwd*
nmap	*land*	*perl*	***httptunnel***
portsweep	***mailbomb***	***ps***	*imap*
saint	*neptune*	*rootkit*	*multihop*
satan	*pod*	***sqlattack***	***named***
	processtable	***xterm***	*phf*
	smurf		***sendmail***
	teardrop		***snmpgetattack***
	udpstorm		***snmpguess***
			spy
			warezclient
			warezmaster
			worm
			xlock
			xsnoop

Table A.2: An overview of the number of instances of each attack in the KDD Cup '99 data set.

Attack name	Class	Training (full)	Training (10%)	Test
apache2	DoS	———	———	794
back	DoS	2203	2203	1098
buffer_overflow	U2R	30	30	22
ftp_write	R2L	8	8	3
guess_passwd	R2L	53	53	4367
httptunnel	R2L	———	———	158
imap	R2L	12	12	1
ipsweep	Probing	12481	1247	306
land	DoS	21	21	9
loadmodule	U2R	9	9	2
mailbomb	DoS	———	———	5000
multihop	R2L	7	7	18
mscan	Probing	———	———	1053
named	R2L	———	———	17
neptune	DoS	1072017	107201	58001
nmap	Probing	2316	231	84
normal	Normal	972780	97278	60593
perl	U2R	3	3	2
phf	R2L	4	4	2
pod	DoS	264	264	87
portsweep	Probing	10413	1040	354

Attack name	Class	Training (full)	Training (10%)	Test
processtable	DoS	————	————	759
ps	U2R	————	————	16
rootkit	U2R	10	10	13
saint	Probing	————	————	736
satan	Probing	15892	1589	1633
sendmail	R2L	————	————	17
smurf	DoS	2807886	280790	164091
snmpgetattack	R2L	————	————	7741
snmpguess	R2L	————	————	2406
spy	R2L	2	2	————
sqlattack	U2R	————	————	2
teardrop	DoS	979	979	12
udpstorm	DoS	————	————	2
warezclient	R2L	1020	1020	————-
warezmaster	R2L	20	20	1602
worm	R2L	————	————	2
xlock	R2L	————	————	9
xsnoop	R2L	————	————	4
xterm	U2R	————	————	13
SUM		4898430	494021	311029

Table A.3: Proportions of attack classes in the KDD Cup ’99 data set.

Class	Training (full)	Training (10%)	Test
Normal	972,780 (19.86%)	97,278 (19.69%)	60,593 (19.48%)
Probing	41,102 (0.84%)	4,107 (0.83%)	4166 (1.34%)
DoS	3,883,370 (79.30%)	391,458 (79.24%)	229,853 (73.90%)
U2R	52 (0.00%)	52 (0.01%)	70 (0.02%)
R2L	1126 (0.02%)	1,126 (0.23%)	16,347 (5.26%)

APPENDIX B

Additional results

Additional results are provided in this appendix to provide supplementary information related to investigations that were not discussed in full detail in the thesis to allow for a more focused discussion. Additional results from the first empirical investigation, related to the discrepancies in findings obtained with the KDD Cup '99 data set, are given in Section B.1. Thereafter, Section B.2 includes results from two empirical investigations that were only discussed in brief in Chapter 5. Additional results from the third, and final, empirical investigation are provided in Section B.3.

B.1 Part one – discrepancies study

The additional results listed here are for *holdout validation*, as the results discussed in Chapter 4 are for cross validation. The following abbreviations are used: DT refers to the Decision Tree classifier; NB refers to the naïve Bayes classifier; CF refers to the confidence factors of the standard pruning algorithm for the DT; REP refers to reduced error pruning for the DT; [o] refers to original data; [d] refers to duplicates having been removed; [n] refers to normal instances identical to intrusions having been removed.

B.1.1 On original data

The results on the training and merged data sets are given in respective sections below.

B.1.1.1 Training set only

Table B.1: DT with CF 0.50 on the training set.

Act \ Pred	Normal	Probing	DoS	U2R	R2L	%correct
Normal	19410	1	44	1	0	99.76
Probing	4	819	0	0	0	99.51
DoS	128	159	78006	0	1	99.63
U2R	9	0	0	1	0	10
R2L	30	12	0	1	183	80.97
%correct	99.13	82.64	99.94	33.33	99.46	

Table B.2: DT with REP on the training set.

Act \ Pred	Normal	Probing	DoS	U2R	R2L	%correct
Normal	19405	1	43	1	6	99.74
Probing	13	810	0	0	0	98.42
DoS	57	238	77998	0	1	99.62
U2R	10	0	0	0	0	0
R2L	10	12	0	1	203	89.82
%correct	99.54	76.34	99.94	0	96.67	

Table B.3: NB on the training set.

Act \ Pred	Normal	Probing	DoS	U2R	R2L	%correct
Normal	18696	585	0	47	128	96.09
Probing	1	821	1	0	0	99.76
DoS	175	20474	57153	488	4	73
U2R	1	1	0	8	0	80
R2L	1	0	0	3	222	98.23
%correct	99.06	3.75	100	1.47	62.71	

B.1.1.2 Merged data set

Table B.4: DT with CF 0.05 on the merged data set.

Act \ Pred	Normal	Probing	DoS	U2R	R2L	%correct
Normal	29283	38	53	23	2178	92.74
Probing	129	1508	10	0	10	91.01
DoS	8	1	124258	0	0	99.99
U2R	11	0	0	16	1	57.14
R2L	608	0	1	3	2888	82.51
%correct	97.48	97.48	99.95	38.1	56.88	

Table B.5: DT with REP on the merged data set.

Act \ Pred	Normal	Probing	DoS	U2R	R2L	%correct
Normal	29151	42	48	1	2333	92.32
Probing	76	1571	9	0	1	94.81
DoS	16	3	124238	0	10	99.98
U2R	13	0	0	14	1	50
R2L	604	0	1	2	2893	82.66
%correct	97.63	97.22	99.95	82.35	55.23	

Table B.6: NB on the merged data set.

Act \ Pred	Normal	Probing	DoS	U2R	R2L	%correct
Normal	23585	118	362	70	7440	74.7
Probing	89	1558	4	6	0	94.03
DoS	1192	1029	121274	531	241	97.59
U2R	1	0	0	25	2	89.29
R2L	58	30	0	22	3390	96.86
%correct	94.62	56.97	99.7	3.82	30.62	

B.1.2 Removing normal instances identical to intrusions

Table B.7: DT CF 0.25 on the merged data set; normal instances identical to intrusions removed.

Act \ Pred	Normal	Probing	DoS	U2R	R2L	%correct
Normal	29310	37	36	10	567	97.83
Probing	121	1523	12	0	1	91.91
DoS	6	1	124254	0	6	99.99
U2R	8	0	0	16	4	57.14
R2L	28	0	1	3	3468	99.09
%correct	99.45	97.57	99.96	55.17	85.71	

Table B.8: DT with REP on the merged data set; normal instances identical to intrusions removed.

Act \ Pred	Normal	Probing	DoS	U2R	R2L	%correct
Normal	29275	46	37	4	598	97.71
Probing	93	1548	11	0	5	93.42
DoS	22	4	124241	0	0	99.98
U2R	14	0	0	13	1	46.43
R2L	56	0	1	1	3442	98.34
%correct	99.37	96.87	99.96	72.22	85.07	

Table B.9: NB on the merged data set; normal instances identical to intrusions removed.

