To The Knowledge Frontier and Beyond:

A Hybrid System for Incremental Contextual-Learning and Prudence Analysis

by

Richard Peter Dazeley (BComp Hons)

A Dissertation submitted to the School of Computing in fulfilment of the requirements for the Degree of

Doctor of Philosophy

University of Tasmania
(March, 2006)

Declaration

This thesis contains no material, which has been accepted for the award of any other degree or diploma in any tertiary institution, and that, to my knowledge and belief, this thesis contains no material previously published or written by another person except where due reference is made in the text of the thesis

Signed_____

This thesis may be made available for loan and limited copying in accordance with the *Copyright Act 1968*

Signed _____ Date_____

Abstract

Increasingly, researchers and developers of knowledge based systems (KBS) have been attempting to incorporate the notion of context. For instance, Repertory Grids, Formal Concept Analysis (FCA) and Ripple-Down Rules (RDR) all integrate either implicit or explicit contextual information. However, these methodologies treat context as a static entity, neglecting many connectionists' work in learning *hidden and dynamic contexts*. This thesis argues that the omission of these higher forms of context, which allow connectionist systems to generalise effectively, is one of the fundamental problems in the application and interpretation of symbolic knowledge.

This thesis tackles the problems of KBSs by addressing these contextual inadequacies over a three stage approach: *philosophically*, *methodologically* and through the application of *prudence analysis*. Firstly, it challenges existing notions of knowledge by introducing a new philosophical view referred to as *Intermediate Situation Cognition*. This new position builds on the existing SC premise, that knowledge and memory is re-constructed at the moment required, by allowing for the inclusion of hidden and dynamic contexts in symbolic reasoning.

This philosophical position has been incorporated into the development of a hybridised methodology, combining Multiple Classification Ripple-Down Rules (MCRDR) with a function-fitting technique. This approach, referred to as Rated MCRDR (RM), retains a symbolic core acting as a contextually static memory, while using a connection based approach to learn a deeper understanding of the knowledge captured. This analysis of the knowledge map is performed dynamically, providing constant online information. Results indicate that the method developed can learn the information that experts have difficulty providing. This supplies the information required to allow for generalisation of the knowledge captured.

In order to show that hidden and dynamic contextual information can improve the robustness of a KBS, RM must reduce brittleness. Brittleness, which is widely recognised as the primary impediment in KBS performance, is caused by a system's inability to realise when its knowledge base is inadequate for a particular situation. RM partly addresses this through providing better generalisation; however, brittleness can be more directly addressed by detecting when such inadequacies occur. This process is commonly referred to as *prudence analysis*. The final part of this thesis proves the methods philosophical and methodological approach by illustrating how RM's use of hidden and dynamic contextual information, allows the system to perform this analysis. Results show how experts can confidently leave the verification of cases when not warned, reducing brittleness and the knowledge acquisition effort.

This thesis shows that the idea of incorporating higher forms of context in symbolic reasoning domains is both possible and highly effective, vastly improving the robustness of the KBS approach. Not only does this facilitate improved classification through better generalisation, but also reduces the KA effort required by experts. Additionally, the methodology developed has further potential for many possible applications across numerous domains, such as Information Filtering, Data Mining, incremental induction and even reinforcement learning.

Acknowledgements

The long, sometimes exciting, occasionally dull, fulfilling road of developing a large solid body of research is at completion. To have the privilege of studying for a PhD is an honour, few find themselves, and even fewer find themselves at the other end holding a completed thesis dissertation. To have reached this stage a student must have strong and dedicated support from those around them and this is my opportunity to say thank you to those that put up with me over these past three years. This was especially a major challenge for them over the last six months when it has been impossible to hold a conversation with me for more than a minute before I would turn it to my PhD progress. This is one personality trait I am hoping to lose soon.

I would very much like to thank my supervisor Dr Byeong Ho-Kang. Byeong made himself available whenever I knocked, offering encouragement and insight. Byeong was a great supervisor as he was able to aid me most in those areas I needed the most help while giving me the freedom to roam when required. I would also like to thank my associate supervisors Dr Ray Williams and Professor Young Choi for their help, support and encouragement when required.

Three years is a long time to go without food, so I'd very much like to thank the Smart Internet Technology Co-operative Research Centre (SITCRC) for their very generous financial support. The SITCRC provided a regular income allowing my family and I to live, secure in the knowledge that we were provided for, while I embarked on my mad adventures. I would also like to thank the SITCRC for their non-financial support, providing research collaboration seminars, conferences and training sessions on everything from business management to ways for dealing with the press.

Over the last stage of producing this final thesis I would like to thank my readers Dr Peter Vamplew and Dr Ray Williams. This occurred at a time of year when both were becoming overloaded with work and in Peter's case was moving house interstate.

I would also like to thank David Herbert, Luke Fletcher, Terry Bigwood and Tony Gray for their technical support throughout this three year process. It's always good to know that there is someone around that can recover my Thesis.doc file when needed.

I would very much like to say thank you to Sung Sik Park, not only for putting up with me in the shared office for three years (a mighty effort on its own – especially when my kids come in and switch his server off), but also for the many discussions on the nuances of MCRDR. Also, I'd like to thank him for the use and help in using his MonClassifier database. This was an invaluable aid to the testing of the methodology developed in this thesis, in a real world environment.

During these three years I have had a number of helpful conversations with many of the other postgraduates and academic staff in the school of Computing. I would especially like to thank Dr Peter Vamplew, Dr Julian Dermoudy, Robert Ollington, Adam Berry, Bart Buelens, David Benda, Pauline Mak, Joel Scanlon, Phil Uren and Young Sok Kim.

Working on a PhD does not stop when you go home at the end of the day, it's an all day, life encompassing task. Therefore, it is not just those at the university that need to be thanked. I would very much like to thank my mother, father and grandmother (affectionately known as SuperNan by my children) for their love and support, both over the last three years and all the years prior. I'd especially like to thank my mother for always being around to hear me waffle on and on about my thesis and to take the time to read it when completed.

I would especially like to say thank you to the three most beautiful and loving people I have ever had the pleasure to know, Uisce (6), Akalia-Shen (4) and Jayva-Kwan (1). Uisce regularly asked me how my thesis was going, Akalia-Shen liked to hear bits of it read to her as a bed time story (I guess because it put her to sleep), and Jayva liked to draw on it. Thank you, to the three of you, for your understanding, patience and never ending affection.

Finally, without the support of your partner writing a thesis would never be possible. I would very much like to thank Anitra Goriss-Hunter for her love and support over these last three years. This is an especially important achievement as she is also simultaneously writing up here final PhD thesis. Thank you very much – and I guess its now my turn to reciprocate.

To Visce, Akalia-Shen
and Jayva-Kwan

Whose intelligence and love, I can only strive to emulate.

Chapter Overview

Contents

Table of Figures

Table of Tables

1

Introduction

It's a funny thought that, if Bears were Bees,
They'd build their nests at the bottom of trees.
And that being so (if the Bees were Bears),
We shouldn't have to climb up all these stairs. *(Milne 1926, p6)*

Since the advent of the computer, people have been searching for ways to make these machines think, learn and acquire knowledge. There have been a number of methods applied to such tasks with varying degrees of success across many problem domains. For instance, artificial neural networks (ANNs) and other function fitting methods have shown great ability at being able to learn generalised solutions to problems. These systems can provide reasonable results to situations never previously seen. However, they generally require extensive and repeated training or calculation to form the required function.

Another highly successful approach to AI is through encoding human knowledge directly, generally referred to as Knowledge Based Systems (KBS) or Expert Systems (ES)[1]. Systems using such knowledge can provide extremely accurate classification. Additionally, once knowledge has been added to them it can be used immediately without needing repeated training. However, unlike ANNs, KBSs generally fail completely when faced with previously unseen situations, especially when they require knowledge from outside the system's domain. Moreover, unless a human expert is checking each conclusion there is no way for such a system to realise when it is arriving at incorrect conclusions.

The differences between these dichotomous approaches were further investigated by Gaines (2000). He looked at where within the overall framework for human activity, each group of methods have been applied with the greatest success. Figure 1-1 shows a reproduction of Gaines's (2000) *Levels and Worlds of Being* diagram, which attempts to capture the entire conceptual framework for human psychology, sociology, action and knowledge. The central region of this

[1] Both Knowledge Based Systems (KBS) and Expert Systems (ES) refer to the same type of system.

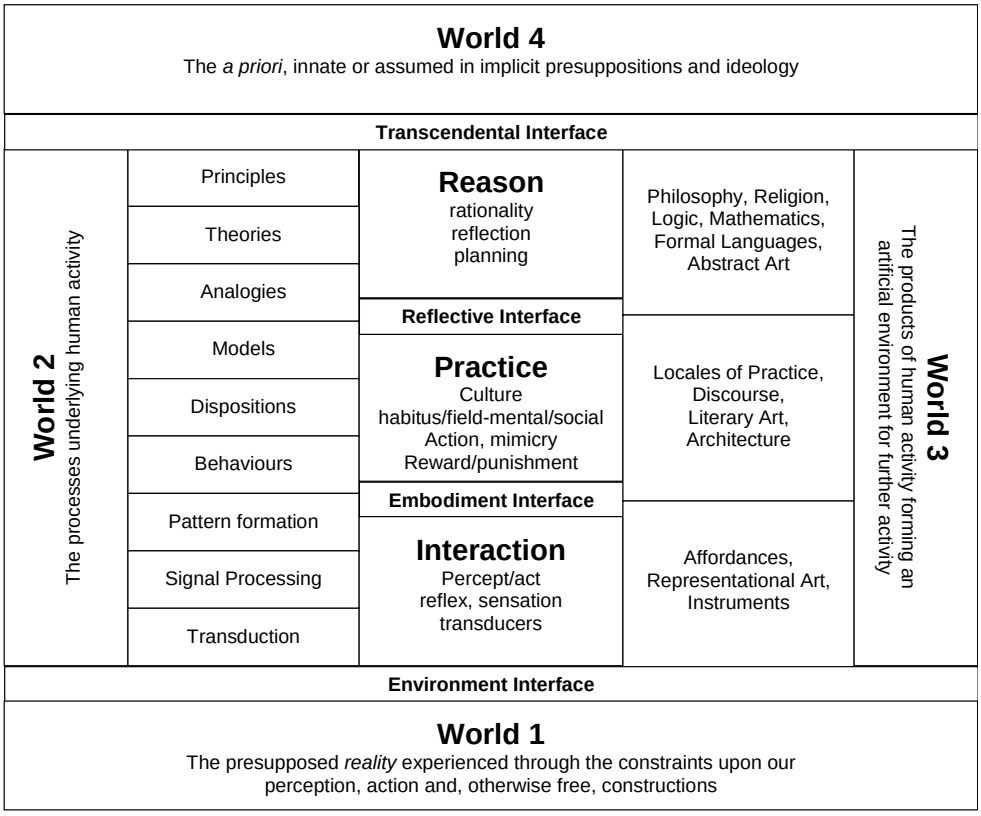


Figure 1-1: Conceptual framework for human psychology, sociology, action and knowledge. The central region shows a three-layer model of human entities. The four outer worlds set human entities within the context of Popper's (1968) worlds. The boxes on the left of the central core identify various models of knowledge transfer in relation to the three layers. The boxes to the right of the central core relate the three layers to various products of human entities. This model is a replication of the levels of worlds of being model (Gaines 2000, p 115).

diagram presents a three layer model of human entities, such as people, companies or governments. Gaines (2000) claims that connectionist approaches have emulated the *interaction* level with increasing effectiveness, while digital computation has done similarly well with the top brain functions of *reasoning*.

1.1 Conceptual Overview

Gaines's (2000) investigation into these different approaches found that the central area concerning the *processes of practice* is relatively untouched. Conceptually this thesis is interested in finding an approach that goes some way to filling this gap. This section will investigate the process of practise and how it relates to deeper forms of contextual representations of knowledge. This directly leads to this dissertation's algorithmic inspiration.

1.1.1 Process of Practise

In human psychology and sociology the process of practice is where the behaviour of people or organisations' is not governed by logic or a reflexive response to an event. Rather, they are ruled by their *cultural*, *mental* or *social* state at the time. For instance, a person's *habitus*, or predisposition to be effected by something (WordNet 2003), can significantly alter their response to a particular set of facts (Gaines 2000). One type of habitus, where behaviour is induced by normative rules, such as government codes or through business operations, has been successfully modelled by KBSs. However, these can still be problematic as normative rule sets are generally incomplete and require significant interpretation (Op cit). Gaines (2000) claims that the failure to adequately model the remaining process of *practice* is the "…greatest impediment to the development of expert systems" (Gaines 2000, p 115).

Currently there are two approaches whittling away at the region of *practice*. Firstly, connectionist researchers have been investigating the emulation of higher brain functions. Generally, connectionist work has struggled to develop systems that model higher brain functions as they tend to be mostly reactive (Gaines 2000). Machine learning methods like temporal difference learning can be integrated with connectionist approaches providing the ability to spread rewards over temporally separated events in a generalised form (Sutton and Barto 1998). This offers the ability to learn some simple actions (Op cit). However, these systems struggle to learn the amount of knowledge required for complex higher brain functions, thereby, limiting their effectiveness. Simultaneously, approaches such as Lenat's (1995) Cyc project, have attempted to extend the application of *reason* downward through the codification of general knowledge in a KBS. This approach is attempting to gain the connectionists' ability to generalise through brute force.

The most promising method for modelling the central region of *practice* though, as claimed by Gaines (2000), is through a system using multiple rule sets based on exceptions within a prioritized hierarchy. Furthermore, Gaines (2000) claims that Multiple Classification Ripple-Down Rules (MCRDR) (Kang and Compton 1992; 1994; Kang et al. 1995) goes some way to providing such a knowledge base structure. The multiple conclusions allow for a less rigid

outcome, which permits the resolution of conflicting constraints by providing a number of alternate possibilities. However, such a system still only provides specific alternate classifications rather than a *general constraint*, as Gaines (2000) claims is needed. Thus, a more general system of selecting admissible actions that satisfy the identified constraints, but are otherwise indeterminate, is more capable of realistically modelling human practice (Gaines 2000).

1.1.2 Context

The strength of Ripple-Down Rule (RDR) based methods in partially addressing the problem of modelling habitus, is based on their incremental nature, exception based knowledge structure and the ability to capture static contexts. Context can be thought of as the starting state or assumptions prior to inferencing. In the case of RDR, context is determined during inferencing. This allows simple maintenance within the context determined, through rule exceptions. This contextual based maintenance allows RDR based methodologies to capture more transient and tacit knowledge (Richards and Busch 2000; 2001). This is without the need to globalise knowledge to a general context, where no prior knowledge or assumptions are required before codifying, unlike many other KBS methodologies, such as KADS.

This failure of most methodologies to consider context has significantly hampered their knowledge acquisition process. However, methods, including RDR based techniques, that do incorporate either implicit or explicit context, assume that it is *a priori*, and therefore, deductive. This assumption leads to *static* representations of contextual based knowledge. However, context in certain situations could be considered *a posteriori*, and therefore, inductive (Brezillon 1999). For instance, during a conversation the context or assumptions of the people involved is constantly altered as the interaction meanders through different topics. Each new topic requires a different set of assumptions. Situations such as this require a dynamic and changeable representation. Dynamic contexts require the ability to have multiple start states that result in the same or similar contexts. Alternatively, a number of similar starting assumptions can result in many different contexts.

1.1.3 Hidden Contexts

Brezillon (1999) suggests that dynamic context can only be modelled during the solving of a problem or through interaction, and thus not directly coded in a static structure. A number of authors, such as Widmer and Kubat (1996) go further, suggesting that these new contexts are often differentiated by variables or facts that are hidden, even to those altering their contextual position. Such contexts are sometimes referred to as *hidden contexts*.

In this thesis the idea of a hidden context is a one that is only identifiable through influences on a system that can not be captured, such as human emotion or intuition. These situations arise where aspects of a contextual situation are hidden from the current state and the path taken to reach that position. A hidden context differs from a dynamic context due to the reason for the contextual position is unknown, whereas dynamic contexts are changing over time. Furthermore, the cause of a dynamically changing context can potentially be due to the influence of a hidden context.

The dynamic, and generally hidden, contextual nature of habitus, potentially prevents contextually static approaches from being able to encroach further into the central region of practice. It is common knowledge that different people react differently to particular events. Furthermore, the same person may react differently to the same event at different times. For instance, people will often wait to tell someone some news because they are '*waiting for the right time*'. This hidden and dynamic context of habitus means facts about a situation will be interpreted differently, resulting in a different response. Thus, the contextually static system's inability to incorporate this contextual information hinders their ability to accurately model the *process of practice*.

While proving the existence of such types of contexts is not the core of this thesis, the idea of these forms of contexts is one of the driving influences in the methodological design and in bridging the gap of the *process of practise*. The methodology developed in this thesis is designed at capturing these hidden contexts and goes some way to learning dynamic contexts. As shown in chapter 6, the success of the system to improve on the performance of a static methodology in many domains, goes some way to justifying that some form of contextual information is present.

1.2 Thesis Objectives

This thesis has been inspired philosophically, by the idea of finding a potential solution to the modelling of human practice by filling in the gap between the two opposing streams of research: ANNs and KBSs. Such a system would need to be able to handle hidden and dynamic contexts, as well as be able to generalise to some degree. This requires similar behaviour to the traditional strengths of connection based approaches. Yet in order to meet the higher brain function requirements of habitus, the system should also be able to acquire and retain knowledge quickly. This requires similar behaviour to the traditional symbolic approaches. In order to achieve such opposing goals, the core of this thesis is interested in developing a system capable of discovering hidden context and constantly adjusting for their dynamically shifting nature in a primarily symbolic based system.

These ideas inspired the aim of this thesis to extend an RDR based method's ability to represent knowledge in context, to also find relationships between those contexts. Such relationships represent a form of hidden context. If these relationships could be represented in a generalised form, which allowed for simple alterations then the methodology would also be capable of learning and adjusting these contextual relationships dynamically. The combination of these abilities would provide a form of *a posteriori* analysis that could provide reasonable classification when dealing with unique situations.

This inspiration and the derived aims led to the notion of combining a KBS with an ANN. The KBS selected for use in this thesis was MCRDR, as this is currently one of the methodologies most capable of modelling the *process of practice* (Gaines 2000). If such an amalgamation is possible then this could incorporate the continuous learning and generalisation ability of the ANN, on top of the immediate incremental knowledge acquisition of MCRDR. Such a system would be trainable in any task for which both a reward structure could be developed and a source of expert knowledge was available.

The second area of interest to this thesis considers the idea that if an algorithm could be developed that met the above criteria then:

What possibilities would that offer?

While this is far too broad a question to be fully addressed by this thesis, potentially systems that incorporated higher forms of context could open up new and interesting areas for future research. For instance, a symbolic approach with a dynamic context based representation could be significantly more proactive during knowledge acquisition and develop its own meta-knowledge. Additionally, it could also be applied in application domains not normally associated with symbolic systems, such as knowledge discovery and reinforcement learning.

In this work the aim is to address one major possibility in relation to how the project's proposed algorithm could be applied. It aims to investigate how extracted hidden and dynamic contextual based knowledge could be used by a KBS for self analysis. By using this derived information, is it possible to detect the limits of its own knowledge? If the frontier of a rule base's knowledge can be detected, then it would be possible to identify situations where knowledge, beyond what is currently known, is needed to correctly classify a particular case or situation.

While not directly studied in this project, other areas of interest in the application of the proposed method is how its hidden and dynamic contextual information can aid in traditional domains, such as Information Filtering (IF), Knowledge Discovery in Databases (KDD), as well as interesting new avenues such as incremental induction. A second view of this work is how such an amalgamation could facilitate the use of expert knowledge within a neural network. Traditionally, such networks do not allow for an expert in the field to provide assistance to the network's ability to learn. This could be significantly useful in a number of areas where there is some existing expert knowledge already and the network is required to build on that knowledge rather than to start learning from scratch. This would provide a much faster learning system than traditional networks, without losing generalisation.

It should be noted, that while this thesis is inspired by developing a method capable of being applied to the process of practice, it is not feasible to directly prove the developed algorithm is suitable. This could only be determined through years of future development of applications requiring process of practice type solutions. Instead, prudence analysis is used as one example of its performance in an area requiring additional contextual meta-knowledge.

1.3 Thesis Hypothesis

The above aims can be summarised into two hypotheses that this thesis will answer in relation to the hybrid system developed.

> *Can careful hybridisation of two existing learning techniques, utilise the advantages of both by producing an improved learning system that is greater than the sum of its parts?*

This first hypothesis essentially addresses the methodology's proof of concept. As previously discussed, fundamentally the advantages and disadvantages of ANNs and KBSs are essentially dichotomous. The advantage of ANNs is that they are good at generalisation corresponding to KBSs' greatest weakness. Likewise, a KBS learns immediately upon a new rule being added to the system[2], whereas ANNs require numerous viewings of the same or similar cases to successfully learn. The above hypothesis asks whether the combination of these techniques result in a methodology that is capable of immediate learning of a KBS, while also being able to generalise like traditional ANNs.

In the general field of classification, and even in prediction tasks, such a capability is certainly of great value. Faster learning with maintained generalisation while incorporating expert knowledge into the training of a neural network, by itself, could be applied successfully in many potential applications. However, rather than exploring that avenue, this thesis will concentrate on the implications for KBS applications by addressing the second hypothesis:

> *Does the discovery of hidden and dynamic relationships between multiple contexts within MCRDR offer significant and useful information?*

This second hypothesis asks whether the resulting methodology's discovered meta-knowledge can be utilised to provide useful information. While the possibilities are really up to the imagination of those utilising it, this thesis will concentrate on the single application area of prudence analysis. The hypothesis is correct if the system can use the information learned to accurately produce warnings when further knowledge is required.

[2] This ability is clearly visible in RDR based methodologies, however, it is hidden in more traditional KBSs. In these systems the effect of rules being added are not visible until the system as a whole is put online. Fundamentally though, the added rules still represent a form of immediate learning.