Act \ Pred	Normal	Probing	DoS	U2R	R2L	%correct
Normal	24862	129	474	73	4422	82.98
Probing	93	1554	4	6	0	93.78
DoS	1148	1028	121268	565	258	97.59
U2R	1	0	0	25	2	89.29
R2L	55	30	0	22	3393	96.94
%correct	95.04	56.69	99.61	3.62	42.02	

B.1.3 Removing duplicates from training set

The results on the training and merged data sets are given in respective sections below.

B.1.3.1 Training set only

Table B.10: DT with CF 0.05 on the training set; duplicates removed.

Act \ Pred	Normal	Probing	DoS	U2R	R2L	%correct
Normal	17516	3	37	2	9	99.71
Probing	2	427	0	0	0	99.53
DoS	148	9694	1075	0	0	9.85
U2R	9	0	0	1	0	10
R2L	4	1	0	2	194	96.52
%correct	99.08	4.22	96.67	20	95.57	

Table B.11: DT with REP on the training set; duplicates removed.

Act \ Pred	Normal	Probing	DoS	U2R	R2L	%correct
Normal	17510	6	34	2	15	99.68
Probing	3	426	0	0	0	99.3
DoS	25	9806	1086	0	0	9.95
U2R	9	0	0	1	0	10
R2L	7	8	0	2	184	91.54
%correct	99.75	4.16	96.96	20	92.46	

Table B.12: NB on the training set; duplicates removed.

Act \ Pred	Normal	Probing	DoS	U2R	R2L	%correct
Normal	14207	2927	3	240	190	80.87
Probing	0	429	0	0	0	100
DoS	65	10064	563	222	3	5.16
U2R	0	2	0	8	0	80
R2L	1	0	0	3	197	98.01
%correct	99.54	3.2	99.47	1.69	50.51	

B.1.3.2 Merged data set

Table B.13: DT with CF 0.05 on the merged data set; duplicates removed

Act \ Pred	Normal	Probing	DoS	U2R	R2L	%correct
Normal	26893	65	55	5	50	99.35
Probing	18	920	6	0	5	96.94
DoS	43	14	14217	0	0	99.6
U2R	10	0	0	18	0	64.29
R2L	44	0	1	3	767	94.11
%correct	99.57	92.09	99.57	69.23	93.31	

Table B.14: DT with REP on the merged data set; duplicates removed

Act \ Pred	Normal	Probing	DoS	U2R	R2L	%correct
Normal	26930	22	56	3	57	99.49
Probing	17	920	6	0	6	96.94
DoS	55	15	14204	0	0	99.51
U2R	12	0	0	16	0	57.14
R2L	40	1	1	3	770	94.48
%correct	99.54	96.03	99.56	72.73	92.44	

Table B.15: NB on the merged data set; duplicates removed

Act \ Pred	Normal	Probing	DoS	U2R	R2L	%correct
Normal	26400	118	5	53	492	97.53
Probing	6	939	4	0	0	98.95
DoS	422	378	12920	228	326	90.51
U2R	1	0	0	23	4	82.14
R2L	60	27	0	14	714	87.61
%correct	98.18	64.23	99.93	7.23	46.48	

B.1.4 Summaries

Table B.16: Overview of classification rates on the training set.

Tech. [data]	Accuracy%	TPR%	FPR%	Normal%	Probing%	DoS%	U2R%	R2L%
DT CF [o]	99.61	99.78	0.24	99.76	99.51	99.63	10	80.97
DT CF [d]	65.97	98.59	0.29	99.71	99.53	9.85	10	96.52
DT REP [o]	99.60	99.89	0.26	99.74	98.42	99.62	0	89.82
DT REP [d]	65.95	99.62	0.32	99.68	99.30	9.95	10	91.54
NB [o]	77.83	99.78	3.19	96.09	99.76	73	80	98.23
NB [d]	52.89	99.43	19.13	80.87	100	5.16	80	98.01

Table B.17: Overview of classification rates on the merged data set.

Tech. [data]	Accuracy%	TPR%	FPR%	Normal%	Probing%	DoS%	U2R%	R2L%
DT CF [o]	98.05	99.42	7.46	92.54	91.01	99.99	57.14	82.51
DT CF [d]	99.26	99.28	0.65	99.35	96.94	99.60	64.29	94.11
DT CF [n]	99.47	99.87	2.17	97.83	91.91	99.99	57.14	99.09
DT REP [o]	98.04	99.45	7.68	92.32	94.81	99.98	50	82.66
DT REP [d]	99.32	99.23	0.51	99.49	96.94	99.51	57.14	94.48
DT REP [n]	99.44	99.86	2.29	97.71	93.42	99.98	46.43	98.34
NB [o]	93.05	98.96	25.30	74.70	94.03	97.59	89.29	96.86
NB [d]	95.04	96.96	2.47	97.53	98.95	90.51	82.14	87.61
NB [n]	94.79	99	17.02	82.98	93.78	97.59	89.29	96.94

B.2 Part two – ENN study

Two empirical investigations were not included in detail in Chapter 5, related to determining the number of epochs to train the MLPs with backpropagation, and the learning rate and momentum. More details on these investigations are provided in respective sections below. Thereafter, a complete listing of results for the ENN is provided in Section B.2.3.

B.2.1 Epoch investigation

The following number of epochs were examined, to cover a range encompassing what has been used previously in the literature: {5, 10, 20, 50, 100, 500, 1000, 1500, 2000, 2500, 5000}. From preliminary experiments, the following reasonable configuration parameters were used: 3 layers, 70 neurons in the hidden layer, learning rate of 0.3 and momentum constant of 0.2.

Even though the error was observed to decrease with increasing number of epochs, the differences were very small already after 5 epochs. The effects of training starts to plan out already after the fifth epoch, which can be seen in Figure B.1. The error measure quickly reaches 0.002, but does not go below this, which is illustrated in Figure B.2.

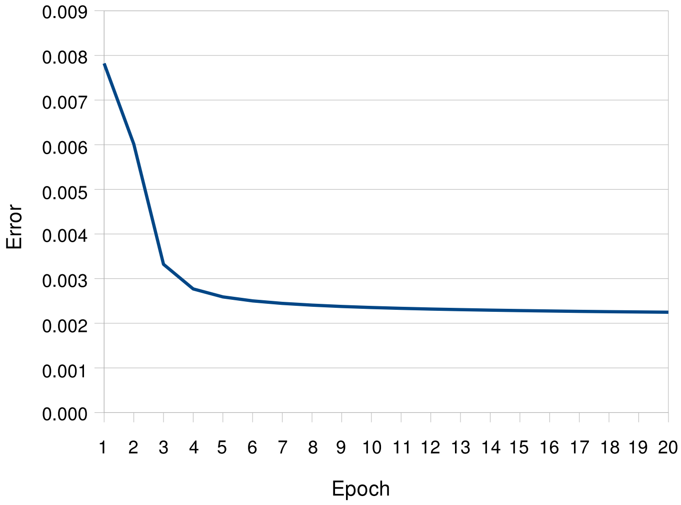


Figure B.1: Training error for the first 20 epochs.

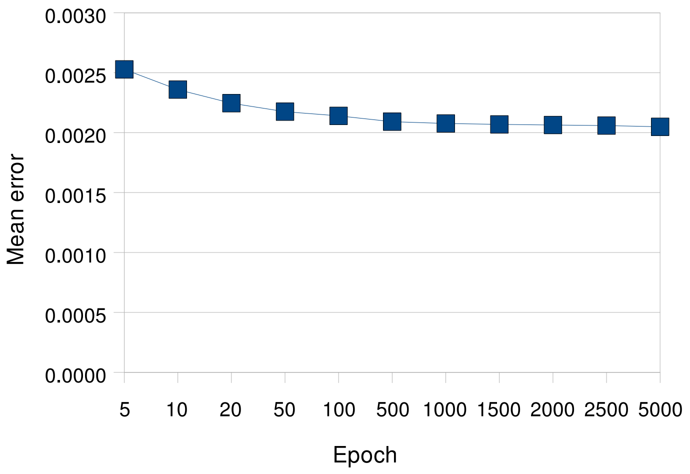


Figure B.2: Mean of final training error with increasing number of epochs.

The differences in the final error measure and classification rates, when validating the training on the training set, are not good measures alone to determine when to stop the training. At a certain point, over fitting the data is likely to happen. What can be observed, though, is that all classifications of *R2L* (on the training set) are identical from 500-5000 epochs, detecting 758. Higher classification rates were possible with short training, as low as 5, classifying 769 *R2L* instances, but then at the expense of misclassifying more *Normal* instances, as seen in Table B.18. Another observation that was made was that the classification rates became more stable after 50 epochs.

Table B.18: Confusion matrix for 5 (left) and 500 (right) epochs on the training set.

Act \ Pred	Normal	U2R	R2L
Normal	77628	0	194
U2R	19	0	22
R2L	131	0	769

Act \ Pred	Normal	U2R	R2L
Normal	77744	0	78
U2R	29	0	12
R2L	142	0	758

Observing the results on the test set, the best performance was obtained with 10 epochs. However, similarly to the observations when evaluating the performance on the training set, this performance is not as stable from run to run as the performance obtained when training longer. Table B.19 shows the confusion matrices for three trials of 10 epochs, where the number of incorrectly classified instances range from 49 (bottom left) to 59 (top left). Compared with the results obtained with 50 epochs, the number of incorrectly classified instances range from 52-53, which coincides with the observations made on the training set too.

Table B.19: Confusion matrices for four trials with 10 epochs on the test set.

Act \ Pred	Normal	U2R	R2L
Normal	19435	0	21
U2R	11	0	0
R2L	27	0	199

Act \ Pred	Normal	U2R	R2L
Normal	19443	0	13
U2R	11	0	0
R2L	29	0	197

Act \ Pred	Normal	U2R	R2L
Normal	19446	0	10
U2R	11	0	0
R2L	28	0	198

Act \ Pred	Normal	U2R	R2L
Normal	19439	0	17
U2R	11	0	0
R2L	27	0	199

Increasing the number of epochs to 100 or above appears to stabilise the performance more. The number of misclassifications never goes below 50 instances. To obtain a better sample, more trials of 100 and 200 epochs were conducted. Table B.20 includes six trials of 100 epochs, where two more trials managed to reach the same best solution as with 10 epochs. Confusion matrices obtained with 200 epochs are provided in Table B.21. The true positive rates are identical; the only difference is slight variations in false positives.

Table B.20: Confusion matrices for six trials with 100 epochs on the test set.

Act \ Pred	Normal	U2R	R2L
Normal	19445	0	11
U2R	11	0	0
R2L	28	0	198

Act \ Pred	Normal	U2R	R2L
Normal	19442	0	14
U2R	11	0	0
R2L	28	0	198

Act \ Pred	Normal	U2R	R2L
Normal	19443	0	13
U2R	11	0	0
R2L	28	0	198

Act \ Pred	Normal	U2R	R2L
Normal	19446	0	10
U2R	11	0	0
R2L	28	0	198

Act \ Pred	Normal	U2R	R2L
Normal	19444	0	12
U2R	11	0	0
R2L	28	0	198

Act \ Pred	Normal	U2R	R2L
Normal	19446	0	10
U2R	11	0	0
R2L	28	0	198

Table B.21: Confusion matrices for six trials with 200 epochs on the test set.

Act \ Pred	Normal	U2R	R2L
Normal	19445	0	11
U2R	11	0	0
R2L	28	0	198

Act \ Pred	Normal	U2R	R2L
Normal	19445	0	11
U2R	11	0	0
R2L	28	0	198

Act \ Pred	Normal	U2R	R2L
Normal	19444	0	12
U2R	11	0	0
R2L	28	0	198

Act \ Pred	Normal	U2R	R2L
Normal	19445	0	11
U2R	11	0	0
R2L	28	0	198

Act \ Pred	Normal	U2R	R2L
Normal	19444	0	12
U2R	11	0	0
R2L	28	0	198

Act \ Pred	Normal	U2R	R2L
Normal	19445	0	11
U2R	11	0	0
R2L	28	0	198

From the observations presented above, the number of epochs is set to 100. The best performance was obtained already after 10 epochs, and, even though, the performance stabilises with longer training, it does not reach the same classification rates after 100 epochs, which is potentially due to over fitting the data. Therefore, early stopping is also adopted here. 100 epochs gives enough time to conduct successful training and, if starting to over fit the data, the training process should then be stopped with early stopping. 20% of the training set is used for validation, according to the notions suggested in the context of cross validation (Haykin 1998, p. 215), and validation threshold is set to 5. That is, the training process will commence if the error on the validation set decreases over 5 epochs.

B.2.2 Investigation of learning rate and momentum

With a low learning rate (0.05–0.1) everything is classified as *Normal* unless a large momentum is used (0.5–0.9). The best classification rates are obtained with a learning rate set to 0.1, which gives similar classifier behaviour as with the learning rate set to 0.05, but detects more *R2L* intrusions (even with low momentum values). The highest TPR is obtained with a momentum of 0.2, which manages to correctly classify 221 R2L intrusions. As the momentum is increased, it is impossible to detect more than 199 *R2L* intrusions, which is also the trend as the learning rate is increased further. Figure B.3 depicts this relationship between learning rate, momentum and neurons in the hidden layer. It is observed that larger momentum values have a stabilising effect on the performance, and that the lowest FP rate is obtained with the highest learning rate and largest momentum.

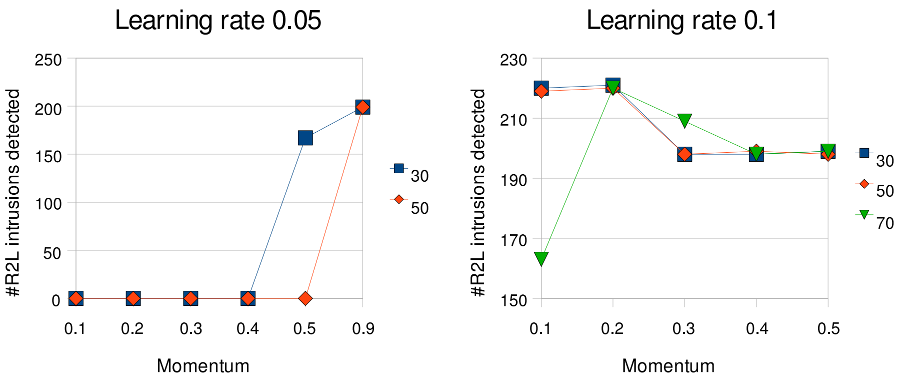


Figure B.3: Detection of R2L intrusions for learning rates 0.05 and 0.1, dependent on momentum and neurons in the hidden layer for the MLP trained with backpropagation.