1.4 Thesis Overview

The methodology developed in this thesis, builds upon an observed weakness in the Multiple Classification Ripple-Down Rules (MCRDR) methodology. The MCRDR inferencing process determines individual classifications, which are all uniquely derived without any consideration for what other paths may also have been followed. This represents a significant lack of cohesion between the multiple classifications found. This dissertation takes the position that because a case was classified in a particular set of classes, there must be either a *conscious* or *subconscious* relationship between those classes in the expert's mind. This relationship essentially identifies a higher level of contextual meaning and could be considered to be a *hidden context.*

1.4.1 Methodology

The algorithm developed in this thesis, referred to as Rated MCRDR (RM), seeks to identify these hidden relationships. This involves online analysis of the actual MCRDR knowledge map during the inferencing and acquisition processes. The extraction of information about the structure of the knowledge stored in an MCRDR knowledge base (KB) is not unique. For instance, a significant body of work by Richards (1998a) investigated using Formal Concept Analysis (FCA) to find hidden concepts in a KB structure. However, this was a once off static analysis and was more useful as a user's tool for knowledge reuse, rather than for actively improving learning or for handling dynamic contexts (3.4) (Op cit).

Rated MCRDR, described in its simplest form *(see chapter 5 for a full description)*, combines MCRDR with an artificial neural network. These methods are hybridised in such a way that the network is able to learn patterns of conclusions found from inferencing through the knowledge base. The position of rules within the MCRDR structure represents the context of the knowledge (Compton and Jansen 1988; Kang 1996) while the network's ability to adjust its function over time provides a means of learning dynamically changing contextual patterns.

This amalgamation, while appearing simplistic conceptually, is by no means trivial. The primary difficulty in the methodology is actually finding a way to

use the output from the tree structure as an input for an ANN. The difficulty arises from the MCRDR structure being an incrementally growing rule base. Thus, if we wish to attach a network to this growing structure then the network must also increase its input space in parallel with the tree. However, nowhere in the neural network literature, or any other function-fitting literature, has anybody attempted to use such a system in an environment where the input space is constantly growing. Therefore, there are no existing techniques that can be used to accomplish this task.

Rather than inventing a new function-fitting algorithm from scratch, this project has looked at a range of methods and attempted to find ones that hold some form of extendable nature. Many very powerful techniques, however, had to be discounted, due to an inability to find a way of extending their input space successfully. The methods found with some form of growth potential, were then altered significantly allowing for their use in RM. As a result a number of techniques were developed for learning the output patterns.

1.4.2 Classification and Prediction

Each of the methods developed were tested in both classification and prediction based problem domains. In this thesis a classification task is one that tests a systems ability to identify individual groupings from a finite collection for each case. A prediction task, on the other hand, involves identifying a precise value or rating from an infinite range for each case.

In this project two types of tests were conducted in both classification and prediction. The first experiment investigated the ability of the different methods at generalisation. This involved providing different amounts of training data and observing how accurately the methods were then able to classify or rate previously unseen cases. The second experiment investigated the ability of the various methods in an online domain. This involved showing cases in series without previous training. The level of accuracy in classification and rating were then recorded over time. These tests were designed to test the different methods' performances so they could be compared against each other finding which the best performers were.

The best performing method was then further compared against the neural network used in the selected method and MCRDR by themselves. These tests

were carried out on the same tasks and datasets. The results from these experiments showed that the hybridisation performed in RM provided a vast improvement over the original individual methods, which answered the first of the dissertation's hypotheses.

1.4.3 Prudence Analysis

While this hybridised technique shows great promise in improving learning and generalisation, the possibility of using the information in more elaborate ways was further explored. This exploration was through its application in the field of *prudence analysis*. Prudence analysis is the process of self verification of the active knowledge base during inferencing. It involves the KBS determining whether the classification concluded is correct. When it believes it may be wrong it warns the expert allowing them to check its validity.

Through careful application of RM, the method is shown to be a powerful predictor of the knowledge base's accuracy. Results show an improved performance, with much greater versatility, over previous attempts at prudence checking. Prudence analysis is potentially a very powerful tool. Primarily it frees an expert from the need to manually check every classified case, which previously has been a major problem with RDR based methodologies. It also is more useful than standard verification and validation techniques as they only indicate the completeness of a knowledge base with hypothetical, machine generated cases.

The true power of RM is further shown by its ability to use prudence analysis in a real world environment, using human knowledge. RM is used in conjunction with the *MonClassifier* (Park et al. 2003; 2004a) utility. *MonClassifier* is a monitoring tool for the World Wide Web (WWW). After being assigned a number of sites to be monitored, such as news sites, it will categorise every new addition into its database. This categorisation has been traditionally performed through MCRDR. The system has been highly effective, however, like all RDR based systems the expert must check each new article to ensure it has been correctly classified. This task is time consuming and limits the power of having such a classification tool. RM has been used to analyse the process to see how accurately it is able to predict misclassifications. The results highlight how useful RM can be in such environments.

1.5 Thesis Organisation

This thesis is presented in a simple and straight forward structure. At the highest level it has been broken up into three distinct parts. Each part is broken up further into a number of chapters, making a simple progression through the thesis. Finally, after the three parts chapter 9 provides a conclusion to the thesis. Following is a brief description of each part and chapter:

Part 1: Philosophy and Literature

The first part discusses the philosophical considerations, along with a detailed review of literature and background material, relevant to this thesis.

Chapter 2: Knowledge Based Systems: Philosophy and Systems

This chapter initially reviews current philosophically based ideas on '*what is knowledge*' and how they apply to the field of symbolic learning. It explores both the historical perspectives and why new philosophies have been sought over the years. This chapter also discusses the more recent notions of *situated cognition*, along with this dissertation's extension to these ideas. The aim of this chapter is to place the research carried out in this thesis within the contemporary philosophical thinking and to discuss how RM addresses some of these issues.

Chapter 3: Ripple-Down Rules

This chapter provides a review of the existing RDR based methodologies. Initially, it explains the base methodologies of RDR and MCRDR in detail and follows this with a very brief overview of some of the techniques developed beyond these original methods. Additionally, it will discuss in more detail other work that has previously been developed for finding hidden context through analysing KB structure.

Chapter 4: Artificial Neural Networks

The final literature review chapter is essentially provided to ensure the reader is sufficiently familiar with artificial neural network technology, in order to follow the later discussions on the development of RM. It briefly covers both the historical progression of ANNs, as well as an overview of the main methods considered in this thesis. Finally, the chapter also briefly discusses other function-fitting methodologies.

Part 2: Methodology and Initial Results

The second part fully details the various possible implementations and presents the initial results. These results justify the implementation decisions made in developing the methodologies, along with addressing the first hypothesis.

Chapter 5: Rated MCRDR (RM) Methodology

This chapter describes the RM methodology in detail. There are seven variations of RM developed, each providing different abilities and weaknesses. This chapter explains why each variation was believed to have merit to warrant its implementation and why particular design decisions were made. Additionally, it formulates the mathematical equations used in various components. Finally, it also elaborates on why some function fitting methods could not be applied to RM.

Chapter 6: Rated MCRDR (RM) Initial Results

This initial results chapter has two main purposes. The first is to fully test each method developed in a small number of differing tasks and to use the results to identify which algorithms operated most successfully. The second is to compare the most successful techniques with MCRDR and the ANN method used in RM. These tests aim to both justify the methodology developed and to answer the first hypothesis.

Part 3: Prudence Analysis

Finally, the third part provides two chapters investigating the application of RM in the prudence analysis domain. This part of the thesis is aimed at showing how RM addresses the second hypothesis.

Chapter 7: Discovering the Knowledge Frontier

This is the first of this dissertation's two prudence analysis chapters. It illustrates one area of great potential for RM in significantly improving knowledge acquisition. This chapter details two ways the algorithm can be easily used for prudence analysis and compares their effectiveness. Secondly, it identifies how the RM warning system compares to previous work in prudence checking and how the results can be altered to cater for different situations. Finally, it looks at how effective the system is when the user only checks cases for which they are warned, and what is the consequence on the final knowledge base developed.

Chapter 8: MonClassifier$_{RM}$: Venturing into the Real World

This second chapter on prudence analysis provides some key results identifying the methodology's true power, by using it in a real world application. This chapter is built around the addition of RM to the MonClassifier utility developed by Sung Sik Park. It highlights how the algorithm performs extremely well when used with human expertise on a real knowledge acquisition task. This chapter also investigates the effect of using the warnings in a real world environment.

Chapter 9: Conclusion

The final chapter provides a review of the thesis as a whole, highlighting the important results and how they answer the two hypotheses. As this thesis develops a new methodology that has potential to be applied in numerous problem domains, this chapter will also discuss a number of other possible applications.

1.6 Summary

This introductory chapter has broadly discussed the underlying philosophical inspirations underpinning this thesis. Primarily, this revolves around finding a method for modelling the *process of practice*, through finding dynamic relationships between static contexts in an MCRDR knowledge base. This chapter also identified and discussed the two hypotheses that this thesis aims to address. Finally, this chapter very broadly described the algorithm developed in this project and the areas it may be applied, such as classification, prediction and prudence analysis tasks.

Part 1:
Philosophy and Literature

2

Knowledge Based Systems: Philosophy and Systems

> *'It's like this,' he [Pooh] said, 'When you go after honey with a balloon, the great thing is not to let the bees know you're coming. Now, if you have a green balloon, they might think you were only part of the tree, and not notice you, and if you have a blue balloon, they might think you were only part of the sky, and not notice you, and the question is: Which is most likely?'*
>
> *(Milne 1926, p11)*

A basic axiom of software development is that any completed system will need to be changed and updated throughout its life span. This is particularly important for knowledge based systems, as Buchanan and Smith (1989) outlines that one of the five desirable attributes of such systems is that they retain flexibility. Conceptually, these systems are designed for adding new knowledge easily, due to being made up of many individual atomic pieces of knowledge or symbols. However, generally each piece is finely woven together with many of the other pieces, making a complex structure of relationships that is, more often than not, impossible to add new components to without introducing errors.

Difficulties can also arise during a system's construction due to the major bottleneck in the deployment of expert systems, knowledge engineering (Feigenbaum 1977). Knowledge must be elicited from an expert and transformed into a form that is suitable for insertion into the knowledge base (KB), such that it adds to the overall knowledge and does not create contradictions. However, conflicting knowledge from different experts or even from only one expert renders this highly problematic. For instance, Shaw (1988) reports that experimental results show that two experts, at best, only agree on 33.3% of the knowledge base constructed and, at worst, merely 8.3%. Additionally, communication problems between the experts and the knowledge engineers cause the knowledge engineers to have to transform the supplied knowledge into an appropriate form for the knowledge base.

One solution to these engineering and maintenance issues has been through using ideas from *situation cognition* (SC). SC rejects the notion, that knowledge is a static entity and once captured remains correct. Instead, it claims that knowledge is dependent on the context of its use and that knowledge is generated or, more accurately, reinterpreted at that point. There have been many methodologies developed that attempt to incorporate the context of knowledge. One such group of methodologies, which uses knowledge in context and directly targets the issues of KBS maintenance, is the Ripple-Down Rules (RDR) family of techniques, which are described fully in chapter 3.

This work, however, has only latched onto ideas from *weak SC*, where context is viewed as static, and has chosen to ignore the concerns posed by advocates for *strong SC*. *Strong SC* researchers claim that the effect of context is so strong that any symbolic based representation is fundamentally flawed and such a system can never achieve any true understanding of the environment that it operates within (Menzies 1998). However, *strong SC* advocates do not recognise the symbolic approach's success. Nor do they adequately define context and why it renders symbolic reasoning futile.

This chapter will look at the philosophical and theoretical basis of Knowledge Based Systems (KBSs) from three perspectives: knowledge acquisition (KA), knowledge representation (KR) and knowledge maintenance (KM). It will also, briefly look at some of the important expert system (ES) methodologies and tools: Knowledge Acquisition and Design Structuring (KADS), Protégé, Cyc, Case Based Reasoning (CBR), and Data Mining. The second section will review the relatively new field of applying the ideas from situated cognition and briefly examining the effects they have had on knowledge engineering. This will be followed by a brief description of two utilities for aiding the KA of context-based knowledge, Formal Concept Analysis (FCA) and Repertory Grids. Finally, the last section will discuss the differences between weak and strong situation cognition and propose a reinterpretation of SC. Fundamentally, this new interpretation allows for the incorporation of hidden and dynamic contexts into the artificial intelligence interpretation of SC. This new perspective will be discussed along with its implications for symbolic learning. It is believed that adequately providing for all forms of context will allow such systems to more effectively cater for the *process of practice*.

2.1 Philosophy of Knowledge applied to Artificial Intelligence

It has long been recognised that the two biggest problems in developing knowledge based systems is the knowledge engineering (KE) process and later the system's knowledge maintenance (KM). The knowledge engineering process consisting of two primary components: knowledge acquisition (KA), and knowledge representation (KR). This section will look at the philosophical and theoretical basis of KBSs development from these three perspectives: Knowledge Acquisition, Knowledge Representation and Knowledge Maintenance. Additionally, it will briefly review four high profile methodologies and tools.

2.1.1 Knowledge Acquisition

Firstly, before the process of acquiring knowledge can be examined, the question of '*what is knowledge?*' should be addressed. This section will look at the traditional view of knowledge (Newell and Simon 1976) and what effects it had on knowledge acquisition theory. These topics are extensively covered by other authors (Newell and Simon 1976; Russell and Norvig 1995); therefore, this section will only discuss it broadly, in order to sufficiently place the work in this thesis.

The traditional expert systems' view of knowledge was founded on the *physical symbol hypothesis* (Newell and Simon 1976), which takes the view that knowledge is comprised of symbols, and the connections between those symbols, representing pieces of reality. Furthermore, that *intelligence* comes from the appropriate manipulation of these symbols and relationships. This AI perspective of knowledge is not new and closely draws from the philosophies of Wittgenstein, Descartes, and ultimately Plato's archetypes (Compton 1992; Compton and Jansen 1990).

Expert System researchers, therefore, assumed that such symbols and relationships should be extractible and, thus, usable without the further need of the expert. This was fundamentally a reductionist strategy, the logical extension of which led to the *knowledge principle* (Feigenbaum 1977; Lenat and Feigenbaum 1988; 1991), essentially suggesting that the success of an expert system is dependent on the amount of information about symbols and their

relationships in the knowledge base and not the inferencing or reasoning strategy employed. Therefore, when difficulties arose, the common misconception was that this came about from ineffective knowledge acquisition, and furthermore, that experts do not "…communicate the underlying knowledge very well" (Compton and Jansen 1990, p 242). A closer investigation of these problems, however, reveals three primary areas of difficulty:

- *Knowledge base consistency.*
  Each new piece of knowledge extracted from an expert, must be integrated so that no conclusions are changed that render other knowledge in the system invalid. Tools have been developed for validating and verifying KBSs that can provide some assistance in a few systems. However, once flagged, an inconsistency still must be corrected and hand tailored by the KE without changing the essence of what the expert claimed in their justification (Compton and Jansen 1988).

- *Inconsistent use of symbols by experts.*
  Shaw (1988) identified that experts often disagree on '*what is correct knowledge'* when presented with it in the form of a knowledge base. However, generally these same experts have no problem discussing ideas with each other, even though they apparently have conflicting views on the individual pieces (Compton and Jansen 1988). This disagreement, however, can significantly hamper knowledge acquisition.

- *Experts' justifications can be affected by the knowledge engineer and the tool used.*
  People, thus also experts, tend to explain concepts based on the listener's level of knowledge. Additionally, Compton et al. (1991) suggests that not only do experts provide contextual knowledge instead of global knowledge, but the context often depends on the framework of the questioner. Thus, a KE may ask a question with certain expectations, such as they may expect the knowledge being provided to be casual. The anticipation of a relaxed and unstructured response is subtly conveyed to the expert and the expert tends to respond within that context.

The last two points challenge the original *physical symbol hypothesis* idea of knowledge as being a static entity that can be extracted. These two points suggest that the individual symbols can vary in meaning between experts and the connections between those symbols vary according to the context in which the knowledge is being used. This shows that any form of static extraction of symbols and connections ultimately is doomed to fail in all but the most rigid of knowledge domains. The problems in knowledge acquisition originate not so much from the extraction of the knowledge from the expert, but from the KE's attempts to twist and warp the acquired knowledge into a form suitable for the KBS in which they were attempting to fit the knowledge.

For example, an expert shows a knowledge based system a *dog*, which it mistakenly classifies as a *bird*. The expert will then offer a reason why the KBS is wrong. They do not explain from first principles why it is a dog, which is the form the KE requires to codify a rule set; instead, they simply offer an explanation that separates the dog from the suggested conclusion of a bird. For instance, the expert may simply claim it is a dog because it has *four legs*. The expert's context assumes we are not talking about a *table* with four legs. Therefore, the knowledge provided is dependant on this context. Thus, when the KE must integrate the newly acquired knowledge into the existing KB they must transform it into a globalised form. Globalised knowledge assumes no prior assumptions. Therefore, apart from the new rule claiming a dog has four legs they must also provide facts that identify it as a domesticated carnivorous mammal from the genus Canis.

2.1.2 Knowledge Representation

Given that acquired knowledge is context dependent; there have been numerous methodologies developed attempting to either acquire already globalised knowledge or of representing knowledge so that it facilitates the easier inclusion of the gathered knowledge. These methods are primarily from what has been referred to as *second-generation ES*. Second-generation ES moved away from the more ad hoc KB structuring used in the earlier shell based systems, called *knowledge transfer* (Newell and Simon 1976), towards a *knowledge-level modelling* (Newell 1982) approach. These new model-based ES approaches were believed to provide a richer knowledge representation and, thus, the ability

of capturing deeper and different types of knowledge, such as tacit knowledge (Richards 2001).

The majority of these methodologies, however, still view knowledge as static, which once captured remains correct. They are in fact just new approaches to facilitating the conversion of experts' knowledge into a form that can be stored in the knowledge base. They generally do this by forcing the expert to provide their knowledge to the engineer in a more structured approach, causing them to globalise their knowledge at the point of extraction. This reduces the need for the engineer to extensively convert the knowledge. While this is a significant improvement over the ad hoc approach, the fundamental problem is that it transfers part of the difficulty of knowledge acquisition from the engineer to the expert, rather than actually solving the root issue of knowledge representation. It would be preferable to simply store the knowledge in the contextual form naturally gathered than requiring anyone having to globalise it first.

2.1.3 Knowledge Maintenance

While knowledge maintenance is presented here separately from KA and KR it is in fact a microcosm of the same issues already described. It requires both the ability to acquire new knowledge, as well as being able to insert it into the existing KB. However, it usually presents as a much more difficult problem for ES developers. There are three primary reasons for this added difficulty. First, the KE usually only wants to make the minimal amount of changes when performing maintenance and does not wish to restructure the entire KB. Second, often the original experts and engineers are no longer as intimately involved with the project, making updating problematic. Finally, the ES's domain can often shift, including new areas not covered by the existing system, after its deployment. Without a complete redevelopment this can significantly increase maintenance problems, as found in the Xcon system (Bachant and McDermott 1984). However, as the previous section briefly argued, KA, KR and KM are not necessarily separate tasks and if a more holistic approach is used then the difficulty created by these separate views can be resolved.

2.1.4 Knowledge Based Systems

There are numerous methodologies and tools that have been developed to aid in the creation of KBSs, which attempt to address the KA problem discussed above. There are many books, papers and reviews (such as Gennari et al. 2002; Leake 1996; Lenat and Guha 1990; Schreiber 1993; Schreiber et al. 1993) that investigate the issues associated with the problem in detail, and so this section will only very briefly introduce the more widely used or relevant approaches developed, so that readers can place the work in this thesis in the overall KBS field of research. The approaches discussed in this section are Knowledge Acquisition and Design Structuring (KADS), Protégé, Cyc, Case-Based Reasoning (CBR) and Data Mining.

2.1.4.1 Knowledge Acquisition and Design Structuring (KADS)

KADS (Knowledge Acquisition and Design Structuring) is the outcome of the European research project ESPRIT-I P1098 initiated in 1983 with the aim of developing a comprehensive, commercially viable methodology for knowledge-based systems (Wielinga et al. 1992). KADS is recognised as one of the first true methodologies developed specifically for the development of KBSs and is still widely used today. It was developed in particular to address the problems found in the *knowledge acquisition bottleneck*. The researchers' view of the problem was that knowledge acquisition failed in traditional systems through an inability to get to the *deep-knowledge* of the experts, primarily due to a lack of structural constraints on the experts during knowledge extraction.

Rather than viewing KA as filling a container of knowledge, the KADS perspective is an operational model that displays a form of observed behaviour which is "…specified in terms of real world phenomena" (Wielinga et al. 1992, p 6). Basically, KADS has an array of modelling techniques, where the expert and engineer work together to build up a set of tasks so that knowledge acquisition can be approached in a systematic way. This ensures, as best as possible, that nothing gets left out and the people concerned are aware of where they are up to in the process (Compton et al. 1993). This divide and conquer approach to KA is the basic underpinning of all *task* orientated methodologies.

The reason approaches, such as KADS, work reasonably effectively during development is because the expert themselves model a particular individual task. This forces the expert to step out of their current context into the global knowledge domain. This process of the expert now providing more globalised knowledge prevents the engineer from being required to extensively convert the knowledge further. However, the resulting knowledge base is still global in context and static in representation. Wielinga et al. (1992) even defines domain knowledge in KADS as being static knowledge.

This static representation, however, leads to one of KADS greatest shortcomings: knowledge maintenance. Nowhere in the KADS methodology is the knowledge maintenance issue addressed directly. Instead it assumes that the spiral life cycle model will continue infinitum. This omission has resulted in alterations to the basic methodology, such as *structure preserving design* (Schreiber 1993), which preserves the *information content and structure* in the knowledge-level model, within the final artefact. Therefore, the system not only provides the static domain knowledge but also the relationships between the artefact and the original knowledge sources and/or meta-classes. This makes development a process of adding implementation detail to a knowledge-level model, which makes it more possible to trace omissions or inconsistencies in an artefact back to the relevant part of the model, considerably simplifying maintenance (Killin 1993; Schreiber 1993). This essentially, although only in a limited form, can be seen as an attempt to include some form of context within the symbols.