Table B.22: R2L detection with learning rate set to 0.05.

neurons\momentum	0.1	0.2	0.3	0.4	0.5	0.9
30	0	0	0	0	68–167	197–199
50	0	0	0	0	0	199–199
70	0	0	0	0	0	198–199
100	0	0	0	0	0	198–199
120	0	0	0	0	0	198–199

Table B.23: R2L detection with learning rate set to 0.1.

neurons\momentum	0.1	0.2	0.3	0.4	0.5
30	219–220	198–221	198–198	198–198	197–199
50	0–219	198–220	198–198	198–199	198–198
70	0–163	196–220	198–209	198–198	198–199
100	0–196	196–220	198–218	197–198	198–198
120	0–196	197–220	197–198	198–198	197–198

Table B.24: R2L detection with learning rate set to 0.5.

neurons\momentum	0.1	0.2	0.3	0.4	0.5
30	197–198	198–198	198–199	197–199	197–199
50	197–198	198–199	197–199	197–199	197–199
70	197–198	198–199	197–198	198–199	198–199
100	198–199	196–199	196–199	197–199	198–199
120	196–199	198–199	198–199	198–198	197–199

B.2.3 Results with ENN

B.2.3.1 Three layers

Table B.25: Eval1 - three layers

HN	Accuracy%	TPR%	FPR%	Normal%	U2R%	R2L%
30	98.17	0	0.64	99.36	0	0
30	98.80	0	0	100	0	0
30	98.75	0	0.05	99.95	0	0
50	98.79	0	0.01	99.99	0	0
50	99.98	0	0.02	99.98	0	0
50	98.70	0	0.10	99.90	0	0
70	94.19	0	4.67	95.33	0	0
70	98.75	0	0.05	99.95	0	0
70	98.60	0	0.20	99.80	0	0
100	96.72	0	2.10	97.90	0	0
100	98.80	0	0	100	0	0
100	98.11	0.42	0.70	99.30	0	0
120	98.68	0	0.11	99.89	0	0
120	95.09	0.42	3.75	96.25	0	0
120	98.73	0.42	0.07	99.03	0	0

Table B.26: Eval2 - three layers.

HN	Accuracy%	TPR%	FPR%	Normal%	U2R%	R2L%
30	98.80	0	0	100	0	0
30	98.80	0	0	100	0	0
30	98.80	0	0	100	0	0
50	98.80	0	0	100	0	0
50	98.80	0	0	100	0	0
50	98.80	0	0	100	0	0
70	98.80	0	0	100	0	0
70	98.83	4.22	0.01	99.99	0	3.98

HN	Accuracy%	TPR%	FPR%	Normal%	U2R%	R2L%
70	98.82	2.11	0	100	36.36	0
100	98.80	0	0	100	0	0
100	98.80	0	0	100	0	0
100	98.79	0	0.01	99.99	0	0
120	99.91	77.64	0.09	99.91	0	81.42
120	99.77	81.86	0.01	99.99	0	85.84
120	98.80	0	0	100	0	0

Table B.27: Eval3 - three layers.

HN	Accuracy%	TPR%	FPR%	Normal%	U2R%	R2L%
30	97.75	97.47	2.24	97.76	54.55	99.12
30	98.26	97.47	1.69	98.31	0	99.12
30	98.99	84.81	0.82	99.18	0	88.05
50	99.10	84.81	0.71	99.29	0	88.05
50	97.87	99.16	2.10	97.90	0	100
50	98.70	94.94	1.27	98.73	0	98.67
70	97.71	96.62	2.26	97.74	0	99.56
70	98.20	92.41	1.70	98.30	18.18	93.36
70	98.70	94.51	1.24	98.76	0	98.23
100	98.73	98.73	1.24	98.76	63.64	98.23
100	97.71	95.78	2.26	97.74	0	99.56
100	97.97	95.36	1.98	98.02	0	98.67
120	98.56	96.20	1.37	98.63	0	97.35
120	97.71	95.78	2.17	97.83	0	98.67
120	97.48	97.89	2.49	97.51	0	99.56

Table B.28: Eval4 - three layers.

HN	Accuracy%	TPR%	FPR%	Normal%	U2R%	R2L%
30	79.04	96.20	21.17	78.83	54.55	97.79
30	97.91	95.78	2.05	97.95	0	99.12
30	97.81	98.73	2.16	97.84	0	99.56
50	97.12	98.73	2.85	97.15	0	99.12
50	98.20	97.89	1.76	98.24	9.09	99.12
50	97.07	97.89	2.90	97.10	0	99.12
70	97.69	97.89	2.30	97.70	81.82	98.23
70	99.11	97.05	0.83	99.17	0	98.67
70	98.49	94.94	1.45	98.55	0	98.67
100	98.67	97.47	1.40	98.60	54.55	97.79
100	97.35	99.16	2.65	97.35	63.64	99.12

HN	Accuracy%	TPR%	FPR%	Normal%	U2R%	R2L%
100	98.18	96.20	1.76	98.24	0	97.79
120	98.40	96.20	1.56	98.44	27.27	98.23
120	99.52	83.97	0.29	99.71	9.09	87.61
120	98.31	95.36	1.63	98.37	0	98.23

Table B.29: Eval5 - three layers.

HN	Accuracy%	TPR%	FPR%	Normal%	U2R%	R2L%
30	76.80	88.61	23.34	76.66	90.91	88.05
30	73.38	96.62	26.73	73.27	63.64	83.19
30	77.51	94.51	22.66	77.34	9.09	95.13
50	72.06	98.31	28.07	71.78	81.82	95.58
50	98.92	96.62	1.02	98.98	0	98.67
50	70.22	97.89	30.04	69.96	81.82	92.04
70	96.69	97.05	3.23	96.77	81.82	90.71
70	75.43	97.89	24.70	75.30	81.82	86.73
70	98.21	98.31	1.76	98.24	72.73	97.35
100	77.25	98.73	23	77	81.82	96.76
100	76.45	96.20	22.76	77.24	36.36	10.62
100	76.94	98.73	23.15	76.85	81.82	84.96
120	93.70	98.73	6.19	93.81	81.82	84.96
120	97.33	97.89	2.67	97.33	72.73	98.23
120	98.43	85.65	0.41	99.59	72.73	0

B.2.3.2 Four layers

Table B.30: Eval1 - four layers.

HN	Accuracy%	TPR%	FPR%	Normal%	U2R%	R2L%
5-5	98.80	0	0	100	0	0
5-5	98.80	0	0	100	0	0
5-5	98.80	0	0	100	0	0
10-10	98.80	0	0	100	0	0
10-10	98.80	0	0	100	0	0
10-10	98.80	0	0	100	0	0
15-15	98.80	0	0	100	0	0
15-15	98.79	0.42	0.01	99.99	0	0
15-15	98.80	0	0	100	0	0
20-20	98.66	0.42	0.13	99.87	0	0
20-20	98.80	0	0	100	0	0
20-20	98.80	0	0	100	0	0
25-25	98.80	0	0	100	0	0

HN	Accuracy%	TPR%	FPR%	Normal%	U2R%	R2L%
25-25	98.72	11.81	0.10	99.90	36.36	0
25-25	98.77	0	0.03	99.97	0	0

Table B.31: Eval2 - four layers.

HN	Accuracy%	TPR%	FPR%	Normal%	U2R%	R2L%
5-5	98.80	0	0	100	0	0
5-5	98.80	0	0	100	0	0
5-5	98.80	0	0	100	0	0
10-10	98.80	0	0	100	0	0
10-10	98.80	0	0	100	0	0
10-10	98.80	0	0	100	0	0
15-15	98.80	0	0	100	0	0
15-15	98.80	0	0	100	0	0
15-15	98.78	0	0.02	99.98	0	0
20-20	98.79	0	0.01	99.99	0	0
20-20	98.80	0	0	100	0	0
20-20	98.80	0	0	100	0	0
25-25	98.80	0	0	100	0	0
25-25	98.80	0	0	100	0	0
25-25	98.80	0	0	100	0	0

Table B.32: Eval3 - four layers.

HN	Accuracy%	TPR%	FPR%	Normal%	U2R%	R2L%
5-5	92.26	96.20	7.75	92.25	0	98.23
5-5	97.64	97.89	2.32	97.68	0	98.67
5-5	96.11	94.09	3.87	96.13	0	98.67
10-10	97.65	84.81	2.15	97.85	0	85.40
10-10	98.58	98.31	1.40	98.60	63.64	98.23
10-10	97.96	84.81	1.88	98.12	0	88.94
15-15	98.50	96.62	1.44	98.56	0	97.79
15-15	97.71	94.94	2.25	97.75	0	98.67
15-15	98.38	94.09	1.57	98.43	0	98.23
20-20	98.52	97.89	1.43	98.57	0	99.12
20-20	98.12	98.89	1.87	98.13	63.64	98.23
20-20	98.72	96.20	1.22	98.78	0	98.67
25-25	77.86	98.31	22.34	77.66	0	98.67
25-25	97.46	94.51	2.50	97.50	72.73	95.13
25-25	98.35	94.94	1.59	98.41	9.09	98.23

Table B.33: Eval4 - four layers.

HN	Accuracy%	TPR%	FPR%	Normal%	U2R%	R2L%
5-5	97.44	95.36	2.53	97.47	0	99.56
5-5	97.94	88.61	1.92	98.08	0	90.71
5-5	97.95	97.89	2	98	0	98.67
10-10	97.98	94.51	1.97	98.03	36.36	96.90
10-10	96.28	95.36	3.70	96.30	0	99.12
10-10	97.72	96.62	2.25	97.75	0	99.56
15-15	97.57	98.73	2.40	97.60	0	100
15-15	99.10	95.78	0.84	99.16	0	98.67
15-15	97.81	94.51	2.14	97.86	0	98.23
20-20	97.52	91.98	2.40	97.60	0	95.13
20-20	98.16	97.89	1.80	98.20	0	99.56
20-20	96.83	98.31	3.18	96.82	81.82	98.23
25-25	98.58	88.19	1.24	98.76	0	88.05
25-25	97.31	97.89	2.66	97.34	9.09	99.12
25-25	98.11	84.81	1.72	98.28	0	87.61

Table B.34: Eval5 - four layers.

HN	Accuracy%	TPR%	FPR%	Normal%	U2R%	R2L%
5-5	83.81	86.50	15.21	84.79	72.73	0
5-5	77.68	85.23	21.41	78.59	63.64	0
5-5	95.31	97.47	4.66	95.34	81.82	93.36
10-10	75.87	97.47	24.38	75.62	72.73	97.79
10-10	73.16	99.16	27.07	72.93	81.82	92.04
10-10	98.06	98.31	1.90	98.10	0	99.12
15-15	97.65	89.03	2.23	97.77	72.73	88.50
15-15	98.53	97.05	1.32	98.68	63.64	87.61
15-15	97.82	83.97	1.01	98.99	27.27	0
20-20	77.23	87.76	22.84	77.16	72.73	83.19
20-20	97.32	96.62	2.50	97.50	54.55	84.07
20-20	92.91	98.31	7.08	92.92	81.82	92.48
25-25	97.68	97.89	2.28	97.72	81.82	95.13
25-25	96	96.20	3.95	96.05	63.64	92.92
25-25	97.84	97.47	2.14	97.86	72.73	97.79

B.3 Part three – MABLE study

Additional results for the two-class investigation are given below in Section B.3.1, and for the three-class investigation in Section B.3.2.

B.3.1 Two-class investigation

The additional results listed here are summaries of ensemble combinations from MABLE, and other combination approaches based on threshold filtering.

B.3.1.1 Summaries of ensemble information

Table B.35: Summary of variable sized ensemble data.

ID	Normal%	U2R%	TPR%	FPR%	Size	Dom
#1	98.21	92.55	92.55	1.79	8	4
#2	96.51	97.87	97.87	3.49	3	3
#3	96.97	96.81	96.81	3.03	3	2
#4	99.95	64.89	64.89	0.05	17	3
#5	100	45.74	45.74	0	5	1
#6	97.61	95.74	95.74	2.39	10	4
#7	99.96	62.77	62.77	0.04	17	3
#8	98.92	88.3	88.3	1.08	8	5
#9	99.95	63.83	63.83	0.05	15	3
#10	99.6	78.72	78.72	0.4	12	3
#11	99.94	67.02	67.02	0.06	13	4
#12	99.9	70.21	70.21	0.1	14	3
#13	99.85	72.34	72.34	0.15	8	4
#14	99.93	68.09	68.09	0.07	15	4
#15	99.55	79.79	79.79	0.45	15	2
#16	99.1	87.23	87.23	0.9	14	4
#17	99.99	54.26	54.26	0.01	11	4
#18	98.72	91.49	91.49	1.28	10	5
#19	99.97	60.64	60.64	0.03	10	2
#20	99.98	59.57	59.57	0.02	12	4
#21	99.98	57.45	57.45	0.02	15	5
#22	99.94	65.96	65.96	0.06	12	4
#23	99.88	71.28	71.28	0.12	11	4
#24	99.54	80.85	80.85	0.46	12	3
#25	99.99	51.06	51.06	0.01	10	5
#26	99.52	81.91	81.91	0.48	13	3
#27	99.5	82.98	82.98	0.5	17	3
#28	99.7	76.6	76.6	0.3	14	4
#29	99.23	86.17	86.17	0.77	16	5
#30	99.8	74.47	74.47	0.2	14	4
#31	99.99	53.19	53.19	0.01	11	6
#32	99.91	69.15	69.15	0.09	14	3
#33	99.36	85.11	85.11	0.64	17	4
#34	97.72	93.62	93.62	2.28	11	4

ID	Normal%	U2R%	TPR%	FPR%	Size	Dom
#35	99.98	56.38	56.38	0.02	11	5
#36	99.68	77.66	77.66	0.32	14	4
#37	99.96	61.7	61.7	0.04	10	2
#38	99.41	84.04	84.04	0.59	19	3

Table B.36: Summary of fixed sized ensemble data; size 5.