2.1.4.2 Protégé

Protégé, a generalisation of the OPAL and ONCOCIN systems, is a knowledge-based systems development environment that has been evolving since the mid 1980s (Gennari et al. 2002; Musen 1987). Initially, it was a simple program designed for the medical domain, protocol-based therapy planning, but has since had many reimplementations, becoming a much more general-purpose set of tools. The original goal of Protégé, like the majority of new methodologies at the time, was to reduce the knowledge-acquisition bottleneck. This was accomplished through reducing the role of the KE in the construction of KBs (Gennari et al. 2002; Grosso et al. 1999).

Musen's (1987) *information-partitioning hypothesis*, which asserts that there exists a qualitative division among the types of information a KBS requires, is the primary basis behind the Protégé methodology. This hypothesis proposes that the knowledge acquired in one stage is also meta-knowledge for the subsequent stage, and thus, can be used to determine what KA-tools should be used in that next stage (Grosso et al. 1999). It's Protégé's use of this information-partitioning hypothesis along with its utilization of task specific knowledge to generate customised KA-tools that allows for the simplification of KA (Gennari et al. 2002; Grosso et al. 1999).

Later incarnations of Protégé worked towards making knowledge-bases more reusable. For instance:

- Protégé-II removed knowledge concerning the problem-solving method (PSM) from the KB, by formally modelling the PSMs and then using the method ontologies to define mappings. This converted Protégé's original informal model to a formal one (Grosso et al. 1999; Puerta et al. 1992).
- Protégé/Win allowed for modularity of knowledge bases through the use of components (Gennari et al. 2002; Grosso et al. 1999).
- Protégé-2000 adopted the Open Knowledge Base Connectivity (OKBC) (Chaudhri et al. 1998; Fikes and Farquhar 1997) knowledge protocol, allowing greater expressivity, a clean model-theoretic semantic and a greater possibility for maintenance and reuse (Gennari et al. 2002; Grosso et al. 1999). It was also rebuilt around a three layer model with fully replaceable and interchangeable components (Grosso et al. 1999).

Protégé's underlying methodology of building separate ontologies and then constructing knowledge bases from these components is not unique. This can be seen for example in LOOM (MacGregor 1991) and GKB (Karp et al. 1999). Like KADS, Protégé's knowledge-level modelling of framework ontologies can be effective and help with knowledge reuse. These ontologies, however, are constructed prior to the knowledge-base, and therefore, the knowledge within is still global to the component. Theoretically though, individual ontologies may be created in context, however, this usually conflicts with the components' potential for reuse.

2.1.4.3 Cyc

Cyc, short for en*cyc*lopaedia, did not start out as an attempt to develop new methodologies for knowledge acquisition. Rather, its intention was primarily to solve the problem of *brittleness*, where a small amount of missing knowledge can be significantly detrimental to the system. However, as a result it has had implications for both KA and KR methods. A KB's brittleness comes from its concentration of specialised knowledge within a single narrow domain. Therefore, when knowledge is required from just beyond this domain the KBS collapses. Fundamentally, this is part of the same problem the previous methodologies were attempting to fix. Their unarticulated view had been if we can extract deeper knowledge then the system will be less brittle. Lenat et al. (1990) however, argues that brittleness is the result of insufficient commonsense knowledge within the KB. It is this lack of commonsense knowledge that Cyc has been developed to address. The Cyc system is a universal schema with millions of directly entered and inferred commonsense axioms that make up hundreds of thousands of general concepts (Lenat 1995).

While methodological development was not the driving force behind Cyc, it was one of the results. During the system's development it was obvious that existing methodologies were woefully inadequate at scaling to the required size or at representing particular concepts. Cyc incrementally developed its own representation language then, to address repetition issues that eventuated, and periodically smoothed out the resulting structure (Lenat et al. 1990). It uses a frame-based language embedded in a first order predicate calculus framework with a series of second-order extensions that allow the representation of defaults, reification, and for reflection (Guha and Lenat 1994; Lenat 1995; Lenat et al. 1990; Pittman and Lenat 1993). The inference engine used for Cyc was also incrementally constructed using more traditional computer science data-structures and algorithms.

Lenat (1995) argues that the majority of *assertions* could not be made correctly without the use of some form of context. For instance, the statement '*you cannot see a persons heart*' assumes that the person is not currently undergoing open heart surgery. Alternatively, the assertion could also represent a metaphorical meaning. Cyc's solution was to place each assertion into one or

more explicit contexts, through the use of *microtheories*. Within each context, the assertion is then given its default conclusion. Each context is itself an individual KB. Additionally, the provision for being able to import assertions from other contexts was also included in Cyc allowing the combination of contexts and contradictory assertions to be resolved (Lenat 1995; Lenat et al. 1990).

The Cyc project's brute force approach shows potential as a basis for domain specific KBs to be built upon. However, Cyc's ad-hoc development methodology requires explicitly-stated knowledge that can often be dated or invalidated by the time it is eventually used in a real world system. This thesis asserts that: commonsense knowledge is one of the most *a posteriori-contextually* dependent forms of knowledge, due to its high dependence on culture and time. For instance, individual axioms not only change conclusions between contexts, but they also can change within the same context between different culturally independent minority groups. Furthermore, the contexts themselves within these groups also change over time. One of Lenat et al's (1990) own examples of knowledge in the Cyc system is:

> Payments of less than ten dollars are usually made with cash; those over fifty dollars are usually made via check or credit card (Lenat et al. 1990, p 43).

Such an assertion is highly cultural, location and time specific and, therefore, very susceptible to failing in a contextually-dynamic environment. For example, given such knowledge, one must ask *'how relevant is this?'* to the following: a peasant farmer in central china, who has no concept of how much a dollar is worth; to people in a war zone that have no access to secure financial institutions; or, to someone living in 2010 where all transactions are made with smart cards. Therefore, even though various static-contexts are identified in Cyc, the absence of *dynamic-contexts* renders its approach as far too simplistic and highly susceptible to obsolescence, for the development of a (near) complete KB of commonsense knowledge. For instance, Clancey (1991) likens common sense knowledge to that of chaos theory, where projects such as Cyc attempt to collect it like "…so many butterflies" (Clancey 1991, p 245).

2.1.4.4 Case-Based Reasoning (CBR)

Case-based reasoning (CBR) is not a single methodology but a field of research as diverse as KBSs. It originated initially to solve the KA bottleneck by attempting to capture the context of knowledge through the use of cases. The idea was to represent a concept through extension. That is, a concept is defined by a set of instances (cases). The basic idea behind CBR is two fold. Firstly, a CBR system attempts to solve a particular problem, in the form of a case, by searching and finding a similar, previously seen, case (or cases) and reusing that earlier case's solution, or a modified version of the solution. Secondly, the CBR system attempts to incrementally learn by using the success or failure of a solution (Aamodt and Plaza 1994). For instance, if a solution is correct then the new case is added with the solution given so it can be used in future similar situations. However, if the solution is wrong, then the reason for the failure is ascertained, as best as possible, and stored to avoid the error in the future.

The majority of research in CBR is around the problems of:

- Knowledge representation – must allow for effective and efficient searching and the inclusion of new case knowledge.
- Retrieval – using a partial problem description the CBR system must find the closest matching previous case(s) using an efficient method for case comparison.
- Reuse – investigates which aspects of a case are useful for future problem solving, as well as finding the difference between the current and previous case.
- Revision/Adaptation – if the solution was wrong then use domain-specific knowledge or user input to revise the solution.
- Retainment – if the solution was correct then use an aspect of the case to expand the area of the systems solution space by integrating the new case into the memory structure (Aamodt and Plaza 1994).

CBR methods are effective to some extent because experts are significantly more open to discussing details of a case and the associated solution than abstract general rules (Leake 1996). Additionally, their effectiveness also stems

from the provision of contextually relevant information being provided regarding the solution, improving maintenance issues and significantly reducing the KA bottleneck (Leake 1996). However, the development of effective CBR based systems is highly dependant on effective retrieval and adaptation functions (Khan 2003), which generally require domain dependant knowledge.

Fundamentally, CBR fits within a more situated cognitive (2.2) view of knowledge. However, most of the current research has overlooked that dynamic knowledge does not only relate to the domain knowledge held within the individual cases, but also in the control and problem solving knowledge of the domain. The problem in CBR is that the majority of systems rely on static predefined rules or procedures for their retrieval and adaptation functions. CBR system developers struggle, however, to anticipate all the difficulties that may be encountered in a domain during development, thus, causing a major bottleneck (Khan and Hoffmann 2003). Yet the nature of learning in CBR is implicitly incremental (Aamodt and Plaza 1994; Leake 1996), and thus, should be capable of handling such difficulties. Khan (2003) proposes that the process of acquiring and using both case specific knowledge and general domain knowledge should also be made within the context of the problem solving process and monitored by an expert.

2.1.4.5 Data Mining

The world is becoming more like the infinite library from *The Library of Babel* (Borges 1956) every day, overflowing with data from which people are unable to extract meaningful information. Knowledge discovery in databases (KDD) is a field of research attempting to solve this dilemma and is seen as the process of extracting "…implicit, previously unknown and potentially useful knowledge from data" (Frawley et al. 1992).

Data mining is not a single methodology but a field of research as diverse as KBSs, nor is it specifically related to Machine Learning techniques. It also crosses into statistical analysis and database systems. One area of research within KDD is concerned with the ability to find meaningful classifications and predictions of values for the tuples contained within a database.

Expert Systems in data mining generally use knowledge extraction methods to form a classifier or predictor. These have the advantage of forming high

quality results due to the inclusion of expert knowledge. The problem, however, is that they do not allow for autonomous knowledge discovery. Therefore, such systems will only find results that the expert is capable of giving examples about and will not find unknown patterns. Thus, Knowledge Discovery in Database (KDD) generally relies on other machine learning tools, forgoing the advantage of expert knowledge.

One method that is highly effective at automatically generating rule bases, capable of classifying and predicting values from smaller data sets is decision trees. However, while in theory decision trees could scale effectively to larger databases, due to only having $n(\log n)$ complexity, they suffer from requiring the training set to be resident in memory (Han and Kamber 2001). More recent systems such as SLIQ and SPRINT have attempted to address this issue. However, they require pre-sorting of data sets as well as complex and expensive data structures that reduce their effectiveness with large training sets (Han and Kamber 2001).

Another application of rule based approaches in data mining is in combination with other classifiers. In these methods a number of classifiers can be trained or built separately then their various guesses combined to find the ultimate classification. Alternatively, classifiers can be chained as in stacking (Wolpert 1992) and cascading (Gama 1998) methods. These methods have multiple layers of classifiers where the results from the first layer classifiers are fed in as input to the second layer of classifiers (Estruch et al. 2003). It should be noted that the method developed in this thesis is a form of stacked classifier with the exception that the input into the second layer comes solely from the output of a single classifier in the first layer, rather than from multiple classifiers.

One of the primary drawbacks of stacking and cascading is that comprehensibility is generally lost. This is due to the subsequent layers only receiving attributes that are in terms of the previous layers conclusions (Estruch et al. 2003). This is potentially not the case in the method developed in this thesis due to their being a direct continuity between layers through the use of only a single classifier in the first layer. However, to prevent the scope of this thesis from expanding this is not explored further.

2.2 Situated Cognition applied to Artificial Intelligence

Problems such as inconsistent use of symbols and experts' justifications being affected by the context of the listener, oppose the original *physical symbol hypothesis* idea of knowledge as being a static entity that can be extracted. This returns us to the original question of *'what is knowledge?'* Situation Cognition moves away from these traditional definitions of knowledge and instead argues that knowledge is generated at the time of use and that context-independent assertions cannot accurately model human cognition (Menzies 1996; 1998). The proponents of Situated Cognition tend to fall into one of two camps.

- Weak Situated Cognition argues that when the human agent uses a particular description of knowledge that they use the context of the current problem or situation to continually reinterpret that description.
- Strong Situated Cognition takes a significant step further, claiming that context has such a potent influence on the human agent that systems should be purely reactive and that we should discard symbolic representations altogether (Menzies 1996; 1998).

Therefore, Situated Cognition (SC), in its *weak* and most common interpretation, views knowledge instead as being mostly context based (Menzies 1996). SC and its subfields, situated automata (Maes 1990; Waldrop 1990) and situated action (Agre 1990; Suchman 1987), are philosophically justifiable through work such as Bartlett (1932), Piaget (1970), Jenkins (1974) and Bransford et al (1977). SC attacks the *Platonic*[3] view of knowledge and memory and instead claims that knowledge is re-constructed each time it is needed (Agre 1990; Clancey 1991; Compton and Kang 1993; Maes 1990).

This dynamic re-construction of knowledge involves the combining of two factors: firstly, where the agent has come from to reach this particular point, and the location of the agent's eventual goal. For instance, Clancey provides the example that *"…at a base level a person is always like a dancer balancing [his/]her next steps against the inertia of past movements and [his/]her view of where s[/]he is going"* (Clancey 1991, p 244).

[3] Plato's archetypes are often regarded as a major influence to the origins of the physical symbol hypothesis (Compton, P. (1992). Insight and Knowledge. AAAI Spring Symposium: Cognitive aspect of knowledge acquisition, Stanford University.

2.2.1 Knowledge Acquisition in Context

A new perspective on knowledge does not necessarily imply that the method by which knowledge is acquired needs to be reconsidered. Essentially, all that SC does is allow a knowledge engineer (KE) to view the knowledge they gather differently. The problems identified for knowledge acquisition in section 2.1.1 did not originate from the extraction of knowledge from the expert, but from the KE attempts to globalise the acquired knowledge into a form suitable for the KBS. Therefore, the key to improved KBS development is not so much in *how knowledge is acquired*, but *how that knowledge should be represented*.

2.2.2 Representing Knowledge in Context

As discussed earlier the traditional *physical symbol hypothesis* (2.1.1) views extracted knowledge as unchanging. Therefore, ES methodologies can only build knowledge bases (KB) with a static knowledge representation (KR). The situated cognition view of knowledge, however, indicates that knowledge is relative to the context it was re-constructed, and therefore, changeable. Thus, to truly reduce the problem of KA, a representation that incorporates context should be used. It follows then that a KR methodology that intends to incorporate context, must be able to assimilate change. This does not negate the possibility of specifying a KB prior to its application, but it does mean that the previously specified store of knowledge should be dynamically alterable.

Ideally a system should have a knowledge representation technique that is sufficiently flexible that it could change knowledge or include the context of the supplied knowledge. This would allow knowledge extracted from an expert to be significantly more easily included within the KB and, thereby, preventing the need for either the KE or expert to interweave knowledge into a global perspective. This would significantly simplify the problems associated with knowledge acquisition, as the conversion from context knowledge to global knowledge would no longer be required.

2.2.3 Maintaining Knowledge in Context

Menzies (1998) identifies one of the primary changes in knowledge engineering brought about due to the situated cognition view as a new emphasis on

maintenance rather than design. This is due to the context in which knowledge can be located is often only identifiable in an online environment during the maintenance phase. This directly challenges the approach of second generation KBS methodologies like KADS, which focus on design and don't even discuss maintenance in the core set of tasks. Menzies (1998) suggests that one approach to situated knowledge engineering is to build two knowledge bases. The first captures system knowledge like any other KB, while the second models the impact of the first KB on its environment and vice versa. For example, the BRAHMS system by Clancey et al. (1998) and REMAP by Ramesh and Dhar (1992) both attempt to model the environment around a KB.

Another approach is to directly represent the context of the knowledge being stored within the KB itself. This context can either be stored implicitly or explicitly. For instance, the Cyc ad hoc approach uses microtheories, which allows for a form of explicit contextual information to be included in the KB. Ripple-Down Rules (RDR), discussed in Chapter 3, implicitly includes context within the structure itself..

2.2.4 Contextual Knowledge-Based Systems

There are only a limited number of tools and methodologies that have been developed to aid in the creation of KBS through the incorporation of context. This section will briefly introduce some widely used approaches so that readers can place this dissertation's work in the overall KBS field of research. The two tools used for identifying contextual information discussed in this section are Formal Concept Analysis (FCA) and Repertory Grids. A third methodological approach, Ripple-Down Rules (RDR), is discussed in chapter 3.

2.2.4.1 Formal Concept Analysis (FCA) in Contextual Knowledge Acquisition

Formal Concept Analysis (FCA), originally developed by Wille (1981), is based on lattice theory. FCA is not a KA methodology, rather it is mathematical tool used for the discovery of concepts. It was designed to provide a basic answer to two fundamental questions in relation to a concept; how to appropriately classify objects within a particular context, and what the dependencies are between attributes (Wille 1981). Its relevance to KBS comes from its representation theory and how that can be used in knowledge representation.

FCA is rooted in the philosophical idea of a concept as a single unit of thought that is made up of two components: *extent*, which covers all objects (entities) belonging to a particular concept; and *intent*, which comprises the attributes (properties) that are valid for all those objects (Wille 1981). Put another way, a formal context, *K*, is a triple *K(G, M, I)* containing the set of objects G^4 forming the *extension* of the concept, and a set of attributes M^5 represent the *intention* of the concept, which are linked together by the binary relation *I* (Richards 1998a; Wille 1981).

The advantage in the FCA approach is that lattice theory provides a simple mathematical vocabulary for discussing order, and especially when used in systems where there is a natural sense of hierarchies (Birkhoff 1938), due to its theory of substructures (Wille 1981). This advantage results in the ability of a complete lattice, called a *concept lattice* or *semantic net*, providing a conceptual clustering of objects via the *extents* as well as a representation of the implications between the attributes via the *intents* (Wille 1992). It is this ability to express all relationships between attributes, describe objects in terms of the concepts contained and to show the relationships between those concepts, which makes FCA a powerful representation for knowledge (Richards 1998a).

Generally, FCA has not been used as a knowledge representation methodology by itself. Instead it has been used for knowledge acquisition (Wille 1989; 1992) in many KBS based areas, such as CBR (Diaz-Agudo and Gonzalez-Calero 2001) and ontology construction and maintenance (Stumme et al. 2000), as a means for discovering knowledge embedded in cases through these identified relationships. Within these areas FCA has been highly effective at extracting contextual knowledge.

Clancey (1991) mentions, however, that although semantic network based approaches can embody a cognitive model that show human-like behaviour, they are limited to a "… grammatical model of cognition" (Clancey 1991, p 251). Therefore, they fail to capture non-verbal concepts or to "…model the perceptual-conceptual learning…" (Richards 1998a, p 155) evident in humans' attachment of meaning to a grammar (Richards 1998a). Clancey (1991) continues by pointing out that concepts are being regarded as *things* rather than

[4] G stands for *Gegenstande* in German.
[5] M stands for *Merkmale* in German.

the "…processes of perceiving and processes of behaving" (Clancey 1991, p 252). Additionally, Wille (1981) himself identifies the method's lack of scalability for the inclusion of all relevant attributes and relationships with large contexts, potentially limiting its use.

2.2.4.2 Personal Construct Psychology and Repertory Grids

Kelly's (1955) book *'Psychology of Personal Constructs'* introduces his *personal construct theory* (also referred to as *Personal Construct Psychology* (PCP)), where he postulates that "A person's processes are psychologically channelled by the way in which he anticipates events" (Kelly 1955, p 46). Basically, he viewed all people as their own *personal scientist at anticipation*, where an anticipatory model of the environment was reflexively applied to the self (Shaw and Gaines 1992). Shaw and Gaines saw PCP as a "…constructivist position in modelling human knowledge…that characterizes conceptual structures in axiomatic terms" (Shaw and Gaines 1992).

Kelly's (1955) emphasis is on the space created by the process of making distinctions rather than being defined by the *elements* identified. Within this *psychological space* a *construct* is like a single plane that slices through a large collection of events (Kelly 1970). Fundamentally, a construct contains a triple of two disjoint distinctions that are mutually subsumed by the third, referred to as the *range of convenience* or the *subsuming concept* (Richards 1998a; 1998b). Essentially, the disjoint pair bound the extremes of the range of convenience or concepts. Therefore, this provides a means for bounding concepts, allowing knowledge acquisition to start with a more encompassing view and acquire knowledge inwardly; rather than the traditional approach, of starting at the centre and acquiring knowledge outwardly.

In order to actually acquire the relevant constructs and elements for a domain, Kelly also developed the *repertory grid technique* (Richards 1998a; 1998b). It is designed to bypass a person's cognitive defence and provide an avenue to their underlying construction system by asking them to compare and contrast various examples. Thus, repertory grids use the idea that people are more able to offer context based examples than defining globally based rules. Essentially, the repertory grid is a method of finding concepts, the structures within those concepts and the relationships between those structures without

eliciting them directly. This can be more effective than the conventionally based methodologies that directly extracted models (Richards 1998a).

Kelly's repertory grid technique has had widespread success in a number of domains and continues to be applied in both manual and computerised applications. Some systems that use the repertory grid are the Knowledge Support System Zero (KSS0), Expertise Transfer System (ETS) and AQUINAS (Boose et al. 1989). Gaines and Shaw (1993) also have provided a general framework to assist users in the elicitation of the conceptual structures when using repertory grids (Richards 1998a).

PCP is similarly simple in its acquisition of knowledge to RDR. Both view the concept of model derivation as being difficult and problematic, preferring the simpler technique of eliciting knowledge directly from an expert without the need of a knowledge engineer. However, the captured knowledge is different, in that RDR develops an assertional KBS, while PCP captures a conceptual model from which a terminological KBS can be extrapolated (Richards 1998a). As discussed in section 3.4.1, Richards (1998a; 1998b) developed a combination of RDR and FCA in order to create a terminological KBS with RDR.

2.3 Hidden and Dynamic Context

The application of ideas from weak situation cognition has already shown great potential to significantly improve Knowledge Acquisition and maintenance in many domains. However, in its weak form SC assumes context is static and once found can be codified. This is once again falling into the same trap that KBS developers did when using the *physical symbol hypothesis.* This assumption simplifies the problem domain allowing KBS developers to ignore more difficult unsolved situations. This is fine when used in situations where context is for the most part static, such as well defined areas like medical diagnostics. However, in many real world situations, such as general knowledge, this abstraction reduces the potential of such methodologies being successfully applied.

The more radical view of strong SC agrees that the process of abstraction and representation simplifies the environment too much. Brooks (1991) claims

that giving a computer a human *Merkwelt*[6] is flawed because it is based on our own introspection. There is no evidence that what we perceive as our *Merkwelt* is in fact what occurs internally and "…it could just as easily be an output coding for communication purposes" (Brooks 1991, p 141). Brooks identifies two primary problems with the use of the human *Merkwelt*. The computer, robot, or software agent does not perceive the world in the same way a human does. Furthermore, even if it did, there is no guarantee the human introspectively derived *Merkwelt* is correct.

While these arguments for strong SC are legitimate, the conclusion to reject all symbolic representation is flawed. As noted by various authors (Anderson 1990; Menzies 1996; 1998; Patel and Ramoni 1997; Vera and Simon 1993a; 1993b; 1993c) there are numerous examples showing that the symbolic based descriptions of experts are richer and more abstract than those of novices (Menzies 1998). Therefore, these descriptions must represent some interpretable meaning. However, what the strong SC argument shows is that weak SC based KBSs still suffer from an over simplification of the environment being represented.

It can, however, be argued that a middle ground could be taken. Given the weak SC argument that knowledge is constantly reinterpreted in the context of the problem, then it is possible to expand upon this notion by elaborating on '*what is context?*' Context in the Cyc, PCP, FCA and RDR have all assumed context to be *a priori* known and, therefore, identifiable, codifiable and static. For instance, when driving a car in Australia you should give way at an intersection to cars coming from the right. Likewise, cars to your left will give-way to you. However, if one of the cars to your left is an emergency vehicle with its siren blaring then you also give-way. Therefore, the different context in this case is the presence of the emergency vehicle. These two contexts are static, because the knowledge used in each is constant and does not change.

Strong SC, however, claims that the effect of context is overpowering, rendering such work as futile. For instance, long term weather prediction is inaccurate due to the existence of so many factors making up the forecast. Each

[6] *Merkwelt* is the term used by Jakob von Uexkull in his 1934 paper '*A Stroll through the Worlds of Animals and Men: A Picture Book of Invisible Worlds*', to refer to the complete set of environmental factors that have an affect on a species regardless of whether they are perceptible or not.

combination of these factors represents a different context. *Strong SC* does not, however, alter the underlying notion that context is an influence; just that it has a greater affect than weak SC claims. Nor does it adequately define context or detail why some form of symbolic representation does not aid learning to some degree. Thus, a middle ground reinterpretation of SC, referred to as *Intermediate Situated Cognition* in this thesis, should provide an elaboration on the types of context possible. Building on from the *weak SC* claim:

> ***Intermediate Situated Cognition*** *suggests that humans continually reinterpret a particular description of knowledge based on their current situation, which is dynamically alterable due to hidden contexts from within their Merkwelt.*

An interpretation of SC such as Intermediate Situation Cognition still allows for symbolic reasoning but indicates that it should be able to handle different reinterpretations of the same knowledge in the same context. For instance, a blackjack player's decision to draw a card when on seventeen is only partly based on the mathematical odds. Numerous factors from within their Merkwelt could also be affecting their choice. For example, they may think there are many smaller cards than usual left in the deck or they may simply have a *feeling*. In this situation the Merkwelt is being influenced by hidden emotional or subconscious states or contexts, which alters their response from previous times they have had the same hand. This outwardly produces what appears to be a dynamic contextual response, because a different action was taken for the same context, the hand dealt.

Some may claim, however, that dynamic context is represented in the contextually-static methodologies, through breaking down a drifting context into a series of discrete static representations. However, this is abstracting a level of complexity from a domain that could contain important information. It is also turning a domain into a *'blocks world'* type of toy problem. It is this over simplification that Strong SC truly opposes.

Furthermore, recognition of hidden and dynamic contexts is certainly not new. For instance, Arbib (1993) alludes to the existence of hidden contexts in schema theory, in that "… no single, central, logical representation of the world need link perception and action" (Arbib 1993, p 273). Widmer and Kubat's

(1996) FLORA system was designed with the idea that contexts may be hidden and unrealised.

One of the most widely researched areas of dynamic context is in the Information Filtering (IF) research stream, dealing with the issue of concept drift. Fundamentally, there is little difference between a user's concept and the context of their behaviour. The concept is the idea behind a user's action, whereas, the context represents the circumstances of an event. Thus, the user's concept governs the circumstances of the event, and therefore, the current context. Consequently, as the user's concept drifts, often dynamically, so does the context.

While proving the existence of such types of contexts is not the core of this thesis, the idea of these forms of contexts is one of the driving influences in the methodological design and in bridging the gap of the *process of practise* (chapter 1). The methodology developed in this thesis is designed at capturing these hidden contexts and goes some way to learning dynamic contexts. As shown in chapter 6, the success of the system to improve on the performance of a static methodology in many domains, goes some way to justifying that some form of contextual information is present.

2.3.1 Implications for Knowledge Engineering and Maintenance

The idea of introducing hidden and dynamic contexts into KBS research could potentially aid in understanding many of the historical and current problems in knowledge engineering and the maintenance of such systems. The fundamental issue, however, is how does a symbolic representation capture such information. Once a symbol is captured as a piece of knowledge, we not only are required to store a contextual relationship in parallel with that knowledge, we must also be able to alter it dynamically. Additionally, we should be able to find new contexts not expressed by an expert.

This thesis solves this problem by hybridising the symbolic representation with a function-fitting algorithm. The function-fitting algorithm learns the patterns of relationships between the partial symbolic representations of knowledge. Essentially working on Arbib's idea that "…the representation of the world is *the pattern of relationships between all its partial representations*" (Arbib 1993, p 273).

2.4 Summary

This chapter briefly outlined two philosophies of knowledge. The first, *physical symbol hypothesis* and its later, less ad hoc, form of *knowledge-level modelling* were discussed. The primary difficulty in these views was the conversion from contextually based to globally based knowledge and that these methodologies are inherently static in nature. Also discussed were a number of methodologies that use this interpretation of knowledge and how this presented further difficulties during knowledge acquisition and especially maintenance.

The emergence situated cognition (SC) view of knowledge addressed these issues, which was the second philosophy applied to AI discussed in this chapter. It was identified that knowledge representation methodologies need to be used that allow for dynamically changing knowledge. Ripple-Down Rules (chapter 3) is one methodology that has attempted to meet this SC view. Other methodologies that also use SC as their underlying philosophy, such as Personal Construct Psychology (PCP) and Formal Concept Analysis (FCA) were discussed.

These systems have performed exceptionally well in many areas where knowledge-level systems have repeatedly failed. Nevertheless, strong SC literature argues that such systems will not achieve true robustness and intelligence as context is too problematic to be represented symbolically. Finally, this thesis has suggested a clarification to what forms context can appear and suggest that for symbolic systems to succeed they must embrace all these contextual forms and that this meets the concerns of Strong SC. This redefinition of SC as it applies to AI, referred to as *Intermediate SC* is the primary philosophical influence behind the methodology in this thesis.

3

Ripple-Down Rules

> *If there's a buzzing-noise, somebody's making a buzzing-noise, and the only reason for making a buzzing-noise that I know of is because you're a bee. … And the only reason for being a bee that I know of is making honey. … And the only reason for making honey is so I can eat it.* *(Milne 1926, p 4)*

Discussed in the previous chapter was a small selection of methodologies, both mainstream and context-based approaches to solving the knowledge acquisition bottleneck and maintenance issues prevalent in knowledge based systems. Generally, the mainstream approaches have become fixated on the development of knowledge-level models with a knowledge engineer. So much so, that they have lost sight of the observed and frequently reported fact that users want ownership of their knowledge (Freidson 1994; Ignizio 1991; Kidd and Sharpe 1987; Richards 1998a; 2000a). RDR, FCA (2.2.4.1) and PCP (2.2.4.2), on the other hand, represent a paradigm shift in the approach to KA and KM through the development of new context-sensitive representations for knowledge.

Compton et al. (1988) extended Popper's (1963) theory of hypothetico deductive reasoning to the application of knowledge engineering. He suggested that experts do not provide information on their *insight* or how they reached a particular conclusion; but instead, they provide a *justification* for excluding the other possible hypotheses from within a particular context (Compton et al. 1988; 1989; Compton and Jansen 1988; Compton and Jansen 1990). It was suggested that these dichotomies between *insight* and *justification* (Compton et al. 1988; 1989) arise from a traditional misinterpretation of the form of knowledge provided by experts. This new hypothesis for knowledge engineering has resulted in the development and deployment of a new collection of methodologies and applications based on the knowledge acquisition and inferencing philosophy of Ripple-Down Rules (RDR).

After a discussion of the GARVAN-ES1 case study, which directly led to the development of RDR, this chapter will present a detailed description of the

basic RDR approach. Additionally, it will compare RDR with other methodologies and identify some of the recognised failings of RDR. Secondly, due to this thesis using Multiple Classification Ripple-Down Rules (MCRDR) as one of its base methodologies, it will discuss this primary adaptation of the basic philosophical ideas to the development of a multiple classification domain. These two KR structures are the basis for all RDR research and all the extension work presented in the final section of this chapter build on these fundamental methodologies.

3.1 Ripple-Down Rules Background

The original impetus for the development of Ripple-Down Rules came from Compton et al's (Compton et al. 1988; 1989) work on the GARVAN-ES1 expert system and the maintenance issues that arose. This section will discuss some issues with knowledge maintenance (KM) by briefly reviewing the GARVAN-ES1 system as a maintenance case study.

3.1.1 GARVAN-ES1: A Maintenance Case-Study

The traditionally developed GARVAN-ES1 medical expert system first came into regular use during 1984, providing clinical interpretations for thyroid hormone assay diagnostic reports. The domain for this system was ideal for an expert system to be applied, due to:

- The knowledge domain was bounded due to the expert using the same diagnostic test results that were also provided to the system.
- It did not require cooperation from an expert beyond what the expert would normally perform.

While these issues made it a viable expert system application in a real world domain, its true contribution to the further development of RDR, came from the system's idyllic maintenance requirements.

- All interpretations made by the system were checked by the expert – per their normal duties. Therefore, allowing the system to incrementally improve its knowledge base without increasing the burden on the experts.
- The system was never required to encompass new additional tasks.

- Changes tended to be only minor corrections or small additions to the knowledge base.

This system provided an example of the perfect maintenance situation for an expert system. It was shown, however, that it was still very difficult, highlighting how expert systems struggle with maintenance issues, even though maintenance is widely regarded as one of the most important aspects of expert systems. For instance, the GARVAN-ES1 knowledge base increased in size by approximately 80% in four years, improving its performance from 96% to 99.7% accuracy (Compton et al. 1988; 1989). Furthermore, many rules increased dramatically in complexity, as illustrated in the overall change, over three years, of rule 22310.01, shown in Figure 3-1. This highlights the difficulty a knowledge engineer would have in being able to comprehend and continue making adjustments to such a rule base in the future (for further information on the GARVAN-ES1 system, see (Compton et al. 1988; 1989; Horn et al. 1985).

More importantly though, was the realisation that the experts tended to provide justifications for changes by only identifying data that distinguishes the case presented from a small set of similarly likely concluding hypotheses (Compton et al. 1991; Compton et al. 1988; 1989; Compton and Jansen 1988; Compton and Jansen 1990). The primary difficulty for the knowledge engineer is then to fit the expert's justifications into the expert system, such that:

- The resulting rules can distinguish between *all* the possible hypotheses.
- The resulting rules maintain *consistency* for all the previously correctly classified cases.

1984

```
RULE(22310.01)
IF  (bhthy or utsh_bhft4 or vhthy)
    and not on_t4
    and not surgery
    and (antithroid or hyperthyroid)
THEN DIAGNOSIS("…thyrotoxicosis")
```

1987

```
RULE(22310.01)
IF  ((((T3 is missing)
    or(T3 is low and T3_BORD is low)
    and TSH is missing
    and vhthy
    and not (query_t4 or on_t4 or surgery or tumour
    or antithyroid or hypothyroid or hyperthyroid))
    or(
    (((utsh_bhft4 or
    (hithy and T3 is missing and TSH is missing))
    and (antithyroid or hyperthyroid))
    or
    utsh_vhft4
    or
    ((hithy or borthy)
    and T3 is missing
    and (TSH is undetect or TSH is low)))
    and
    not on_t4 and not (tumour or surgery)))
    and (TT4 isnt low or T4U isn't low)
THEN DIAGNOSIS("…thyrotoxicosis")
```

Figure 3-1: Shows the change in rule 22310.01 over a three year period, from 1984 when the system first went on-line, to 1987 (Compton et al. 1989).

As identified in chapter 2 it is this conversion between contextual knowledge and global knowledge, paralleled, with maintaining consistency that renders long term maintainability next to impossible in traditional expert systems. Therefore, in order to create maintainable ES through dialogue with domain experts, the system should build into its methodology a means for incorporating an expert's justification of a hypothesis within a particular context. The conclusions identified for both KA and KM are essentially identical, both have the potential to be significantly reduced through the inclusion of context within the KR used by the system.

3.1.2 Implications

Compton et al. (1988; 1989), through this maintenance experience, found that in order to create a maintainable ES through dialogue with domain experts, then the system should build into its methodology a means for incorporating an expert's justification of a hypothesis within a particular context. Based on these observations, Compton et al. (1988; 1989), identified three approaches to developing context sensitive knowledge bases:

- To set up an ES, such that, the context of a provided justification forms part of the knowledge base.
- To incorporate the notion of debate or argument, following from Popper's hypothesis of knowledge, within the ES.
- Finally, is to use inductive techniques to learn from datasets.

A KR that incorporates some or all of these techniques is attractive because they implicitly recognise that an expert provides the best source of knowledge. Inductive learning methods find features in a dataset which best distinguish between hypotheses, which have a strong correlation with an expert's justification of a hypothesis within a context (Compton et al. 1988; 1989). While many systems have been developed for inductive learning, such as C4.5 (Quinlan 1993), concepts for incorporating context, are not central components.

3.2 Ripple-Down Rules Methodology

Ripple-Down Rules[7] was first proposed by Compton and Jansen (1988). It uses a simple exception structure aimed at partially capturing the context that knowledge is obtained from an expert. It was assumed that the context was the sequence of rules that had evaluated to provide the given conclusion (Beydoun 2000; Compton et al. 1991; Compton and Jansen 1988; Compton et al. 1993; Preston et al. 1993; Preston et al. 1994). Therefore, if the expert disagrees with a conclusion made by the system they can change it by adding a new rule with whichever level of generality was required by the expert. However, the new rule will only fire if the same path of rules is evaluated with the same outcomes as previously (Compton et al. 1993). Fundamentally, since its development it has accurately been described as knowledge acquisition through *fault patching* (Menzies and Debenham 2000). Ripple-Down Rules was so named to capture this notion of appending *fix-up* rules to the bottom of a knowledge base, thereby, rippling the formula for a conclusion down over many rules, rather than attempting to fix existing rules (Jansen and Compton 1989).

The context of a new rule is its position within the RDR structure. A rule can only fire when the rules that led to it, before its creation, have fired. Therefore, the rules context is the previous path of rules and their outcomes. This is effectively capturing the rules context implicitly within the structure itself. This corresponds to how a human expert, explaining the justification of their conclusion to a trainee, will start from where they believe the trainee deviated from the correct solution, providing the justifications required from that point to the expert's correct conclusion. Furthermore, it also corresponds to a person's method of debate or argument. When debating an issue each person presents their view but they start their justifications from the current context of their opponents.

[7] Since the advent of Multiple Classification RDR (3.3), RDR has sometimes been referred to as Single Classification RDR (or SCRDR) or Simple RDR (XRDR) (Le Gia et al., 1997) to differentiate it from MCRDR. RDR is often used to refer to the whole collection of RDR-based methodologies, which can cause some confusion. It is also commonly regarded in the RDR community that RDR refers to the single classification version. Thus, throughout this thesis, the acronym RDR will always be used to refer to the single classification version and when a reference to the collection of RDR-based methodologies is made, it will be clear from the context of the sentence.

RDR's very simple *no-model* approach to KA has revealed many advantages. For instance, because a rule can only fire within a particular context, then when it is being created it only requires validation within that context (3.2.1). Therefore, new knowledge can easily be validated during the moment of acquisition. Furthermore, the expert does not need to know anything about the actual underlying structure of the knowledge they are constructing. This is a marked difference from model based approaches where the KE needs to consider and validate each new piece of knowledge across the entire knowledge base. In this system the normal KB development phases of requirements definition, analysis, design, implementation and maintenance can be eliminated. RDR omits the design stage completely and merges the implementation and maintenance phases seamlessly (Beydoun 2000), effectively providing a holistic approach to KA and KM where there is little or no difference between the tasks.

3.2.1 Structure and Inference

The basic underlying structure of RDR[8] is a binary tree; each node contains a rule and a conclusion (or classification). Each node has two possible branches, representing whether the rule was satisfied or not by the data provided in the presented case. If a rule is found to be *'true'* then the successful branch[9], is followed to the next rule and visa versa for the unsatisfied branch[10]. This process continues until either a leaf node is reached or until a node has no appropriate branch to follow[11]. The conclusion returned by RDR is the conclusion from the last *successful* rule. That is, if the last node tested was *true* then its conclusion is returned. However, if it was *false* then its parent-node's conclusion (if its rule was satisfied) is returned. This process of checking the parents, is continued until a satisfied rule is found, thereby returning that rule's

[8] RDR has evolved into a general representation since its initial years with the GARVAN-ES1 system where it was first applied with alternative representation strategies. It is this generalised form that will be presented in this chapter rather than the earlier incarnations, as this is the generally-accepted basic methodology that all other work has been based upon.

[9] The *'true branch'* or *'successful branch'*, is also sometimes referred to as the *'except branch'*, as the child rule can be thought of as an *exception* to the parent rule.

[10] The *'false branch'* or *'unsuccessful branch'*, is also sometimes referred to as the *'if-not branch'*.

[11] The majority of the current literature states that inference terminates only at a leaf node. This is inferring that RDR is using *'dummy'* leaf-nodes that are added at the alternate branch when the user corrects a conclusion. Yet no diagrams or explanations explicitly represent or state any such type of node as being part of the methodology. Thus, this thesis is correcting this anomaly in stating that inferencing is concluded at a leaf node or *when no branch exists to follow*, and therefore, there is no need for the inclusion of *'dummy'* leaf nodes.

conclusion. The root node is a special case, of the form *'If true then – default conclusion'*, which is always satisfied. Therefore, if no other rule is satisfied then the default conclusion is returned, guaranteeing a conclusion will always be found.

For example, a case with the attributes *{a, b, c, g, h}* is presented to the RDR KB shown in Figure 3-2. In this tree it can be seen that: rules *1* and *3* have both true and false branches leading to further rules; rule *2* only has a false path; rules *4* and *6* (and obviously the root node) only have a true path; and, rules *5*, *7* and *8* are leaf nodes. When the case is presented it ripples down the tree using the path *{0 – 1 – 3 – 6}* where, because there is no attribute '*f*' and no *false* branch, the inferencing process completes. The conclusion returned is *1* from rule *1*, due to this being the last rule to be satisfied.

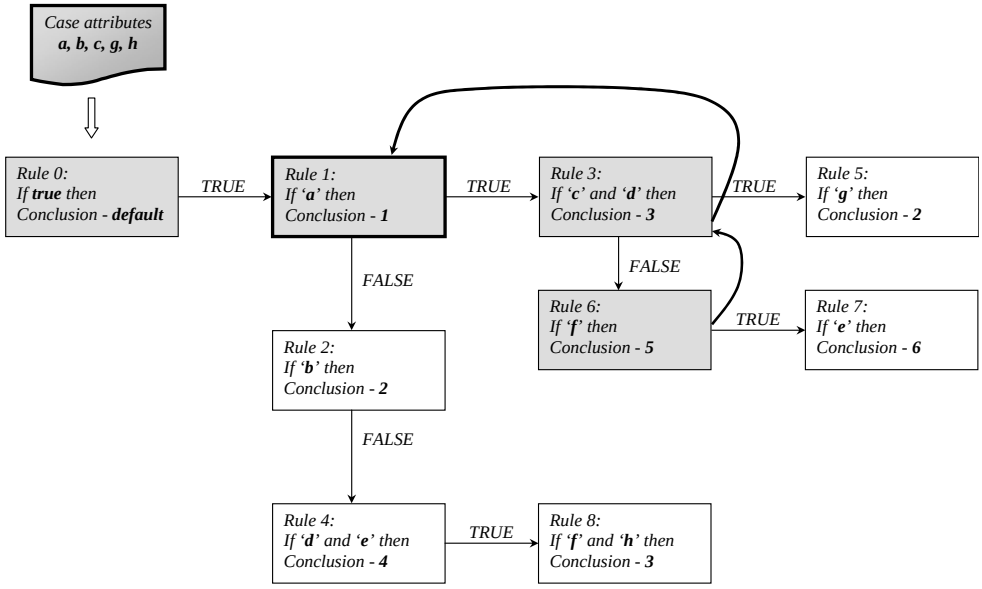


Figure 3-2: Example of the RDR binary tree structure. The rectangle nodes of the tree contain a rule and a classification. The branches of the tree are either 'true' or 'false'. The straight arrows indicate the direction of inference. The empty block arrow and the document shape indicate a case being presented to the KB for inferencing. The grey filled boxes show the rules that are tested during the inferencing process for the example case. The curved arrows show the direction taken when searching for the last satisfied rule and the bold outline rectangle shows that last satisfied rule. Thus, the final conclusion of the inferencing process is class '1'.

It should be observed in the above example that even though our case had the attributes '*b*', '*g*' and '*h*' and that the RDR tree has rules *2*, *5* and *8* that test for these attributes, they were never used, because the context that those attributes are required in, never occurs. It should also be noted that the rule numbers have no meaning in the inferencing process and if used in implementation generally indicate the order rules are created. They are shown here only so particular rules can be identified and discussed. Also, it can also be seen that some conclusions are repeated at different locations. This is usually the case as most classifications have multiple possible definitions.