ID	Normal%	U2R%	TPR%	FPR%	Size	Dom
#1	99.99	55.32	55.32	0.01	5	6
#2	97.34	97.87	97.87	2.66	5	5
#3	98.94	88.30	88.30	1.06	5	5
#4	99.89	71.28	71.28	0.11	5	4
#5	99.96	63.83	63.83	0.04	5	3
#6	99.95	64.89	64.89	0.05	5	3
#7	99.99	47.87	47.87	0	5	4
#8	99.44	81.92	81.92	0.56	5	3
#9	97.84	96.81	96.81	2.16	5	6
#10	99.94	65.96	65.96	0.06	5	4
#11	99.99	57.45	57.45	0.01	5	5
#12	99.50	78.72	78.72	0.50	5	2
#13	98.04	93.62	93.62	1.97	5	4
#14	98.24	92.55	92.55	1.76	5	4
#15	99.90	70.22	70.21	0.10	5	3
#16	99.45	80.85	80.85	0.55	5	3
#17	99.38	82.98	82.98	0.62	5	3
#18	99.98	58.51	58.51	0.02	5	5
#19	99.50	79.79	79.79	0.50	5	2
#20	99.91	69.15	69.15	0.09	5	3
#21	99.96	60.64	60.64	0.04	5	2
#22	99.99	54.26	54.26	0.01	5	6
#23	99.79	74.47	74.47	0.22	5	4
#24	99.63	75.53	75.53	0.37	5	2
#25	97.92	95.75	95.74	2.08	5	5
#26	98.29	91.49	91.49	1.71	5	3
#27	99.79	72.34	72.34	0.21	5	4
#28	98.76	89.36	89.36	1.24	5	4
#29	99.92	68.09	68.09	0.08	5	4
#30	99.32	84.04	84.04	0.68	5	2
#31	99.15	86.17	86.17	0.86	5	4
#32	98.38	90.43	90.43	1.62	5	4

ID	Normal%	U2R%	TPR%	FPR%	Size	Dom
#33	99.52	77.66	77.66	0.48	5	2
#34	99.56	76.60	76.60	0.44	5	2
#35	99.03	87.23	87.23	0.97	5	4
#36	99.96	62.77	62.77	0.04	5	3
#37	99.30	85.11	85.11	0.70	5	3

Table B.37: Summary of fixed sized ensemble data; size 10.

ID	Normal%	U2R%	TPR%	FPR%	Size	Dom
#1	98.13	91.49	91.49	1.87	10	3
#2	97.79	93.62	93.62	2.21	10	4
#3	98.6	89.36	89.36	1.4	10	4
#4	99.92	69.15	69.15	0.08	10	4
#5	99.96	63.83	63.83	0.04	10	3
#6	99.42	82.98	82.98	0.58	10	3
#7	99.63	77.66	77.66	0.37	10	3
#8	99.04	87.23	87.23	0.96	10	4
#9	99.84	73.4	73.4	0.16	10	4
#10	99.97	62.77	62.77	0.03	10	3
#11	99.97	59.57	59.57	0.03	10	2
#12	99.95	64.89	64.89	0.05	10	3
#13	99.35	84.04	84.04	0.65	10	3
#14	99.98	56.38	56.38	0.02	10	5
#15	99.98	57.45	57.45	0.02	10	5
#16	98.84	88.3	88.3	1.16	10	4
#17	98.36	90.43	90.43	1.64	10	4
#18	99.99	52.13	52.13	0.01	10	5
#19	99.56	80.85	80.85	0.44	10	3
#20	99.88	71.28	71.28	0.12	10	4
#21	99.93	68.09	68.09	0.07	10	4
#22	99.89	70.21	70.21	0.11	10	2
#23	99.87	72.34	72.34	0.13	10	4
#24	99.73	76.6	76.6	0.27	10	5
#25	99.99	55.32	55.32	0.01	10	6
#26	99.48	81.91	81.91	0.52	10	3
#27	99.74	75.53	75.53	0.26	10	4
#28	99.95	65.96	65.96	0.05	10	4
#29	99.93	67.02	67.02	0.07	10	4
#30	99.97	58.51	58.51	0.03	10	2
#31	99.27	86.17	86.17	0.74	10	5

ID	Normal%	U2R%	TPR%	FPR%	Size	Dom
#32	99.75	74.47	74.47	0.25	10	3

Table B.38: Summary of fixed sized ensemble data; size 20.

ID	Normal%	U2R%	TPR%	FPR%	Size	Dom
#1	99.44	84.04	84.04	0.56	20	3
#2	98.95	87.23	87.23	1.05	20	4
#3	99.5	82.98	82.98	0.5	20	3
#4	99.64	77.66	77.66	0.36	20	3
#5	99.86	73.4	73.4	0.14	20	4
#6	99.93	68.09	68.09	0.07	20	4
#7	99.89	70.21	70.21	0.11	20	2
#8	99.77	75.53	75.53	0.23	20	4
#9	99.24	85.11	85.11	0.76	20	3
#10	99.95	62.77	62.77	0.05	20	2
#11	99.96	61.7	61.7	0.04	20	2
#12	99.2	86.17	86.17	0.8	20	4
#13	99.91	69.15	69.15	0.09	20	3
#14	99.94	64.89	64.89	0.06	20	3
#15	99.57	79.79	79.79	0.43	20	2
#16	99.62	78.72	78.72	0.38	20	3
#17	99.96	60.64	60.64	0.04	20	2
#18	99.88	72.34	72.34	0.12	20	4
#19	99.74	76.6	76.6	0.26	20	5
#20	99.8	74.47	74.47	0.2	20	4
#21	99.95	63.83	63.83	0.05	20	2
#22	99.94	65.96	65.96	0.06	20	4

B.3.1.2 Other combination approaches

Table B.39: Ensemble performance after filtering out classifiers according to classification thresholds for both classes.

Threshold%	Normal%	U2R%	TPR%	FPR%	Size
40	99.73	73.40	73.40	0.27	31
50	99.05	85.11	85.11	0.95	24

Table B.40: Ensemble performance after filtering out classifiers according to classification thresholds for the normal class only.

Threshold%	Normal%	U2R%	TPR%	FPR%	Size
95	99.84	67.02	67.02	0.16	30
97	99.91	64.89	64.89	0.094	27

B.3.1.3 Generalisation investigation: U2R attacks

Table B.41: Base classifiers (50HN MLPs) - the number of instances correctly classified during training.

	buffer overflow	loadmodule	perl	ps	rootkit	sqlattack	xterm
#1	37	8	4	7	15	1	10
#2	35	5	4	2	5	1	7
#3	41	8	4	11	18	1	10
#4	37	6	4	9	14	1	10
#5	37	8	4	9	16	1	10
#6	40	8	4	9	17	1	10
#7	27	3	4	1	5	1	4
#8	41	8	4	8	15	1	9
#9	36	8	4	7	14	1	9
#10	35	8	4	4	5	1	9
#11	36	7	4	9	16	1	10
#12	37	7	4	5	9	1	9
#13	41	8	4	12	18	1	10
#14	36	8	4	5	5	1	8
#15	27	4	3	2	3	1	7
#16	37	6	4	5	14	1	9
#17	35	5	4	3	5	1	9
#18	40	8	4	8	17	1	10
#19	26	5	4	2	4	1	6
#20	23	3	4	1	4	1	5
#21	29	5	4	2	4	1	5
#22	24	5	4	1	4	1	7
#23	36	5	4	3	5	1	7
#24	32	3	4	2	5	1	6
#25	41	8	4	11	17	1	10
#26	36	7	4	5	8	1	9
#27	19	2	4	0	4	1	5
#28	37	6	4	7	15	1	10
#29	21	3	4	1	4	1	5
#30	22	3	4	1	4	1	7

	buffer overflow	loadmodule	perl	ps	rootkit	sqlattack	xterm
#31	26	3	4	1	4	1	5
#32	41	8	4	10	16	1	10
Mean	33.38	5.91	3.97	5.09	9.66	1.00	8.03
Median	36	6	4	5	6.5	1	9
Mode	37	8	4	1	4	1	10
Total	41	8	4	12	18	1	10

Table B.42: Base classifiers (50HN MLPs) - the number of instances correctly classified during testing.

	buffer overflow	loadmodule	perl	ps	rootkit	sqlattack	xterm
#1	8	3	1	3	3	1	3
#2	8	3	0	1	0	1	2
#3	8	3	1	3	5	1	3
#4	8	3	0	3	5	1	3
#5	8	3	1	3	2	1	3
#6	8	3	0	3	5	1	3
#7	7	2	0	1	0	1	1
#8	8	3	1	3	0	1	3
#9	8	3	0	1	2	1	3
#10	8	2	1	1	0	1	3
#11	8	3	0	2	5	1	3
#12	8	3	0	2	0	1	3
#13	8	3	1	3	5	1	3
#14	8	3	0	1	0	1	3
#15	6	1	0	1	0	1	3
#16	8	3	0	2	2	1	3
#17	7	3	0	1	0	1	3
#18	8	3	0	2	5	1	3
#19	7	3	0	1	0	1	2
#20	7	3	0	1	0	1	1
#21	8	2	0	0	0	1	1
#22	7	2	0	0	0	1	2
#23	7	3	0	1	0	1	2
#24	8	3	0	1	0	1	1
#25	8	3	1	3	5	1	3
#26	8	3	1	2	0	1	3
#27	7	3	0	0	0	1	0
#28	8	3	0	1	2	1	3

	buffer overflow	loadmodule	perl	ps	rootkit	sqlattack	xterm
#29	7	2	0	0	0	1	0
#30	6	1	0	0	0	1	2
#31	6	1	0	1	0	1	2
#32	9	3	1	3	2	1	3
Mean	7.59	2.66	0.28	1.56	1.50	1.00	2.38
Median	8	3	0	1	0	1	3
Mode	8	3	0	1	0	1	3
Total	11	3	1	4	5	1	3

Table B.43: Ensembles (size 5) - the number of instances correctly classified during training.

	buffer overflow	loadmodule	perl	ps	rootkit	sqlattack	xterm
#1	29	3	4	3	5	1	7
#2	41	8	4	11	17	1	10
#3	37	7	4	8	16	1	10
#4	36	8	4	5	5	1	8
#5	35	5	4	3	5	1	7
#6	36	5	4	3	5	1	7
#7	23	3	4	2	5	1	7
#8	36	8	4	5	14	1	9
#9	41	8	4	10	17	1	10
#10	36	5	4	3	5	1	8
#11	30	4	4	3	5	1	7
#12	35	8	4	4	13	1	9
#13	41	8	4	8	16	1	10
#14	41	8	4	7	16	1	10
#15	35	8	4	5	5	1	8
#16	35	8	4	5	14	1	9
#17	36	8	4	6	14	1	9
#18	30	4	4	3	5	1	8
#19	36	8	4	4	13	1	9
#20	35	7	4	5	5	1	8
#21	32	5	4	3	5	1	7
#22	27	4	4	3	5	1	7
#23	36	8	4	5	7	1	9
#24	36	7	4	5	8	1	10
#25	41	8	4	9	17	1	10
#26	41	8	4	7	15	1	10

	buffer overflow	loadmodule	perl	ps	rootkit	sqlattack	xterm
#27	36	8	4	5	6	1	8
#28	37	8	4	9	15	1	10
#29	35	6	4	5	5	1	8
#30	35	8	4	6	15	1	10
#31	37	8	4	6	15	1	10
#32	37	8	4	9	16	1	10
#33	37	8	4	5	9	1	9
#34	36	8	4	5	9	1	9
#35	37	8	4	7	15	1	10
#36	34	5	4	3	5	1	7
#37	36	8	4	6	15	1	10
Mean	35.51	6.86	4.00	5.43	10.32	1.00	8.72
Median	36	8	4	5	9	1	9
Mode	36	8	4	5	5	1	10
Total	41	8	4	12	18	1	10

Table B.44: Ensembles (size 5) - the number of instances correctly classified during testing.

	buffer overflow	loadmodule	perl	ps	rootkit	sqlattack	xterm
#1	7	3	0	1	0	1	2
#2	8	3	1	3	5	1	3
#3	8	3	0	2	5	1	3
#4	8	3	0	1	0	1	3
#5	7	3	0	1	0	1	2
#6	7	3	0	1	0	1	2
#7	7	2	0	0	0	1	2
#8	8	3	0	1	2	1	3
#9	8	3	0	3	0	1	3
#10	8	3	0	1	0	1	3
#11	7	3	0	1	0	1	2
#12	8	3	0	1	2	1	3
#13	8	3	0	3	2	1	3
#14	8	3	0	2	2	1	3
#15	8	3	0	1	0	1	3
#16	8	3	0	1	2	1	3
#17	8	3	0	1	2	1	3
#18	7	3	0	1	0	1	3
#19	8	3	0	1	2	1	3

	buffer overflow	loadmodule	perl	ps	rootkit	sqlattack	xterm
#20	8	3	0	1	0	1	3
#21	7	3	0	1	0	1	2
#22	7	3	0	1	0	1	2
#23	8	3	0	1	0	1	3
#24	8	3	0	1	0	1	3
#25	8	3	0	3	5	1	3
#26	8	3	0	3	0	1	3
#27	8	3	0	1	0	1	3
#28	8	3	0	2	5	1	3
#29	8	3	0	1	0	1	3
#30	8	3	0	1	2	1	3
#31	8	3	0	1	2	1	3
#32	8	3	0	2	5	1	3
#33	8	3	0	1	0	1	3
#34	8	3	0	1	0	1	3
#35	8	3	0	1	2	1	3
#36	7	3	0	1	0	1	2
#37	8	3	0	1	2	1	3
Mean	7.76	2.97	0.03	1.35	1.27	1.00	2.78
Median	8	3	0	1	0	1	3
Mode	8	3	0	1	0	1	3
Total	11	3	1	4	5	1	3

B.3.2 Three-class investigation

These additional results listings are for combinations of base classifiers based on clustering and thresholds.

Table B.45: Ensembles based on clustering of the entire archive (70HN).

k	Normal%	U2R%	R2L%	TPR%	FPR%
5	98.92	37.14	11.87	14.61	1.08
10	98.49	31.43	29.86	31.68	1.51
20	99.17	44.29	32.28	36	0.83
30	99.10	44.29	29.33	32.10	0.90
40	99.31	37.14	28.88	30.37	0.69
50	99.37	41.43	25.67	27.33	0.63
all (347)	99.27	38.57	31.43	33.22	0.73

Table B.46: Ensembles based on clustering of the entire archive (70HN) - including boundary solutions.