3.2.2 Learning

As mentioned previously, the primary advantage of RDR is the ease that new knowledge can be added and the method of adding knowledge is the same, regardless of whether the system is in the development or maintenance stages. The inclusion of new knowledge is simple because of the context based structure of the rules. When a new rule is being added it is merely appended to the tree as a leaf node and it only needs to be validated within this narrow contextual domain.

RDR learns through the acquisition of rules, increasing the size of the KB. The process consists of first passing the case to the inferencing system (3.2.1) which classifies the case. If the expert disagrees with the conclusion then they simply provide a justification for why it was wrong, which is used as the new rule. The justification is determined by first comparing the current case with the previous case that had originally created the parent rule, referred to as the *cornerstone case*. A list of differences, in the form of attributes, between these two cases is generated from which the expert selects one or more. Those attributes selected justify the new rule. The new rule created is unable to mistakenly classify the old cornerstone case with the new rule because only differences were used in creating the rule. This concept of using differences is not new and has been used in KA methodologies based on PCP. It's a useful acquisition technique as people are good at identifying differences (Compton et al. 1991; Gaines and Shaw 1990; Shaw and Gaines 1992). The RDR approach contrasts these other PCP based methods, because, rather than asking the expert

to *think* of differences, RDR simply asks them to *select* the relevant differences (Compton et al. 1991).

For example, continuing from the previous example in Figure 3-2, Figure 3-3 illustrates how a new rule is created and added to the RDR structure. It is assumed that the above inference has occurred and the expert has decided the conclusion of class 1 is incorrect. Firstly, the cornerstone case that was used in the creation of rule 6 (this rule is used as it was the last one visited during the inference process) is loaded and compared with our new case identified as having been misclassified. The two cases are merged and the unique attributes extracted, as identified in the difference list. The expert then selects the relevant differences that best describes the difference between the documents, for instance '*h*' and '*!i*' has been selected in this example. The new node is then created by writing a rule with the attributes selected, the correct class given by the expert and our current case. The current case will become the cornerstone case for this new rule ready for future corrections. The new rule is then attached as a child node to rule 6 on the false branch.

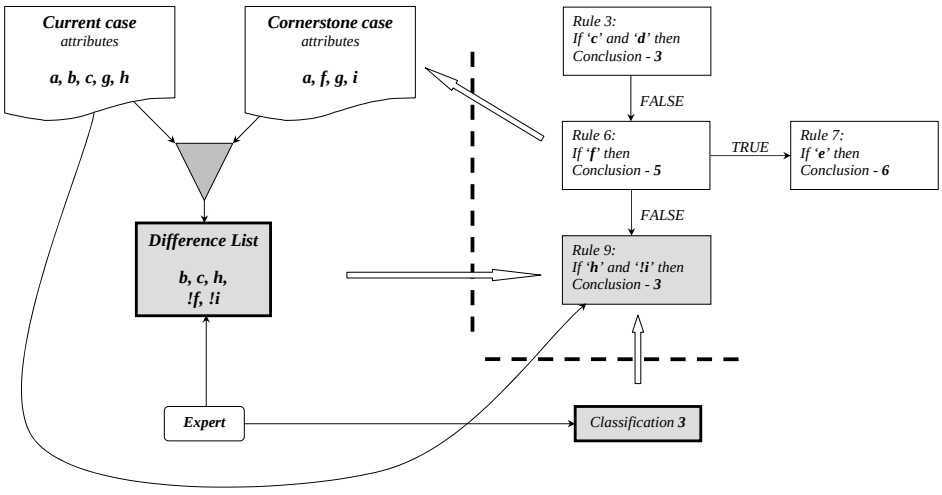


Figure 3-3: Example of creating and incorporating new knowledge in RDR. The grey triangle represents the merging of the cornerstone case (extracted from the terminating rule during inferencing) and the current case, producing a list of attributes that differentiate the two cases, shown in the grey box with a bold border. The expert selects from this list the attributes that best identifies the reason for the conclusion. A new leaf node is created incorporating a rule, compiled from the expert's selections, the classification identified by the expert, and the current case which will become the new node's cornerstone case. This node is then added to the tree at the point where the inferencing process was halted.

As this example shows, through the inclusion of context within the structure of the KB, two major advantages are achieved. Firstly, the cause of the failure by the KBS during inferencing is automatically determined. This significantly improves on existing KA paradigms where a KE will have to study the rule base to locate the cause of an error even before looking at how it is to be corrected. Secondly, when creating the new rule it is guaranteed to be consistent with the existing KB without the need of extensive testing of the new rule (Beydoun 2000).

Another advantage to this approach to incremental rule creation is that the task remains the same regardless of the size of the KB. Each KA step has the same simplistic process of selecting differences (Mansuri et al. 1991b) whether it is the first rule or the ten thousandth rule created. Finally, it should also be noted that this method of fault patching means there is no need to delete or change rules once entered. In fact such changes are explicitly excluded from the methodology as they can hamper the contextual validation of the system. Instead, errors in the KB can simply be corrected by adding a new rule rectifying the earlier incorrect assertion.

3.2.3 Comparison of RDR with Other KBS Methodologies

The fundamental difference between RDR and the vast majority of other KBSs is its philosophical approach. Menzies (1998), when considering a conceived KB to be built and used, puts it rather simply:

> If most of the changes to that knowledge base occur before its usage, then we would look to optimizing the design process: i.e. conception to construction. … [However,] if most of the changes to that knowledge base occur once it is being used, [then] we should look to optimizing the maintenance process: usage to reconception to reconstruction (Menzies 1998, p 867).

It is this different situated cogitative view adopted by RDR which separates it from the mainstream systems. As discussed in section 2.1.4.5, it is SC's view that knowledge is created at the moment it is articulated, thus, it is highly contextually dependant and constantly changing. Therefore, the design-time modelling methodologies (such as KADS) cannot easily update their knowledge in this changing environment. RDR concentrates on optimising the maintenance

process rather than attempting to design and model solutions prior to KB development.

In achieving a maintenance oriented technique, RDR had to be designed to perform validation of knowledge at the moment of acquisition. As mentioned previously, this is achieved by limiting the expert to only identify domain knowledge within the context of the previous error. This removal of structural and control knowledge from the knowledge base, traditionally hidden within the KB, means that only domain expertise is still required. A traditional KE task is to incorporate knowledge into a KB by making structural and control decisions. KADS and numerous other methodologies directly refer to these knowledge types and offer methods of extracting and modelling such knowledge. Yet RDR is able to remove the need for such knowledge entirely, effectively eliminating the need for the KE. One interesting observation, resulting from the removal of the KE in KBS development, is that RDR can be applied in a significantly more diverse range of classification based applications. As will be discussed in section 3.4, while RDR can be applied in traditional KBS domains like medical diagnosis, it can also be used in domains such as personalisation tools, monitoring and management.

It is argued that one advantage of knowledge level methodologies is the opportunity for knowledge reuse through the use of problem solving methods or ontologies. Systems such as Protégé have targeted such knowledge reuse (Gennari et al. 2002). Knowledge reuse, however, was not an aim of the RDR methodology and its use of raw contextually based, often specialised, knowledge is a major draw back in the RDR technique from a reuse perspective. However, extension work to RDR, by Richards (1998a), has opened up the possibilities of knowledge reuse. It is this specialised, non-generalised, knowledge that makes RDR a powerful tool. The generalisation of knowledge in traditional approaches, by definition, is removing knowledge. This removed knowledge is in the form of contextual knowledge and some degree of domain knowledge. However, this lack of generalisation leads to the greatest criticism of the RDR approach by the proponents of the knowledge level methodologies.

3.2.4 Problems with RDR

Without a doubt, the greatest criticism of RDR is the very use of specialised contextual knowledge that gives RDR much of its strength. The obvious problem is that knowledge can easily be repeated in many places throughout the RDR structure (Compton et al. 1993; Kang et al. 1995; Mansuri et al. 1991b; Preston et al. 1994). This repeated knowledge is not only in the form of individual rules but potentially entire sections of a tree. A second problem is due to the lack of control over the structure of the knowledge. Tests have shown that the RDR tree can become extremely unbalanced (Compton et al. 1991; Preston et al. 1993; Preston et al. 1994).

By observing expert behaviour when creating rules, it has become clear that neither of these are major problems and certainly do not vitiate the method (Compton et al. 1993). For instance, when the GARVAN system was rebuilt using RDR it was found that only 13% of the knowledge was repeated and because KA is significantly faster than in traditional systems the small amount of repetitive KA required is not seen as being a major problem (Op cit). Comparisons with induction methods such as Quinlan's C4.5 (1993) and Gaines's InductRDR (1989a) suggest that the manually constructed RDR tree only has approximately twice as many rules, which is remarkably good against the highly optimised induction trees (Compton et al. 1993).

The reason repeated knowledge does not occur in large quantities is believed to be related to the second problem. Experts appear to mainly create rules when the system fails to give a conclusion (other than the default conclusion), rather than to correct inconsistencies (Compton et al. 1991; Compton et al. 1994; Mansuri et al. 1991a; Mansuri et al. 1991b). This is believed to be due to experts tending to provide fairly accurate rules for a particular case, leaving a range of possible cases not covered (Compton et al. 1991; Mansuri et al. 1991b). This causes the unbalanced tree but also means that there is less chance of the expert needing to repeat knowledge. Work has been carried out to attempt to address these issues: such as Chellen's (1995) system for reducing the size of a manually developed RDR system; Richards's (1998a) use of rough sets and induction; and, most recently Suryanto and Compton's (2003) use of predicate logic. For further details on methods used to reduce repetition see section 3.4.3.

A third significant problem with the basic RDR methodology is its inability to handle multiple classification domains. This is not unique to RDR as it is shared with many induction methods, such as Quinlan's C4.5 (1993). There are two possible work arounds with the basic RDR structure as it stands. The first is through the use of combinatorial classifications with each single classification provided, which vastly exacerbates the repetition problem (Compton et al. 1993), as has been found in the PEIRS KBS (Edwards et al. 1995a). A second possibility is using multiple KBs, which require a predetermination of how many KBs are required and can also result in significant repetition depending on the domain and runs contrary to the RDR philosophy. This later approach was trialled successfully by Shiraz and Sammut (1998) with a system referred to as Dynamic RDR (DRDR) applied to learning a flight control task. The alternative to these approaches is to redesign the structure to allow for multiple classifications, discussed in detail in the following section.

3.3 Multiple Classification Ripple-Down Rules Methodology

Ripple-Down Rules has been shown to be a highly effective tool for KA and KM. However, its lack of ability to handle tasks with multiple possible conclusions significantly limits the method's ability to be applied in many domains. The aim of Multiple Classification Ripple-Down Rules (MCRDR) was to redevelop the RDR methodology to provide a general approach to building and maintaining a KB for multiple classification domains, with all the advantages found in RDR. Such a system would be able to add fully validated knowledge in a simple incremental contextually dependant manner without the need of a KE (Kang 1996; Kang et al. 1995).

3.3.1 Structure and Inference

The new methodology developed by Kang (1996) is based on the proposed solution by Compton et al (Compton et al. 1991; Compton et al. 1993). The primary shift was to switch from the binary tree to an *n-ary* tree representation. The context is still captured within the structure of the KB and explanation can still be derived from the path followed to the concluding node. The main difference between the systems is that RDR has both an *exception* (true) branch

and an *if-not* (false) branch, whereas MCRDR only has exception branches. The false branch instead simply cancels a path of evaluation. Like with RDR, MCRDR nodes each contain a rule and a conclusion if the rule is satisfied. Each, however, can have any number of child branches and each one represents an exception branch.

Inference occurs by first evaluating the root and then moving down level by level. This continues until either a leaf node is reached or until none of the child rules evaluate to *true.* Each node tests the given case against its rule. If *false* it simply returns, *X* (no classification). This is essentially the same as following the *false* branch in RDR. However, if this node's rule evaluates to *true* then it will pass the case to all the child nodes. Each child, if *true*, will then return a list of classifications. Each list of classifications is then collated with those sent back from the other children and returned. However, if none of the children evaluate to *true*, and thus they all return *X*, then this node will instead return its classification. Like with RDR the root node's rule always evaluates to *true*, ensuring that if no other classification is found then a default classification will be returned.

For example, a case with the attributes *{a, c, d, e, g, i}* is presented to the MCRDR KB shown in Figure 3-4. The case ripples down through two child

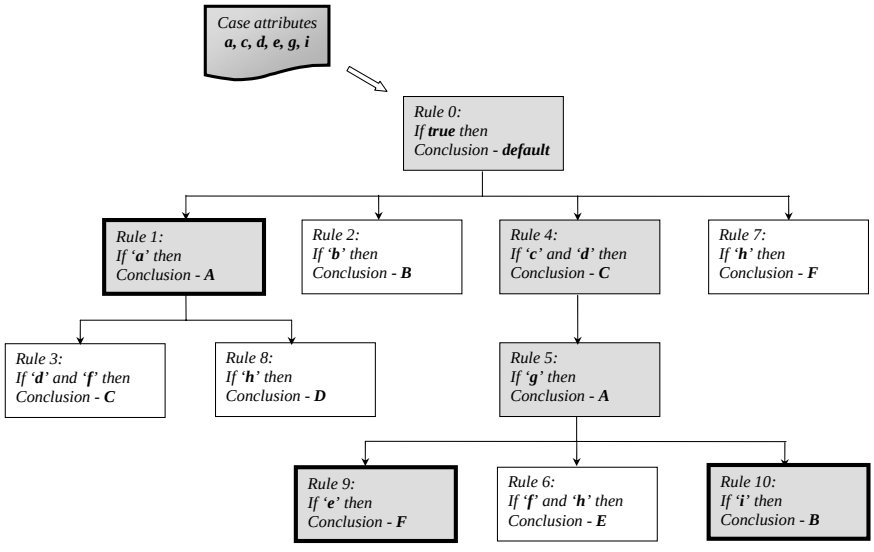


Figure 3-4: Example of the MCRDR n-ary tree structure. The rectangle nodes of the tree contain a rule and a classification. All branches are essentially exception (true) branches. The straight arrows indicate the direction of possible inference. The grey filled boxes show the rules that are tested and evaluate to true for the identified case. The rectangles with bold borders are the terminating rules.

rules, *1* and *4*, from the root node as these were the only nodes with rules evaluating to *true*. The path followed to rule *1* is, however, immediately halted as all its children return, *X*. The path continuing from rule *4* passes through rule 5 and concludes at rules *9* and *10*. These three concluding rules, *1*, *9* and *10* return our conclusions *A*, *F* and *B*. The conclusion of *A*, in this example, however, is redundant because the inference process for classifications *F* and *B* had the rule paths *{0 – 4 – 5 – 9}* and *{0 – 4 – 5 – 10}*. Both paths past through rule *5*, and due to rule *5* having the same classification as rule *1*, and because an exception was found to rule *5*, then that exception is also an exception to rule *1*. Thus, the final conclusion is F and B.

3.3.2 Learning

Knowledge acquisition in MCRDR occurs for the same situations as with RDR, when a case is misclassified or when no classification other than the default is found. Once an error is found the MCRDR procedure for refinement is mildly more complicated than the approach used in RDR. This complexity comes from enforcing the need for validation during the KA task. Kang et al (1995) describe a three step process:

1. *The expert supplies the correct classification.* This is a trivial step and was also required in RDR. It is simply the process of the expert informing the system of the correct classification for the case presented
2. *The system identifies the new rule's location.* This is a new step and will be discussed in the following section (3.3.2.1)
3. *The expert defines the new rule from a series of difference lists.* This step was also in RDR but has been expanded (3.3.2.2).

3.3.2.1 Locating Rules

In the RDR methodology each new node was simply attached to the appropriate node at the bottom of the tree where inference had halted. However, in MCRDR there can be multiple nodes attached to a parent node, thus a new node should be able to be attached at any location in the tree hierarchy, between the root and the terminating inference node. Kang et al. (1995) identifies three reasons for

Wrong Classification	Method to Correct the KB
A new independent classification	Add the new rule at an appropriate higher level so that it does not cancel the already correct classification.
A classification is wrong and should be replaced by another classification	Add a rule at the end of the path to give the new classification. This cancels the old incorrect classification
Classification is wrong and should simply be stopped	Add a new stopping rule at the end of the path to prevent the classification

Table 3-1: Three ways for adding new rules when correcting an MCRDR KB (Kang et al. 1995).

wishing to correct the knowledge base and identifies recommendations for what types of rules should be created and where they should be placed, shown in Table 3-1. The middle situation is identical to RDR's reason for creating a new rule. However, the other two are unique to MCRDR. The first circumstance allows for the possibility of increasing the number of classifications of a case. The final situation introduces a special rule, referred to as a stopping rule, which is used to halt and cancel a particular path of evaluation. The stopping rule plays an important role in MCRDR by preventing incorrect classifications being given for a presented case.

It can be seen here that a decision must be made as to whether the new rule being created is a refinement (or exception) to an existing rule, or whether it is an entirely new classification. If it is new then it should be added further up the tree, after the last relevant node. The last relevant node is the final one in the original evaluation path that the new rule could be considered to be offering a refinement. However, in some ways it does not matter as any location between the root and the terminating node will still work regardless of the expert's view of the new rule. The primary effect of location is on the evolution and maintenance of the KB. If the chosen rule creation strategy tends to place new rules towards the end of inference paths, then the domain coverage will be slow but each rule will generate fewer errors because they are seen less frequently. If the strategy used tends to position new rules higher in the path then the KB structure will be flatter and broader, meaning more rules are in context at any

time. This will result in the faster coverage of the entire domain but each rule will generate more errors, because the use of context is not as great (Kang 1996).

While these added decisions may imply the need to return to employing a KE, research has shown that through using various interface strategies, the RDR philosophy of the expert having no knowledge of the underlying structure, can remain. For instance, Kang et al (1995) offers a number of possible interaction methods that can lead towards either a more general wide ranging KB or a more highly specified KB. These outcomes can be achieved by asking the expert to select the important data used to reach the new conclusion, or alternatively, to delete the irrelevant data. Then the selected or remaining data is used to create a *mini-case*[12], which is processed by the tree to find how far down the identified path the new rule should be placed. This process is effectively searching for the correct context to place the rule.

3.3.2.2 Rule Creation

The idea in incremental validation is that the new rule does not cause any previously seen correctly-classified cases to now be misclassified after the new rule is added. Generally, validation is performed by maintaining a database of previous cases. In RDR this is simply done by testing the new case against the cornerstone case that was stored at the time of the parent rules creation, thus there is only the one previous case stored in the database. In MCRDR this notion of a cornerstone case is retained, however, it has been extended due to the possibility of relevant rules being created elsewhere in the tree. Each node in the MCRDR tree can contain a number of cornerstone cases. There are two situations when a case can be added as a cornerstone case:

- Firstly, as in RDR, when a new rule is created, the case that was incorrectly classified and caused the new rule is added as the first cornerstone case of the new node.
- Secondly, if a case is correctly classified at a particular node but incorrectly classified simultaneously at another node causing a new rule to

[12] A *mini-case* is a case that only has a selection of attributes of the actual case that caused a misclassification. Thus, if a *mini-case* is represented by ‘c’, then $c \subseteq C$.

be created elsewhere in the tree. The case is added to the newly created rule, (per the first point) and also to the node that classified it correctly.

The second point is performed because it is known that this is a correctly represented case by the KB. The system knows this because the expert has taken the time to correct a rule. Cases that the expert has simply accepted as correctly classified are not stored, because they could be wrong. This would occur when the expert had little concern about the error and had not taken the time to correct them. Thus, they are never used for validation.

A second complexity in the multiple classification knowledge acquisition methodology, is that with the possibility of multiple children, the system would be required to also validate any rule being created against not only all the cornerstone cases at the parent node, but also all the cornerstone cases at all the sub branches from the parent node. Figure 3-5, illustrates which nodes' cornerstone cases would need to be checked for validation. It can be seen that the higher up the tree a correction is being made the more cornerstone cases from which the current case will be required to be differentiated.

To create a new rule a difference list will need to be generated between the current case and all the relevant cornerstone cases. Equation (3-1) shows an example of what needs to be accomplished. In this example, *case A* is the

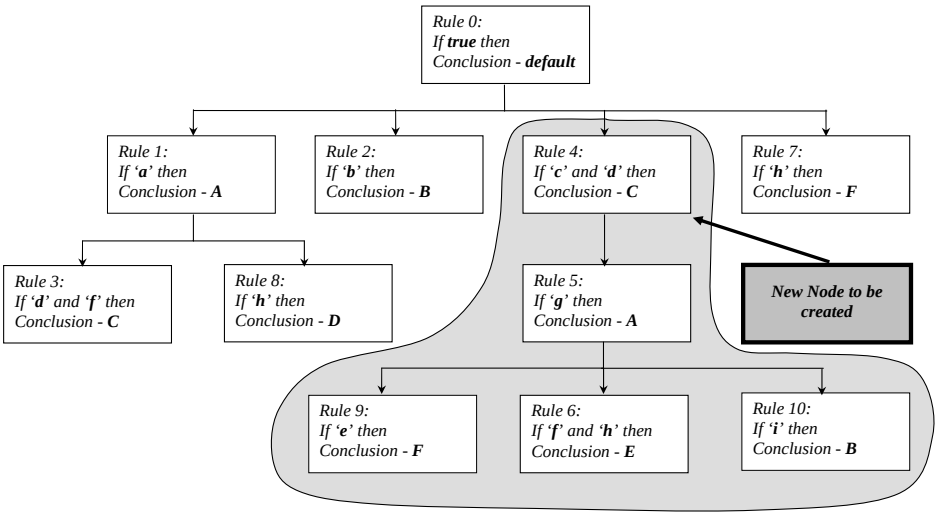
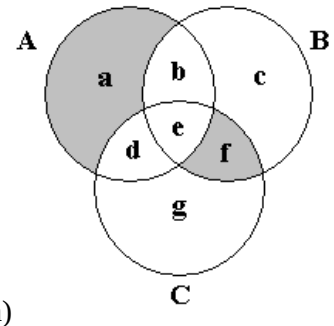


Figure 3-5: MCRDR tree showing the nodes that cornerstone cases should be taken from when a new node is added to the node containing Rule 4. This is the same tree as in Figure 3-4. The new node to be created, when added, will have the rule 4 node as a parent. Cornerstone cases should be drawn from all the nodes below the parent node in the tree, shown in the grey section. Thus, they will be taken from the nodes with rules 4, 5, 6, 9 and 10.

current case needing to be differentiated from two cornerstone cases, *case B* and *case C*.