Size	Normal%	U2R%	R2L%	TPR%	FPR%
11	99.41	37.14	14.29	15.79	0.59
16	99.33	34.29	27.37	28.48	0.67
26	99.23	40.00	32.31	34.72	0.77
35	99.26	41.43	29.56	30.82	0.74
46	99.36	37.14	28.78	29.89	0.64
54	99.41	41.43	25.08	26.34	0.59
all (347)	99.27	38.57	31.43	33.22	0.73

Table B.47: Ensembles based on clustering after filtering out classifiers according to a 0% threshold.

k	Normal%	U2R%	R2L%	TPR%	FPR%
5	98.99	40.00	20.93	24.23	1.01
10	99.25	40.00	21.68	23.95	0.75
20	98.98	44.29	25.80	29.60	1.02
30	99.30	40.00	30.22	32.03	0.70
40	99.17	42.86	33.52	36.09	0.83
50	99.34	35.71	33.19	33.98	0.66
all (333)	99.19	41.43	31.82	34.18	0.81

Table B.48: Ensembles based on clustering after filtering out classifiers according to a 10% threshold.

k	Normal%	U2R%	R2L%	TPR%	FPR%
5	98.40	44.29	33.22	36.35	1.60
10	98.58	42.86	34.17	36.60	1.42
20	97.96	44.29	32.70	36.86	2.04
30	98.81	38.57	38.23	40.76	1.19
40	98.84	47.14	33.22	38.46	1.16
50	98.70	47.14	37.80	42.04	1.30
all (273)	98.83	44.29	35.19	38.87	1.17

Table B.49: Ensembles based on clustering after filtering out classifiers according to a 20% threshold.

k	Normal%	U2R%	R2L%	TPR%	FPR%
5	98.88	38.57	35.12	37.95	1.12
10	98.91	47.14	32.50	36.13	1.09
20	98.23	45.71	34.17	38.62	1.77
30	98.47	42.86	37.84	43.41	1.53
40	98.47	45.71	38.13	43.89	1.53
50	98.66	44.29	37.25	41.88	1.34
all (247)	98.72	47.14	36.07	40.63	1.28

Table B.50: Ensembles based on clustering after filtering out classifiers according to a 30% threshold.

k	Normal%	U2R%	R2L%	TPR%	FPR%
5	97.50	34.29	37.44	41.46	2.50
10	98.46	48.57	34.17	38.46	1.54
20	98.57	44.29	37.48	41.98	1.43
30	98.18	50.00	35.68	42.26	1.82
40	98.50	45.71	37.08	42.62	1.50
50	98.74	48.57	36.27	40.70	1.26
all (223)	98.63	48.57	36.53	41.53	1.37

Table B.51: Ensembles based on clustering after filtering out classifiers according to a 40% threshold.

k	Normal%	U2R%	R2L%	TPR%	FPR%
5	96.62	50.00	37.34	43.09	3.38
10	97.07	55.71	37.90	45.33	2.93
20	98.20	50.00	35.64	41.56	1.80
30	98.10	48.57	37.21	43.35	1.90
40	98.02	48.57	37.44	42.58	1.98
50	98.28	47.14	36.59	41.69	1.72
all (190)	98.46	48.57	36.82	42.04	1.54

Table B.52: Ensembles based on clustering after filtering out classifiers according to a 50% threshold.

k	Normal%	U2R%	R2L%	TPR%	FPR%
5	98.19	47.14	36.36	42.68	1.81
10	97.46	45.71	36.89	42.07	2.54
20	98.05	50.00	38.39	44.82	1.95
30	97.80	45.71	37.38	43.41	2.20
40	97.78	48.57	37.28	44.41	2.22
50	97.83	50.00	38.52	44.66	2.17
all (155)	98.20	48.57	38.16	44.15	1.80

Table B.53: Ensembles based on clustering after filtering out classifiers according to a threshold of 50% for the normal class only.

k	Normal%	U2R%	R2L%	TPR%	FPR%
5	99.21	27.14	15.99	17.10	0.79
10	99.43	38.57	29.10	30.08	0.57
20	99.11	40.00	32.05	35.36	0.89
30	99.16	37.14	27.40	30.47	0.84
40	99.27	44.29	28.29	30.53	0.73
50	99.27	37.14	30.80	32.32	0.73
all (328)	99.37	37.14	29.43	30.50	0.63

Table B.54: Ensembles based on clustering after filtering out classifiers according to a threshold of 60% for the normal class only.

k	Normal%	U2R%	R2L%	TPR%	FPR%
5	99.68	31.43	15.14	18.45	0.32
10	99.64	28.57	20.50	21.58	0.36
20	99.41	35.71	29.53	31.11	0.59
30	99.12	41.43	31.13	33.31	0.88
40	99.43	37.14	26.85	27.88	0.57
50	99.36	38.57	29.01	30.66	0.64
all (326)	99.39	37.14	29.27	30.34	0.61

Table B.55: Ensembles based on clustering after filtering out classifiers according to a threshold of 70% for the normal class only.

k	Normal%	U2R%	R2L%	TPR%	FPR%
5	99.6886	32.8571	17.397	18.5102	0.311435
10	98.9152	35.7143	26.3571	30.3708	1.0848
20	99.3437	38.5714	20.0131	22.3785	0.656313
30	99.4628	34.2857	28.2538	29.3798	0.537174
40	99.3918	41.4286	23.2505	26.2788	0.608239
50	99.3897	37.1429	27.6324	29.6355	0.61033
all (317)	99.4377	37.1429	27.5343	28.5806	0.562256

Table B.56: Ensembles based on clustering after filtering out classifiers according to a threshold of 80% for the normal class only.

k	Normal%	U2R%	R2L%	TPR%	FPR%
5	99.69	32.86	17.50	18.35	0.31
10	99.38	34.29	25.83	27.01	0.62
20	99.75	34.29	10.96	12.12	0.25
30	99.70	34.29	15.08	16.15	0.30
40	99.45	35.71	29.50	30.40	0.55
50	99.34	37.14	26.91	29.12	0.67
all (300)	99.51	35.71	23.97	24.94	0.49

Table B.57: Ensembles based on clustering after filtering out classifiers according to a threshold of 90% for the normal class only.

k	Normal%	U2R%	R2L%	TPR%	FPR%
5	99.84	34.29	11.90	13.24	0.16
10	99.80	30	17.04	17.84	0.20
20	99.77	31.43	12.10	13.20	0.23
30	99.86	30	9.88	10.84	0.14
40	99.83	34.29	8.21	9.37	0.17
50	99.81	32.86	12.72	13.75	0.19
all (243)	99.80	34.29	12	13.08	0.20

Table B.58: Ensembles based on clustering after filtering out classifiers according to a threshold of 95% for the normal class only.

k	Normal%	U2R%	R2L%	TPR%	FPR%
5	99.90	20.00	6.93	7.67	0.10
10	99.91	24.29	4.38	5.31	0.09
20	99.92	30.00	1.83	2.81	0.08
30	99.90	30.00	1.34	2.37	0.10
40	99.90	28.57	4.02	5.05	0.10
50	99.91	31.43	0.62	1.69	0.09
all (190)	99.91	30.00	1.83	2.85	0.09

Table B.59: Ensembles based on clustering after filtering out classifiers according to a threshold of 97% for the normal class only.

k	Normal%	U2R%	R2L%	TPR%	FPR%
5	99.93	30.00	1.05	2.01	0.07
10	99.92	18.57	4.15	4.86	0.08
20	99.93	22.86	0.36	1.15	0.07
30	99.94	27.14	0.33	1.15	0.06
40	99.94	28.57	0.52	1.37	0.06
50	99.92	25.71	0.59	1.34	0.08
all (162)	99.93	28.57	0.49	1.37	0.07