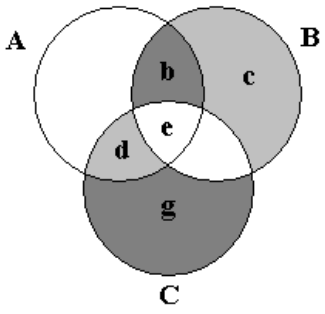
$$\begin{array}{c} case\ A - (case\ B \cup case\ C) \\ or \\ negated\ conditions\ \ from\ ((case\ B \cap case\ C) - case\ A) \end{array} \qquad (3\text{-}1)$$

However, finding an attribute in the new case that is not in any of the cornerstone cases is often not possible, as illustrated in Figure 3-6. Kang (1996; Kang et al. 1995) argues that the exclusivity of the attributes themselves is, however, not what is important, only the uniqueness of the *attribute combination* is required. To extract such a combination, Kang suggests a simple incremental approach to rule construction, where a series of difference lists are



(a)

Difference List between A and (B and C)	*a not f*



(b)

Difference List between A and (B and C)	*empty*
Difference List between A and B	*d not c*
Difference List between (A and C)	*b not g*

Figure 3-6: Comparison of two rule selection methods using two similar situations. In both situations: case A is the current case requiring a new rule; and, cases B and C are both cornerstone cases. In situation (a) the simple method is able to find the attributes {a, !f} separating the new case from the two cornerstone cases, identified by the light grey areas. In situation (b) there are no attributes in the locations used in situation (a), therefore, the simple method fails to find any attributes separating A from B and C. However, the second method finds {d, !c} differentiating A from B, identified by the light grey areas, and {b, !c} separating A from C, shown in dark grey. Thus, case A can be uniquely identified through the attribute combination of {b, d, !c, !g}. Note: in this example all possible attributes were selected, the expert, however, would only need to select a minimum of 1 from each difference list (taken from Kang et al. 1995).

generated. Therefore, in the example of Equation (3-1, *A* is first compared against *B*, and then compared against *C*. Figure 3-6, shows how, in certain situations, this can find a rule where the simpler approach could not.

This method appears rather time consuming for the expert, as they potentially would have to select differences in hundreds of comparisons. However, in practice this is not the case. When the expert selects an attribute from the first difference list all the other cornerstone cases are checked against this partial rule. All cornerstone cases not satisfied are removed from the list of cornerstone cases. Then the case is compared against one of the remaining cases. This process of eliminating cornerstone cases significantly reduces the amount of comparisons required and only very rarely does an expert need to make more than 3 or 4 comparisons (Kang 1996; Kang et al. 1995; Kang et al. 1998).

3.4 Refinements and Extensions

The methodologies described were previously covered in detail due to their importance as the two central algorithms at the core of all RDR based research. There have been a number of attempts to enhance these original RDR methodologies. These enhancements, however, rarely can be applied universally, as they usually only have a narrow field of application. One of the main areas of research has been in the development of methods for improving knowledge acquisition through using induction or machine learning algorithms usually with significant success. Other areas that have also been investigated are methods to reuse knowledge already captured and the development of methods to create fuzzy rules. There has also been some work on finding ways of reducing the problems of knowledge repetition in RDR and how to resolve conflicting knowledge when acquiring knowledge from multiple experts.

This section aims to briefly review two of the major extensions: Richards's MCRDR/FCA and Suryanto's GENICA. These extensions are related to the work in this thesis as they both, through different approaches, attempted to extract information from the RDR based structure. Additionally, a very brief survey of a number of refinements and extensions are described. Further details on RDR research into prudence analysis will be reviewed in chapter 7.

3.4.1 MCRDR/FCA

A purely automated approach to the extraction of concepts for the development of a concept hierarchy is taken by Richards and Compton (1997b; 1999) in the MCRDR/FCA system. This is the first major body of work to investigate the extraction of information from the MCRDR KB structure itself. The primary focus of this work is the discovery of generalised concepts for knowledge reuse. However, it has also been used as a means to develop a concept model, improving RDR explanation, as a tool for construction tasks and for building Problem Solving Methods (PSMs) and Ontologies.

MCRDR/FCA is one of the most significant contributions to knowledge reuse with an RDR based philosophy. It stemmed from Richards's (1998a) PhD thesis, with various other papers by Richards (1998b; 1999), Richards and Compton (1997a; 1997b; 1999), Richards and Busch (2000; 2001) and Richards and Simoff (2001). Richards used FCA to create a formal method for identifying the concepts and the relationships between those concepts, captured by MCRDR. It accomplished this through the construction of abstraction hierarchies from the raw production rules. This information was then used to extract further concepts with minimal additional effort for the expert (Richards 1998a; Richards and Compton 1997a). This valuable contribution showed that the complex modelling of domain and problem solving knowledge of main stream KBSs prior to KA commencing, is not required. Later research into MCRDR/FCAs application into modelling tacit knowledge (Busch and Richards 2004; Richards and Busch 2000; 2001) has also shown the method's conceptualisation power and ability to capture knowledge that other KBSs cannot accomplish.

Other uses for MCRDR/FCA have also been explored. For instance, the natural opportunity for explanation within the RDR methodology was extended (Richards 1998a; 2001) to include the retrospective analysis of the domain knowledge that has been captured in the RDR rules. This is achieved through the use of FCA (2.2.4.1), which produces a retrospective model allowing the expert to easily peruse the interlinked concept lattice generated from the rule base. The combination of this feature with the ability to review the rule traces and case differences allows for a much deeper contextually based explanation. The use of FCA line diagrams in Richards's work is similar to the connected

graphs which Clancey (1986) refers to as a casual model. Additionally, Richards (1998a), Richards and Compton (1999), Richards and Simoff (2001), and more recently Richards (2004) have used MCRDR/FCA for both PSM and ontological construction through incremental retrospective reverse engineering.

3.4.2 GENICA

More recent work deriving and restructuring knowledge in RDR is by Suryanto and Compton (2002; 2003; 2004) and in Suryanto's PhD thesis (2004). This work stems from first flattening the MC/RDR trees into a set of disjunctive normal form rules (*DNF rules*), which can then be run in any order. These *DNF rules* are then used through three phases, *reorganization of RDR*, *generalization on RDR* and *discovery from RDR*. The first process of *reorganisation of RDR* rules is performed during the rebuilding of the tree, by removing flat rules that have been subsumed by other flat rules. This is effectively removing any duplicated sections of the tree. The *generalisation on RDR* is performed using GENICA (*GEN*eralization using *I*ntra *C*onstruction and *A*bsorption), based on Duce (Muggleton 1987), which generates generalised rules through invented predicates from expert created rules. This has the advantage of reducing KA effort in certain situations by up to 50% (Suryanto 2004; Suryanto and Compton 2003; 2004).

Suryanto's *discovery from RDR* investigates the detection of ontologies, referred to as *ontology learning* (OL). Relevant sections of this work have also been published by Suryanto and Compton (2000a; 2000b; 2001; 2002). The goal of OL is to discover relations between classes from the multiple conclusions of MCRDR, which is similar conceptually to the work in this thesis. Suryanto's system will find one of three possible relationships, which are mutual-exclusivity, similarity and subsumption. These relationships are found by observing commonalities and differences between the conditions used in rules in the rule path (Suryanto 2004). These relations are then used to construct compound relations, which are used to extract matching relations from all the basic relations. Finally these matched compound relations are used to create a class model, which Suryanto asserts, "…then enables possible ontologies to be explored on a reasonable scale" (Suryanto 2004, p 136).

3.4.3 Other RDR Research

There has been a significant amount of research in RDR based systems, thus this section will only very briefly mention the more important bodies of work. Validation and Verification research, along with work on prudence analysis is covered separately in chapter 7.

3.4.3.1 Enhancements

In section 3.1.2 it was mentioned that one way of developing context sensitive knowledge bases was to use induction, such as Quinlan's C4.5 (1987; 1993). Induction, the process of automatically deriving knowledge through statistical analyses or machine learning, however, is not part of the basic RDR methodologies. Compton et al (1988; 1989) originally claimed that the merging of an RDR-like system with induction techniques could be advantageous. There have been two fully inductive methods developed for RDR. The first, and most widely used, is InductRDR (Gaines and Compton 1992; 1993) based on the Induct algorithm by Gaines (1989b; 1991), which used induction with exceptions. Induct, is similar to C4.5, except it derives rules directly using an extension of the Prism algorithm proposed by Cendrowska (1987), in order to extract knowledge from noisy data. The second full induction algorithm is Scheffer's (1996) Cut95, which has shown good empirical results.

There have also been a number of methods developed to merge inductive learning and knowledge acquisition from a human expert. These partially inductive techniques move closer to a complete solution for context sensitive knowledge bases as they incorporate all of Compton et al's (1989) three possible approaches. One of the early attempts was by Catlett (1992). Catlett's approach was to simply develop a methodology for converting an existing KB from an RDR structure to a decision tree structure and vice-versa. Another method developed by Wada et al. (2000; 2001b), took an approach similar to Scheffer's Cut95. Wada et al's (2000; 2001b) system first used RDR as a knowledge acquisition tool in the usual fashion, but once sufficient cases are available it then uses the minimum description length principle, proposed by Rissanen (1978), as a complement to a lack of expert knowledge.

Other machine learning techniques have also been developed for aiding rule development or extraction, usually however, for a specific application. One method was Learning Dynamic RDR (LDRDR), developed by Shiraz and Sammut (1998; 1995; 1997) as an adjunct of Dynamic RDR (DRDR). DRDR was used for a control task and uses multiple RDR trees to learn different aspects of the task at hand, forming a kind of multiple classification method.

Recall (3.2), one of the original philosophical foundations of RDR was the *no-model-approach* for KA. It was argued that the domain specific modelling prior to any KA taking place was problematic and time consuming, and simply transferred the KA bottleneck to a modelling bottleneck. However, the model-based approach is not without its own significant advantages. For instance, the development of a model is seen as highly desirable, if not essential, in areas such as knowledge reuse, problem solving methods and ontologies. Thus, a few researchers have investigated methods of crossing the gap between the RDR *'no model'* approach and the *knowledge-level* idea that the '*model is core'*.

One approach to developing a model for RDR, called RDR-Oriented Conceptual Hierarchies (ROCHs) was developed by Martinez-Bejar et al (1998a; 1997). In this approach it is argued that an expert accomplishes the mental processes, such as abstraction, required for the development of conceptual models in a better way than automatic concept extraction techniques such as Richards and Compton's MCRDR/FCA (Martinez-Bejar et al. 1998a). Another major body of work has been carried out by Beydoun (2000; 2002a), Beydoun and Hoffman (1997a; 1997b; 1998a; 1998b; 1998c; 1999a; 1999b; 1999c; 2000a; 2000b; 2000c; 2001) and Beydoun et al (2000) in the development and use of Nested RDR (NRDR). NRDR builds a conceptual knowledge base from smaller simple KBs. The upper layer is an RDR tree; however, each node contains an expert defined concept rather than a rule. The concept is then made up of its own knowledge base establishing its definition and gives a Boolean conclusion, which is used by the upper conceptual KB during its inferencing process. More recently NRDR has been redeveloped using an object oriented database management system (OODBMS) for improved performance (Al-Jadir and Beydoun 2002; Beydoun 2002b; Beydoun and Al-Jadir 2002).

Knowledge Reuse, like software engineering, aims to reduce the effort of repeating a development task through carefully crafting the original solution in a

generalised and extendable manner. Thus, successful reuse of knowledge should reduce the KA bottleneck for later KB developments. The RDR KE approach is to capture behavioural knowledge which is validated simultaneously with the cases. The earliest work in knowledge reuse with RDR was carried out at the outset of the RDR methodological development with GARVAN-ES1 by Jansen and Compton (1988; 1989). The initial work stemmed from the notion of the reuse of data for the purposes of data warehousing and data mining. However, instead of storing data Jansen and Compton stored decomposed production rules, or *knowledge*. The constituent components of each production rule were stored in relation tables, forming a *knowledge dictionary* that could be easily browsed with any starting point, knowledge was automatically cross referenced to other concepts in the domain and the knowledge could be accessed independently of the KR language. The ROCH approach for developing a concept model is an effective tool for knowledge reuse and ontology construction. This work was also extended to incorporating the reuse of knowledge into MCRDR specifically (Martinez-Bejar et al. 1998a; Martinez-Bejar et al. 1999a). The authors, however, note that this work shifts marginally from the RDR philosophy as additional conceptualisation work, through abstraction of concepts and their relationships, is required by the expert, increasing the KA burden (Martinez-Bejar et al. 1999a).

All RDR based methods discussed so far have dealt with crisp knowledge. However, there has been a move to further adapt RDR to handle fuzzy logic, while retaining the context based incremental nature of RDR. Fuzzy logic, based on mathematical fuzzy set theory (Zadeh 1965), is basically a method for representing partial membership to a class or likeness between two objects. The ability to include the use of fuzzy rules within RDR allows for a more natural expression of concepts by experts than the crisp systems used in RDR.

Initial work done in this area is by Martinez-Bejar et al. (1998b) and Shiraz et al. (1998) on the adaptation of ROCH into Fuzzy ROCH, or FROCH. This work looks at a theoretical framework for defining a model that allows experts to use fuzzy modifiers as well as fuzzy frequency quantifiers when constructing new cases used in further processing by the system (Martinez-Bejar et al. 1998b). A second approach to fuzzification, which extends the theoretical framework of the earlier work on FROCH, has been done by Martinez-Bejar et al. (Martinez-

Bejar et al. 1999b) and Cao et al. (1999), called FMR and FRDR respectively. These systems investigated the use of fuzzy rules in MCRDR. One interesting aspect of this work was the method of generating fuzzy rules through the generation of a certainty factor value for a given condition from the uncertainty underlying the context of a particular rule. A second feature, was the system's ability to infer conclusions that are not in the KB through the use of the Generalised Modus Ponens approach (Martinez-Bejar et al. 1999b).

In section 3.2.4 the problem of repetition in RDR was discussed. It was also claimed that the problem is not as great as might be expected. This is partly because of the tree structure generally created by domain experts and partly because experts often tend to provide fairly general rules earlier on in a KB's development (Mansuri et al. 1991a; Mansuri et al. 1991b). Regardless of this, work has been carried out by a number of researchers to attempt to reduce repetition through reorganising, restructuring and generalisation of components.

Some of the earliest work in reducing the problem of repetition originated from the use of induction. InductRDR (Gaines and Compton 1992) could produce a KB which would have no repetition. However, this did not incorporate expert knowledge. Chellen (1995) reimplemented the InductRDR method to develop the *Vinduct* compaction method, which is aimed at reducing the size of manually built KBs. Richards et al. (1996) and Richards (1998a) also investigated the repetition problem as a precursor to work on MCRDR/FCA. In this study Richards followed a suggestion proposed by Gaines and Compton (1992), that repetition would be best solved through first building a KB manually and only after it has collected enough cases is ML techniques applied to reduce repetition for the remainder of the KB construction.

Richards's work is similar to Wada et al's (2000; 2001b) work using the minimum description length principle (MDLP) for rule encoding and earlier work, by the same team, on refined selection of the default classification (Wada et al. 1999; 2001a). Both of these help in reducing the repetition problems in RDR. Further work by Wada et al (2002a; 2002b) has also been used to reorganise an RDR knowledge base so that it can adapt to environmental changes. This is particularly interesting as it is one of the few attempts in RDR to address *dynamic context*. It combines the previous two areas of research by

the team, using MDLP for rule encoding and finding the best default conclusion along with rule deletion and pruning strategies (Wada et al. 2002a; 2002b).

Rather than trying to induct a KB based on information learned from the expert, Beydoun's NRDR minimises the inclusion of repetition in the first place. This is more of a by-product of the NRDR methodology than a deliberate attempt to solve the problem. Basically, because the expert can create mini KBs that represent some concept, it is straightforward for them to use the already created concept as an attribute in the higher order tree. This of course, does not negate repetition entirely but certainly does aid in its reduction (Beydoun 2000). This work has also been extended further into using parameterised NRDR concepts through allowing the expert to form predicates incrementally instead of the usual RDR approach of only using propositional rules. Drake and Beydoun (2000) argue that allowing for predicate logic may be essential in applying RDR to domains suffering from combinatorial explosion (Drake and Beydoun 2000), a major cause of repeated knowledge in RDR, thus reducing the size of a KB.

It was mentioned during chapter 2 that one of the problems with current KA methodologies is that, according to Shaw (1988), experts frequently disagree with each other on the knowledge captured in a KB. Yet this has since been ignored by all the current work in RDR. In fact the RDR methodologies show no reason to suspect they would improve KA in this regard at all. It may even be arguable that they could hinder KA through each expert having a different contextual position. This area of research into RDR is still very young and definitely still requires further work (Richards 1998a).

The only major body of work that investigates this is by Richards (1998a; 2000b; 2000c; 2000d; 2000e), Richards and Zowghi (1999) and Richards and Menzies (1998). Richards's method, rather than trying to develop a KB that captured everybody's viewpoint, was to allow each knowledge source to construct their own unique KB. Each KB is then combined into one shared knowledge base using a reconciliation process that relies on FCA. The only other work that is investigating this issue is by Kildare et al (2004) and Kildare (2004). The basic idea is to use MCRDR as a tool to facilitate team interaction and conflict resolution. This work is not interested so much in resolving the problems caused by multiple experts, instead it is attempting to exploit these problems (Kildare 2004; Kildare et al. 2004).

3.4.3.2 Information Extraction and Tools

Not all research in RDR has been concerned with simply improving the methodology in some way. A number of papers have been published that have instead investigated ways of using RDR based methods to extract or elicit meta-knowledge, or to be able to provide additional aid to an expert through derived information. For instance, the ability of an expert system to be able to offer an explanation for its derived conclusion is fundamental and sets them apart from most other machine learning techniques. Explanation can take many forms from a simple and fairly unnatural rule trace to explanations focusing on context, goals methods and justification (Swartout and Moore 1993). *Casual explanations* are a form of explanation providing a justification between a user's input and the system's output and are primarily used in KBSs built with production rules. *Casual explanations* are generally derived from *casual models*, which provide interconnections between various component behaviours (Lee and Compton 1995; Richards 1998a).

According to Swartout and Moore (1993), a rule trace in a production rule system usually does not provide the deeper knowledge required, as it is compiled out during the development process. Richards (1998a; 2001), however, claims that RDR, represents a paradigm shift to the traditional explanation based systems. It is argued that because RDR retains the history of the development in the exception structure and in the storage of cornerstone cases, that it retains this deeper knowledge. Another advantage in explanation with RDR is the differences between cornerstone cases along the way.

Earlier work on the development of a casual model for explanation from an RDR rule base was done by Lee and Compton (1994; 1995; 1996) and Lee (1996), called Causal MODelling (CMOD). This work aimed to integrate both rules and casual models, in order to provide a method for cross validation and improving acquisition for each, thereby, providing both good performance and good explanation. This work also offers support for knowledge acquisition, verification and validation, and knowledge maintenance (Lee 1996; Lee and Compton 1995). Webster's (1995) honours thesis extended this work by investigating whether the casual links that Lee derived with the occasional aid of expert knowledge could instead be found purely through machine learning.

3.4.3.3 Research Domains

Rather than attempting to improve the general methodologies or to extract useful information from the knowledge base, significant work has also been carried out on applying various methods on traditional expert system based tasks. The method's successful application in these domains is crucial to the further acceptance of RDR based methodologies. Firstly, the configuration problem can be thought of as the construction of a complex product from a set of pre-defined components, while considering a set of well-defined restrictions on how the components may be combined (Soininen and Stumptner 2003). Essentially, a configuration methodology is one that can perform an inference on a case and then use the results from that process as additional input for another pass through the same or a different inferencing process. It is, broadly speaking, a particular type of classification problem that generally requires unique solutions. Like KBS research in standard classification, configuration tasks suffer from difficulties in KA and especially KM. Therefore, a system directly targeting issues, such as RDR techniques, could be particularly useful. However, the difficulty with a configuration task is that each conclusion must provide an attribute-value pair instead of the usual classification. This is required as the attribute-value pair may be needed as input in the next pass and so must be in the same form as expected by the system (Compton and Richards 1999).

One of the earliest attempts to use an RDR based system, called IONICS (Iterative Organisation of Novel Ion Chromatography Solutions), for a configuration task was by Mulholland et al. (1993a; 1993b; 1993c; 1993d). This work developed a technique for configuring the various machine settings for an ion chromatograph. This was accomplished by extracting rules from a large database of published methods to build eight separate classifiers. Once constructed, inferencing was performed by Recursive RDR (RRDR), which was developed to provide a mechanism for consulting the individual RDR trees (Mulholland et al. 1993d). The RRDR solution was to traverse both the true and false branches when an attribute was unknown, rather than only following the false branch. A second method used, called Possible RDR (PRDR) (Mulholland et al. 1993a; Mulholland et al. 1993b; Mulholland et al. 1993c; Mulholland et al. 1993d) only continued inferencing on the true branch.

A later development by Compton et al (1998), referred to as Configuration RDR, redeveloped the machine learning based IONICS system, into an incremental knowledge acquisition method. First, the system was changed to use MCRDR due to it being more adaptable to finding the multiple plausible conclusions required. Second, a new inferencing process was devised that could operate during multiple passes through the KB. Each pass the expert would resolve conflicting suggestions and add non conflicting suggestions for unused single values and then re-inference the case. This process was repeated until either a complete configuration was found or until no more single values could be found for the remaining attributes (Compton et al. 1998; Compton and Richards 1999).

3.5 Summary

This chapter reviewed the historical and current RDR based research. It looked at how the philosophies, discussed in the previous chapter, related to the original advent of RDR. It did this by reviewing the GARVAN-ES1 expert system case study. This case study represented the perfect ES environment yet still required difficult and extensive maintenance. It was found that experts tended to offer their justifications, when correcting a misclassification, within the context of the error, by identifying where the system went wrong. RDR was, therefore, designed specifically to represent this contextually based justification directly within the structure.

This chapter fully described both the RDR and the MCRDR methodologies. This was important as the work in this thesis builds upon the MCRDR methodology and a solid understanding is required by the reader in order to appreciate components of the RM methodology. Finally, a very brief review of the RDR based research was surveyed. Particular attention was paid to two studies in particular: MCRDR/FCA and GENICA. These are two major previous projects that have investigated the use of the MCRDR knowledge base structure for the purpose of discovering interesting new information. In both cases, like in this thesis, they are using the idea that there is information hidden in the MCRDR KB that in the base methodology is not being extracted.

4

Artificial Neural Networks

> *'If people jump out at you suddenly, that's an ambush,' said Owl.*
> *'It's an ambush, Pooh, when people jump at you suddenly,'*
> *explained Piglet.* *(Milne 1926, p108)*

Artificial Neural Networks (ANNs), also referred to as *Parallel Distributed Processing Systems* (PDPs) and *connectionist systems* are methods for modelling the organisational qualities of the biological central nervous system. The idea is that if nature constructed such a superbly compact and versatile system capable of intelligent thought and reason, then our best opportunity for duplicating such results in a computer is to base our solution on what we know works. The hope is that such biologically inspired systems could allow cognitive and sensory tasks to be performed more easily than with serial processing.

The ANN structures can be loosely classified into two groups: *recurrent* and *non-recurrent*, based on whether they involve feedback. Each network has a number of individual processing elements, also referred to as *neurons*, *neurodes*, *nodes*, *units* or *cells*. Each neuron also has *connections* to other neurons. These connections can be either: *interlayer connection* between nodes in different layers; *intralayer connections* or *lateral connections* between nodes in the same layer; or, *self-connections* joining the input and output of the same neuron. Networks can be further distinguished by topology and learning methods.

The aim of this chapter is not to provide a full and comprehensive survey of ANN technology. This is not a neural networks thesis; it simply uses them as a learning tool. Instead, this chapter will provide a sufficient overview of the research stream so that a reader that is unfamiliar with ANNs can attain an understanding of the methodology developed in the next chapter. This chapter reviews the biological influences behind ANNs and very briefly look at the history of neural computing. It also details the backpropagation and radial basis function methods, as these are used in this thesis. Lastly, the chapter will overview a number of other algorithms that were considered.

4.1 Neural Biology

The human brain is a vastly complicated biological entity which contains around one hundred billion cells, where approximately ninety billion are *glial* cells and the remaining ten billion are *neurons*. *Glial* cells are essentially responsible for supplying nutrients, removing debris, holding neurons in place and providing insulation (myelin) for neurons. The cells that have been traditionally of interest to neural computing researchers, however, are the ten billion *neurons*. Each neuron is *connected* to approximately ten thousand to one hundred thousand other neurons, creating a vast network (Bose and Liang 1996).

The neuron is considered to be the basic building block of the human brain. There are two main types of neurons: the local processing *inter-neuron cells* connecting neurons together within a local region, and *output cells* which connect neurons across brain regions. Neurons, however, differ in various characteristics from each other (Beale and Jackson 1990). In this section though, only the general and most common form of neuron will be described.

A neuron's body (Figure 4-1), called the *soma*, has many fine branches, referred to as *dendrites*, forming a tree like structure. These dendrites conduct input signals to the cell body. Output neurons also contain a single *axon* (interneurons don't have axons – instead they produce output via their dendrites), connected to the soma via the *axon hillock*. The axon carries the electrical output pulse, which splits the signal into numerous *axonal arborizations*. The tips of these axon branches connect to the dendrites, soma and axons of other cells via what is called a *synapse*. The connection is actually a small gap, called the *synaptic cleft*, which allows a small chemical transmission of *neurotransmitters* (Bose and Liang 1996).

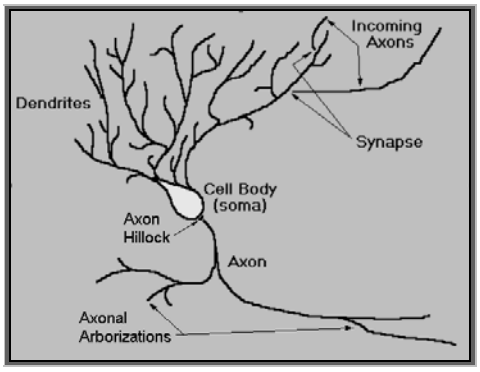


Figure 4-1: Schematic diagram of a single neuron (Zurada 1992).

The neuron operates by receiving either excitatory or inhibitory activations from other neurons via its dendrites. These inputs are aggregated (not always simply a summation) and if they exceed the neuron's threshold at the axon hillock then the neuron fires an electrical charge through the axon. However, the inputs must achieve the threshold within a narrow period of time, called the *period of latent summation*. Each neuron always produces the same action potential when firing regardless of the degree the threshold was exceeded, thus it is an *all or none* process. After a neuron has fired it cannot be excited again by any stimuli for a period of time, called the *absolute refractory period*. This is subsequently followed by a period where the threshold is much higher than usual, called the *relative refractory period*. Therefore, the neuron can now fire, but only if it receives a much larger spike of excitatory inputs. Therefore, it is not the actual strength of the firing neuron that indicates the weight of its inputs, but the length of the interval since it last fired (Bose and Liang 1996).

Learning is believed to be achieved by modifications to the coupling between neurons at the synaptic junction. This is most likely achieved through the release of varying amounts of neurotransmitters. This effectively opens more gates on the dendrite, thus, increasing the joining of the two cells (Bose and Liang 1996). For instance, variations in mood can affect serotonin levels, which are a type of neurotransmitter, while changes in dopamine receptors can cause desensitization.

4.2 History

There are numerous artificial neural networks which represent various methods for simplifying the biological system. Even so called neurobiological models make many assumptions and make numerous simplifications. Yet, if we had the computing power to accurately simulate the biological system, many assumptions would still be required due to an incomplete understanding of the human brain. However, there have been many types of networks that have been developed that have captured different aspects of this natural example (Zurada 1992). The vast multitude of methods nevertheless, cannot possibly be discussed in this thesis.

The first formal model of an individual computing neuron by McCulloch and Pitts (1943), referred to as a perceptron, contained all basic elements to perform logic operations. This work was parallelled by the development of the first learning rule by Donald Hebb (1949), referred to as the *Hebbian learning rule*. This was further elaborated by Frank Rosenblatt in 1958, which allowed for multiple preset layers of perceptrons (Zurada 1992).

One of the most well known approaches in the early years was the ADALINE (*ADA*ptive *LINE*ar combiner) and the later extension MADALINE (*M*any *ADALINE*s) which were developed in the early 1960s. These approaches employed a new learning rule referred to as the Widrow-Hoff learning rule (Widrow and Hoff 1960). These devices were used in a number of applications such as pattern recognition, weather forecasting and control systems (Zurada 1992).

The late sixties saw the end of most neural network research until the mid-eighties, due to the publication of the book *Perceptrons* (Minsky and Papert 1969). In this book the authors revealed the inability of the perceptron based models to learn linearly inseparable problems like the XOR problem without preset interneurons. This difficulty prevailed until Rumelhart and McClelland[13] (1986) introduced the new perceptron using a continuous activation function and a new learning rule (Zurada 1992), discussed in the following section. This has resulted in a multitude of network methodologies subsequently appearing in the literature.

4.3 Backpropagation

The standard backpropagation algorithm, due to its topological use of a directed acyclic graph, is classed as a type of feed forward network. Figure 4-2a shows the most common backpropagation network topology. It consists of many directed edges representing weighted connections between nodes with signals travelling along the direction of the arrows. A network consists of a set of: *source nodes*, called *input nodes*, where the network receives its perceptions of the environment; *sink nodes*, also called *output nodes*, where the network

[13] It became evident, after Rumelhart and McClelland's work that similar results had been published by Parker (1982) and even earlier in a PhD thesis by Werbos (1974).

provides processed interactions with its environment; and sometimes, *hidden nodes*, which are not directly accessible to the outside environment and are used for additional processing. The basic elements can be combined into many possible topologies, which fall into one of two categories: *layered* and *not layered*. Two types of three[14] layered networks are shown in Figure 4-2a and b. The first is a fully connected multilayer network while the second is only a partially connected. Figure 4-2c shows a type of non-layered network[15], where connections are possible from input to hidden and/or output nodes. These jumping connections are generally referred to as *shortcut connections* and are occasionally used for faster learning.

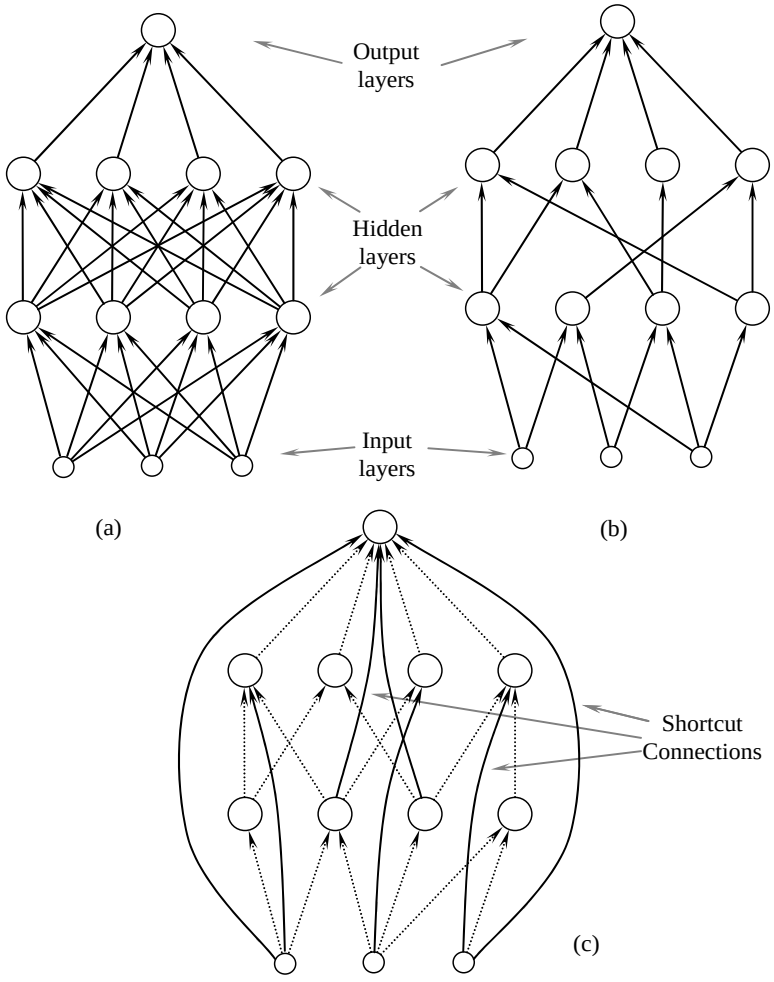


Figure 4-2: Example topologies for feed forward networks. (a) shows a three layer fully connected network. (b) shows a 3 layer partially connected network. (c) shows a non-layered network with shortcut connections.

[14] Generally, the input layer is not counted as a layer when referring to the network topology.

[15] While Figure 4-2c is drawn as if the nodes are in layers, this is only done to simplify its appearance. The connections between nodes do not follow the layering.

4.3.1 Perceptron

Rosenblatt's perceptron (1958) was capable of learning numerous functions provided the first layer of *threshold logic units* (TLUs) , referred to as the *retina layer*, were correctly handcrafted (Bose and Liang 1996). This layer had to be setup by hand to adjust the input function so that the problem to be learned was linearly separable. A second output layer, consisting of a single Rosenblatt perceptron, which could learn based on the first layer's altered perception of the input. This was performed through the use of error minimisation in the *least means square* function through methods like *gradient descent*. However, without the first layer the perceptron was unable to learn linearly non-separable functions. This was the key difficulty with the original perceptron.

The fundamental problem with the Rosenblatt perceptron was the use of the step function for the threshold. This prevented the propagation of an error to the previous layer, because it was impossible to know which inputs to the previous layer were on or off. Therefore, there was no method possible for determining which weights to reinforce. The solution is to smooth out the thresholding function, so rather than simply switching the neuron on or off it gradually switches on and off. This allows information about that node's inputs to be derived from the node's output strength. This smoothing function can be any continuously differentiable monotonic function, such as the sigmoid, tansigmoid or logsigmoid thresholding functions (Bose and Liang 1996).

The resulting network, which can incorporate any number of hidden layers, can still be trained using *gradient descent*. The *backpropagation algorithm* (BPA), often referred to as the *generalised-Δ-rule*, when applied to the output layer's gradient of error using the chain rule of differentiation can be propagated back layer by layer. Interestingly, Bose and Liang (1996) note that the *generalised-Δ-rule* is not restricted to only layered networks but can also be applied to non-layered networks using shortcut connections, as seen in Figure 4-2c.

4.3.2 Mathematics for Backpropagation

Each perceptron sums a set of input sources (can either be an input or hidden neuron), using Equation 4-1, to find the total amount of stimulation received, *net*. Each node's input, x_j, received from the j^{th} input source is multiplied by the associated weight, w_j, where there are n input sources to the neuron. Generally, there is no difference between the hidden and output nodes.

$$net = \sum_{j=0}^{n} w_j x_j \tag{4-1}$$

Additionally, it is common practice to also include a trainable *bias* term, effectively allowing the threshold location to be dynamic. This is usually implemented by having a '*fake'* input neuron which is always *on*. The associated weight is trained in the same manner as the other connections.

The resulting value is passed through a smoothed thresholding function such as the sigmoid function shown in Equation 4-2, where $0 < f(net) < 1$ and k is a positive constant that controls the spread of the function: as $k \to \infty$ then $f(net) \to$ *the step function*. Therefore, generally it still turns *on* and *off* as before if there is a large difference between the threshold and the summation of inputs, but if they are nearly identical then it will give a value in between the two extremes.

$$f(net) = \frac{1}{1 + e^{-k\ net}} \tag{4-2}$$

Some researchers (Han and Moraga 1995; Vamplew 1996) have reported that improved training times can be attained by using the symmetric sigmoid, shown in Equation 4-3, rather than the above asymmetric sigmoid. For these reasons this thesis has used the symmetric sigmoid throughout.

$$f(net) = \frac{1}{1 + e^{-k\ net}} - 0.5 \tag{4-3}$$

After the network has given a value for an input pattern, p, the correct answer according to the dataset is then shown to the output nodes. The error for a particular set of inputs can then be calculated using Equation 4-4 for each output node, o, where: v^r_{po} represents the *value* of the *reward*[16]; v^c_{po} is the *value* of the output node's original *conclusion*; and, δ_{po} is the amount of error. The

[16] In the majority of ANN literature the *reward* is referred to as the *target*, which is then represented by the letter *t*. However, to avoid conflict with the *time* variable, which also uses *t*, in later equations, this thesis will use the reinforcement learning term *reward* with the notation *r*.

function $f'(net_{po})$ is the derivative of the thresholding function, applied to the activation at node *o*, which is simply the weighted sum of the inputs to that node.

$$\delta_{po} = f'(net_{po})(v^r_{po} - v^c_{po}) \tag{4-4}$$

This function, however, does not work for hidden nodes because the correct output required is unknown. Therefore, the error at each of the output nodes is propagated back to the hidden nodes, *h*, and used in Equation 4-5 instead.

$$\delta_{ph} = f'(net_{ph})\sum_{o=0}^{n}\delta_{po}w_{ho} \tag{4-5}$$

The most commonly used thresholding function is the sigmoid mainly because its derivative, as required in the above equations, is simple, making implementation easier (Beale and Jackson 1990). Equation 4-6 shows the derivative of the earlier defined sigmoid function (4-2).

$$f'(net_p) = kv^c_p(1 - v^c_p) \tag{4-6}$$

However, all the networks used throughout this thesis are using the symmetric sigmoid function (4-3). The derivative for this function is given in Equation 4-7.

$$f'(net_p) = k\left[0.25 - (v^c_p)^2\right] \tag{4-7}$$

Equation 4-8 gives the error calculation for the output nodes (4-4) again with the *asymmetric sigmoid threshold* function's derivative (4-6) incorporated.

$$\delta_{po} = kv^c_{po}(1 - v^c_{po})(v^r_{po} - v^c_{po}) \tag{4-8}$$

Equation 4-9 gives the error calculation for the output nodes (4-4) again with the *symmetric sigmoid threshold* function's derivative (4-7) incorporated.

$$\delta_{po} = k\left[0.25 - (v^c_{po})^2\right](v^r_{po} - v^c_{po}) \tag{4-9}$$

Equation 4-10 gives the error calculation for the hidden nodes (4-5) again with the *asymmetric sigmoid threshold* function's derivative (4-6) incorporated.

$$\delta_{ph} = kv^c_{ph}(1 - v^c_{ph})\sum_{o=0}^{n}\delta_{po}w_{ho} \tag{4-10}$$

Equation 4-11 gives the error calculation for the hidden nodes (4-5) again with the *symmetric sigmoid threshold* function's derivative (4-7) incorporated.

$$\delta_{ph} = k\left[0.25 - \left(v_{ph}^{c}\right)^2\right] \sum_{o=0}^{n} \delta_{po} w_{ho} \tag{4-11}$$

Finally, once we have the amount of error to be applied the weights for each of the connections leading to the nodes can be adjusted using Equation 4-12, where η is a gain term, and w_{jl} is the weight from node j to node l[17] at time t.

$$w_{jl}(t+1) = w_{jl}(t) + \eta \delta_{pl} v_{pj}^{c} \tag{4-12}$$

4.3.3 Classification, Generalisation and Fault Tolerance

The original perceptron was limited to only linearly separable problems or at least situations where the developer could reduce the dimensionality to one that was linearly separable. The development of the multiple layer perceptron overcame this, through the use of trainable hidden layers. Now each new perceptron hidden layer increases the complexity of entities that can be learned. For instance, zero hidden layers only solve linearly separable problems, while one hidden layer allows open and closed convex hulls to be learned. Adding a second hidden layer allows the learning of arbitrary regions (Beale and Jackson 1990). Thus, there is no reason to move beyond two hidden layers. Generally, however, most real world problem spaces only require a single hidden layer.

The power of the multilayer perceptron comes from its ability to generalise. After having seen a number of examples the network will find a series of equations that separate the classes within that group of training examples. Then, if a previously never seen input pattern appears it is still able to identify in which classification in the problem space this new example is most likely to sit. ANNs are surprisingly accurate at this, as it chooses the distinguishing features.

This is particularly prevalent in situations where a number of inputs are affected by noise. The noise perturbs the input but the network will still find an approximate answer, whereas other pattern recognition systems and especially expert systems tend to collapse very ungracefully (Beale and Jackson 1990). Additionally, the multiple layer perceptron is very robust when neurons are lost

[17] In equation 4-12 node *l* is being used as a general representation for either a hidden node or output node. Likewise, node *j* simply represents an input source to node *l* not an *input node* specifically.

or damaged. Sudden changes in a single or small group of neurons not only don't cause huge adverse affects but the network can generally recover quite rapidly (Beale and Jackson 1990). This is particularly appealing to the work in this thesis as the number of neurons needs to be altered constantly, thus a robust system for handling this is required.

4.3.4 Problems

There are two primary problems with the backpropagation algorithm. The first is that there is no guarantee of convergence. Occasionally, a network trained with the *generalised-Δ-rule* will settle into a *local minimum*. A local minimum is a stable solution, where changes to the network's weights will produce a worse result, which is incorrect. Sometimes in such a situation there may only be a small 'lip' that needs to be over-come in order to get a much deeper solution. However, the network has no method for determining this and so stays in what it believes to be its best solution.

The second issue with backpropagation is simply the amount of training required to learn a solution. The algorithm's approach of gradient descent is inherently a slow creeping towards a solution. Therefore, the method cannot easily be applied in online environments. This has led to many methods being developed to overcome local minima as well as speeding up learning. One collection of methods has utilised the second order effects in the gradient descent algorithm, significantly reducing the number of iterations that training cases need to be seen. However, they tend to be computationally excessive.

Four of the most common approaches to solving these two issues are: adjusting the *learning rate*, also referred to as the gain; increasing the number of hidden nodes; the addition of noise; or, the use of momentum (Beale and Jackson 1990). At first increasing the gain during the early stage of training can significantly speed learning. Slowly reducing the gain later will allow the network to settle into one of the deeper minima. However, this approach can also increase the time to fully converge to its final stable solution. The second approach is to train the network with more hidden nodes. This is because local minima are considered to occur when two or more disjoint classes are categorised as being identical. Through increasing the number of hidden nodes an improved recording of the input patterns is possible.

One seemingly unlikely approach is the addition of noise. This can be effective in helping avoid local minima by perturbing an input case so that it can jump over the lip and move onto a deeper minima. Similarly, the addition of a momentum term can also help push the system over a lip. Momentum involves adding an additional term into the weight adaptation equation that produces a large change in the weight if the changes are currently large, and will decrease as the changes become smaller. Equation 4-13 shows the momentum term as described by Beale and Jackson (1990), where α represents the momentum factor, $0 < \alpha < 1$.

$$w_{jl}(t+1) = w_{jl}(t) + \eta\delta_{pl}v_{pj}^{c} + \alpha\left(w_{jl}(t) - w_{jl}(t-1)\right) \qquad (4\text{-}13)$$

4.4 Radial Basis Function

The Radial Basis Function (RBF) was originally proposed by Broomhead and Lowe (1988) and has been expanded by many other researchers. RBF networks traditionally fall into a class of networks called *regularised networks*, which also contain methods such as *Counterpropagation Networks* (CPNs) and the more generalised form, *Hyper Basis Functions* (HBF). RBF networks also have many similarities to the above backpropagation network such as a similar topography, as in Figure (4-2a) except using only a single hidden layer. The main difference is the method used for partitioning the pattern space. Backpropagation uses *hyperplanes*, whereas, RBFs instead construct *hyperellipsoids*.

The basic RBF method looks at cutting up the input space by defining a series of multi-dimensional ellipses, or *basis functions*, ϕ. The ellipsoid's arguments are related to the distance from the centre, *y*. This can be seen in Equation 4-14, where the function *s* in *k*-dimensional space, has elements s_k (Beale and Jackson 1990). Additionally, a positive valued width term or shaping parameter, σ_h, is occasionally used to maintain control over the acceptable distance from the centre of the basis function that can be used.

$$s_k = \sum_{h=1}^{n} \lambda_{ho}\phi_h\left(\frac{\|x - y_h\|}{\sigma_h}\right) \qquad (4\text{-}14)$$

The advantage of the RBF network is that once the basis functions have been selected the network is only required to learn the respective coefficients, λ. These coefficients are the weights on the arcs from the hidden nodes to the

output nodes, similar to w_{ho}, in the BPA. Effectively, the basis functions have expanded the inputs into a higher dimensional space where the problem is now linearly separable. This significantly speeds learning over the BPA (Beale and Jackson 1990).

Providing there is a basis function for each input to be classified, then the RBF network is guaranteed to converge. However, this *over-fitting* of the problem space reduces the network's ability to generalise. Therefore, cases not exactly the same as those seen in the training sets or noisy inputs can cause unintended results. The general solution to this problem is to reduce the amount of basis functions, thereby, smoothing the classification surface between the data points. However, this also makes training a process of linear optimisation instead of the exact guaranteed convergence situation.

The main difficulty of the RBF method is in the selection of appropriate basis functions. There are two main approaches to basis function selection. The first is to create basis functions such that they evenly distribute throughout the possible data points. This method is commonly performed in situations where there is little or no knowledge about the data distribution. The second approach is used when the developer has some knowledge of the overall structure of the inputs. In this situation the known structure should be modelled in the configuration of the basis functions.

4.4.1 Mathematics for RBFs

The most common function used for the basis function, ϕ, is the Gaussian function shown in Equation 4-15.

$$\phi(d) = e^{\left(-d^2\right)} \tag{4-15}$$

Usually the distance measure is performed using the Euclidean distance shown in Equation 4-16. The centre of the hyperellipse, y_{ih}, represents the '*weight*' of the connection between the input and hidden nodes, where x_i is the input at node *i*, and *h* is the hidden node.

$$\|x - y\| = \sum_{i=1}^{n} \left(x_i - y_{ih}\right)^2 \tag{4-16}$$

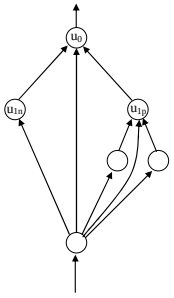
The output layer is then usually trained using standard perceptron based training to learn the λ values for each basis function in the hidden layer.

4.5 Growth Algorithms

The fundamental issue in the attachment of a neural network to the output attained from MCRDR is how to handle the changing size of the input space. As the input space to the neural network increases the network structure will also need to be constantly growing. Potential was seen to be most likely found in the work done in growth based neural networks algorithms. These are a group of algorithms that not only learn weights on connections (such as backpropagation and the basic RBF algorithm) but simultaneously can also learn the structure.

Growth algorithms were developed as a means of addressing the inadequacies of the more standard algorithms. They have been found to converge quickly and overcome problems of local minima (Bose and Liang 1996). The primary difficulty with growth algorithms is the risk of over fitting, resulting in poor generalisation. Growth algorithms generally take a *'divide and conquer'* approach to learning a solution to a problem. For instance, such a system will start with only a single *threshold logic unit* (TLU) and then progressively add additional nodes as they are required. Over fitting occurs when a new node is added to capture the difference between cases for every case, thereby, providing no generalisation.

4.5.1 Upstart Algorithm



The Upstart algorithm, suggested by Frean (1990), starts with a single TLU, u_0. The system is then trained for a given length of time using a perceptron style of algorithm. After training, u_0, it may be linked to two possible errors: wrongly ON, or wrongly OFF. When this occurs new TLUs are added to capture the errors. For instance, another TLU could be added, u_{1n}, which only activates for exemplars that turned u_0 ON incorrectly. This new node will send a large inhibitory signal to u_0, thereby, turning u_0 OFF. This new node is trained using a reduced form of the original dataset, where the nodes that correctly turned u_0 OFF are removed. Similarly, wrongly OFF is corrected by adding another node, u_{1p}. If the new nodes cannot learn a correct solution within a given training period, then additional nodes are added in the same way.

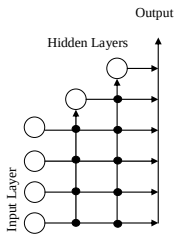
4.5.2 Divide and Conquer

The basic *'divide and conquer'* algorithm, proposed by Liang (1990; 1991), is simple in its approach. The network can be either a single or two layer network shown in Figure (4-2a), which instead of growing deeper like the upstart algorithm, grows wider. It begins with a single hidden TLU, which will attempt to learn the best linear separation between classes within a certain training period. If the problem is not linearly separable then this first node is treated as if it has effectively broken the input patterns into two subsets. At this stage a new neuron can be added to attempt to further divide the input patterns into two smaller subsets. This process continues while the system cannot classify all cases correctly. This will always solve any problem with a finite input pattern space, because at some stage a subset will only contain two cases needing separation which can always be divided giving the correct answer for each. While this method is very effective it is susceptible to generating too many hidden nodes causing a lack of generalisation. One solution is to include a merging phase, where, after dividing the system successfully the system then removes redundant nodes, improving generalisation.

4.5.3 Tiling Algorithm

The *Tiling algorithm*, developed by Mezard and Nadal (1990), grows a multilayer network, Figure (4-2a). It once again starts with a single TLU, called the *master unit*, which is trained for a period of time. Should this node be unable to learn a separation between the classes then a TLU is introduced for each of the two subsets that contain exemplars from both classes. These new TLUs, referred to as *ancillary units*, are then trained with a subset of exemplars. Once again, if the ancillary units are unable to separate their subset into single class exemplars then more ancillary units are added. Eventually, ancillary units will be found that separate the inputs correctly. When this is accomplished a new layer is created with a single new master unit. This new layer takes input from the previously just fully trained layer, which will then follow the same procedure, possibly adding additional ancillary units. Once a new layer is created that can correctly classify all inputs without adding any ancillary units, this becomes the output node of the network.

4.5.4 Cascade Correlation (CC)



The popular Cascade Correlation (CC) network, by Fahlman and Lebiere (1990), starts with only an input and output layer. Over time, hidden nodes are then added to the network, one at a time, as they are needed. The total number of hidden nodes added is dependant on the error bound set in the network, using supervised learning. As each hidden node is added it receives inputs from all the input nodes plus any previously created hidden nodes. Initially, it is not connected to the output of the system. It is then trained, usually using a gradient descent type algorithm. Once the training results in a stable local minimum the inputs to the hidden node are frozen and it is connected to the output. If the result of the network over all is not satisfactory then additional hidden nodes are added and trained in the same way. The resulting structure is not a layered network but instead a cascading of hidden nodes between the inputs and outputs.

4.5.5 Resource Allocating Network (RAN)

One of the problems associated with the RBF function, described earlier in section 4.4, is that the hyperellipsoids need to be defined prior to learning, either from some previous knowledge or from the use of a heuristics. One popular extension to the RBF algorithm is to provide a means of allocating new hyperellipse dynamically during the training process. This was first proposed by Moody and Darken (1988; 1989) and later extended by (Moody 1989) where the hyperellipses learn their centres and widths during training. Platt's (1991) Resource Allocating Network (RAN) extended this further by also adding new nodes as well as learning the appropriate centres, which effectively turned the RBF network into a growth based algorithm. One interesting aspect to the work in this thesis is that the RAN based RBF network allocates units in such a way that they only respond to a narrow region of the input space, therefore, newly allocated units do not interfere with previously allocated units.

Unlike many other growth algorithms it starts with no hidden nodes. The network will select particular input patterns as they are presented and allocate a hidden node to capture the pattern. Input patterns are selected when there is sufficient distance from the previously seen cases. In other words a hidden node

is created when a deficiency is found in a particular area of the input space. Generally, the method allocates neurons that are quite coarse initially. However, over time the widths of the basis functions are reduced, producing a much finer representation.

4.5.5.1 Adaptive Response Function Neurons (ARFNs)

In situations where there is only a small error, the RAN method can move the centres of the basis functions. However, there is no ability to adjust the widths of the functions during online learning. A recent alternative to the RAN method is Ollington and Vamplew's (2003; 2004) Adaptive Response Function Neurons (ARFN). This method was of particular interest to this thesis as it is the only method currently developed that can be applied to the idea of learning inputs.

The ARFN, shown in Figure 4-3, is based on the biological cortical neuron and receives inputs from both an excitatory and an inhibitory interneuron. Each interneuron is excited by the common cortical input and inhibited by a standard bias. An appropriate threshold will allow the output neuron to produce a Gaussian like response to the original cortical input. This forms a selective neuron response rather than more common monotonic function.

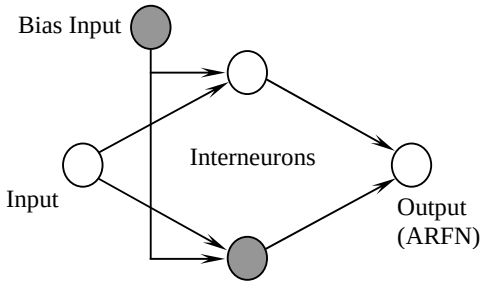


Figure 4-3: ARFN: A neuron for implementing a Guassian-like response function .White neurons are excitory and grey neurons are inhibitory. This diagram is based on the ARFN model in Ollington (2004, p 42).

Training of the interneurons is based on the inputs received rather than the propagated error from the output. For instance, if the response from the excitory interneuron is high then the threshold is trained up, alternatively, if it is low it should be trained down. Similarly, the opposite should occur for the inhibitory interneuron. Effectively, this system is building an asymmetric function which tightly fits the input pattern. The advantage of this approach is that it allows the neuron to reshape its hyperellipsoid centre and widths while training by only looking at the inputs and not at the returning error from backpropagation.

4.6 ANNs Finding Missing Variables

The primary issue in the development of RM is the growth of the input space. This is a problem that neural network techniques do not traditionally handle because usually the input space of a problem is well defined. The only research stream studied in ANNs that really touches on any form of a changing input space is the area of how to handle missing values. These occur when some attributes are occasionally missing from the dataset in particular input patterns.

There have been a number of techniques used for resolving this problem in ANNs. Two of the main approaches used are value substitution and system reduction (Vamplew 1996). Value substitution involves determining some value to use in place of the missing value. For instance, the simplest approach may be to train additional ANNs to predict the missing value. Alternately, the missing value could simply be replaced by an innocuous input that does not affect the output of the network. One of the most computationally expensive approaches is multiple-substitution, where the network is evaluated several times with different values substituted for the missing value. The results are then combined in some way to produce the final output (Vamplew and Adams 1992).

System reduction looks at methods for restructuring the network so that the system can recognise missing values and be able to learn to handle them without being told directly. For example, the *flagged network* provides two inputs per attribute, where the second indicates the attribute's value and the first indicates whether the value was missing. A second approach is to use *shadow weights*, where missing inputs are given a special value which indicates to the network to use a special weight for that attribute instead (Vamplew et al. 1996).

The problem of a missing value differs significantly from the problem faced in this thesis. Missing value researchers generally deal with situations where they at least know how many attributes there are normally. They usually also see the attribute numerous times before it is missing, giving them the opportunity to learn an estimation before hand. Whereas in RM, there is no way to determine how many inputs will be required, therefore, the network cannot be initialised with all the nodes at the start. Nor are there any examples of an attribute's value prior to needing the estimation, thus, no training is possible. Therefore, the systems used in this stream were found inadequate for adaptation for RM.

4.7 Other Classifiers

There are thousands of classification techniques available. It is not plausible for any thesis to consider them all. The aim of this thesis was to find a means for extracting hidden and dynamic contexts from the MCRDR structure. Initially, this was not limited to artificial neural networks specifically, instead many other techniques were also considered, although universally discounted as being unsuitable. For instance: clustering methods such as *K nearest neighbour*; kernel classifiers such as *Support Vector Machines*; and, statistical based methods such as *Bayesian classification*. However, all methods investigated were found to be significantly more restrictive in regards to their ability to learn online and to have their input space changed. For these reasons investigations into general classification methods were dropped in favour of concentrating on forming a new technique based on previous ANN methods.

4.8 Summary

This chapter reviewed the biological and historical background of artificial neural network (ANN) techniques. It investigated in detail the two main algorithms used throughout this thesis: the backpropagation and radial basis function algorithms. The aim of this chapter was not to fully review the ANN research stream, but to simply ensure the material required for the reader to follow the later material in this thesis was provided.

This chapter also reviewed a number of other methods that have been considered during the development of RM. Most of these methods have been rejected, due to various difficulties in extending them for online learning in a growing input space. However, they are still presented here because in some cases they have provided inspiration to the method developed in the following chapter, especially methods such as the resource allocating network (RAN). These were also highlighted to show the difficulties in developing the RM technique, which will be discussed further in chapter 5. Finally, this chapter briefly mentioned methods for finding missing values in the input space, as this is the closest related work to that carried out in this thesis.

Part 2:
Methodology and Initial Results

5

Rated MCRDR (RM): Methodology

> *'What does the North Pole look like?'…'I suppose it's just a pole in the ground?' [asked Christopher Robin]. 'Sure to be a pole,' said Rabbit, 'because of calling it a pole, and if it's a pole, well, I should think it would be sticking in the ground, shouldn't you, because there'd be nowhere else to stick it.'*
>
> *(Milne 1926, p110).*

As stated in the introduction the original inspiration for this thesis was the notion of finding a plausible solution to the *process of practice*. Furthermore, it was argued that this could best be solved through a redefinition of the situated cognition view of context. The proposed definition suggests that context could be considered to be *a posteriori* in particular situations. Therefore, a system that meets the requirements of an *intermediate situation cognition* view would be required to both be able to detect hidden context as well as be able to dynamically adjust its contextual understanding over time.

This inspiration led to the idea of developing a methodology that combined MCRDR and a function-fitting algorithm. As detailed earlier MCRDR was selected due to its being identified by Gaines (2000) as a promising methodology for modelling the *process of practice*. The ANNs used in the implementation in this chapter, were selected as they were adaptable to the requirements of the project. Other function fitting methods were ruled out due to their incompatibility and inflexibility. The purpose of combining these is to capture the advantages of each, for instance, the generalisation ability of the function fitting method, along with the knowledge acquisition and online learning ability of MCRDR. It was determined at the outset that a system capturing the advantages of both algorithms could be used not only to improve classification and learning, but could also be applied in a number of useful areas.

The previous three chapters provided a review of the methods, techniques and ideas that have influenced the development of Rated MCRDR (RM). This extended chapter will detail the core RM algorithm and its development. It will

explain the implementation decisions and variations used in the process. There are seven different methods developed, each of which is thoroughly tested in the following chapter.

First, this chapter describes the justifications and expected outcomes from the methodology. This is followed by a progression through the seven methods developed, from the non-network based weighted method, to the simplest linear gradient descent method to the more detailed Gaussian based growing network. During the development of various methods, explanations will be given for why a number of methods were not used and why certain changes were made to the methods that were used.

It is important to note at the outset the different notations used throughout this chapter. The notation for the concept of the methodology, as previously mentioned, is *RM*. Each implementation variation presented has been given a subscripted tag, for instance $RM_{bp(\Delta)}$. This notation allows the different methods to be easily identified. It should also be noted that in previous publications different methods had simply used the notation RM (Dazeley and Kang 2003a; 2004a; 2004b; 2004c) and others WM (Weighted MCRDR) (Dazeley and Kang 2003b). The notation has been standardised for this thesis and where the algorithm has been previously published with a different notation a footnote will be given informing the reader of the changed name.

5.1 Overview

At various stages throughout this thesis a number of reasons have been identified for why this methodology was developed. In this brief section these will be reiterated and properly explained. First, it will look at deficiencies within the MCRDR algorithm itself and then review the philosophy behind the methodology and how this is achieved. Furthermore, it will broadly describe the basic idea behind the algorithm and how it meets the goals of the philosophy that inspired it. Lastly, it will briefly introduce what the possibilities are for the method.

5.1.1 Problem with MCRDR

MCRDR incrementally builds a tree containing pieces of knowledge. Due to the structure, knowledge is stored, context is automatically captured. This is the context based on what is already known by the system. The multiple classification nature of MCRDR introduces an added complexity to the structure not encountered in single classification RDR. Namely, because any children can fire, you end up with a number of branches potentially firing for any given input case. This is obviously the point of the system; but, if each path being followed is derived individually without any reference to what is occurring in the other paths then there is a lack of cohesion between these separate paths.

At this stage there is little work, with the exception of Richards (1998a) (section 3.4.1), that has been done concerning looking at which paths through the tree are followed in correlation with each other. For instance, if a particular group of paths always occur with a particular type of input case, is this, in some way, significant? It is the position of this thesis, that there is a correlation between the final classifications found and that this is information not currently being captured. This thesis is based on the idea that the MCRDR knowledge map contains hidden information within the structure itself, which can and should be extracted. Furthermore, once this information is extracted, it could be of particular use in understanding more about the areas that the KB needs to operate within.

5.1.2 Philosophy of Knowledge Revisited

The point of the second chapter in this thesis was not purely historical, but also to develop the philosophical background for the method developed. The philosophy of *weak situation cognition* has assisted in the development of contextually aware methodologies (Menzies 1998). There are now many methods developed that seriously attempt to include context in their systems (Menzies 1998). However, these systems still cannot fully enter the *process of practice*. Furthermore, these systems are still limited in their application.

The development of the *intermediate situation cognition* view of knowledge may help penetrate the *process of practise*, by allowing a KBS system to be able to process and understand the knowledge contained in its own KB more

effectively. Additionally, it potentially identifies and clarifies many of the issues that strong situation cognition practitioners believe to be the primary impediment to KBS techniques.

The hypotheses of this thesis are not related to proving the philosophy of intermediate situation cognition (ISC) or claiming that it solves the issues of the *process of practice*. These issues require a number of appropriate methodologies applied in a number of *process of practice* type applications to justify such claims. However, if the developed algorithm is able to meet this thesis's identified hypotheses then this goes some way to showing that there are hidden contexts within the MCRDR knowledge map and that RM is capable of capturing this information. If hidden context can be found in MCRDR then this also goes someway to meeting the ISC view of knowledge. Furthermore, if this information can be used to improve the behaviour of the system, then this highlights that the philosophy of incorporating hidden and dynamic context is a plausible direction for KBS research to explore into the *process of practice.*

5.1.3 Proposed Solution

The proposed methodology in this thesis is a hybrid system combining MCRDR with an ANN. The output from the MCRDR inferencing process is passed to the neural network as its input. The network then feeds the input forward producing either a single output or a multitude of outputs. This hybridisation is simple in concept, but far from trivial to design and implement, in a way that captures the advantages of both of the original methodologies.

Basically, the system is designed to recognise patterns of rules and classifications for particular cases and to attach a weighting to this observed pattern. Nowhere in the actual knowledge map is this information actually recorded; it is simply derived information from the pattern of rules evaluated in the MCRDR tree. This pattern exists because there is either a conscious or subconscious relationship between these classes in the expert's mind. Therefore, the captured pattern of rules in their static context is effectively a type of hidden or unknown context. This now discovered context can be given a value representing its contribution, positive or negative, to a particular task. Additionally, through the use of an ANN, it is possible to constantly re-adjust this value as the context changes dynamically over time.

5.1.4 RM Possibilities

Like with any neural network, RM requires a means of receiving training examples so it can effectively learn. Also, as in a normal ANN, this is really only limited by the imagination of the developer. Therefore, the RM method potentially can be applied in numerous domains. Additionally, within these domains it can provide many advantages over standard networks and KBSs.

One major use for this method that is explored in this thesis is prudence analysis. As will be detailed in chapter 7, the network, if given careful feedback, can learn how to recognise cases from outside its knowledge domain that it has not encountered. Other possible uses for the method occur in data mining, information filtering and even reinforcement learning. These are by no means the limit of the proposed method. Other areas, such as natural language processing, induction, Web-spidering and robot navigation are all possible areas of application, just to name a few. If training examples and expert knowledge is available then the system should be able to learn fast and generalised solutions.

5.2 Implementation Overview

This section provides an overview of the RM implementation. It describes the overall design of the system, as well as details of the MCRDR component's implementation. The second ANN component is not directly discussed in this section. Instead, it mainly discusses the problem faced in using a function fitting method to capture patterns in MCRDR inferenced results. The different implementations of the second component are discussed, in sections 5.3 to 5.9, in detail. The final part of this section discusses how the two main components of the method can be integrated.

5.2.1 System Design

The full RM algorithm, given in pseudo-code in Figure 5-1 and shown diagrammatically in Figure 5-2, consists of two primary components. Firstly, a case is pre-processed to identify all of the usable data elements, such as stemmed words or a patient's pulse. The data elements are presented to the

1. Pre-process Case
Initialise Case *c*
c ← Identify all useful data elements.

2. Classification
Initialize *list* to store classifications
Loop
If child's rule evaluates Case c to *true*
list ← goto step 2 (generate all classifications in child's branch).
Until no more children
If no children evaluated to true then
list ← Add this nodes classification.
Return *l*.

3. Evaluate Case
$\overline{x}$ ← Generate input vector from *list*.
ANN ← $\overline{x}$
$\overline{v}$ ← *ANN* output value.

4. Return RM evaluation
Return *list* of classifications for case *c* and
Value $\overline{v}$ of case *c*.

Figure 5-1: Pseudo code algorithm for RM. This describes the general inferencing and feedforward process used in RM and does not detail the network component as this is different for each version.

standard MCRDR engine, which classifies them according to the rules previously provided by the user. Secondly, for each attribute, rule or class identified, an associated input neuron in the neural network will fire. The network finally produces a set of output values, $\overline{v}$, for the case presented. The system, therefore, essentially provides two separate outputs; the case's classifications and the associated set of values for those classifications.

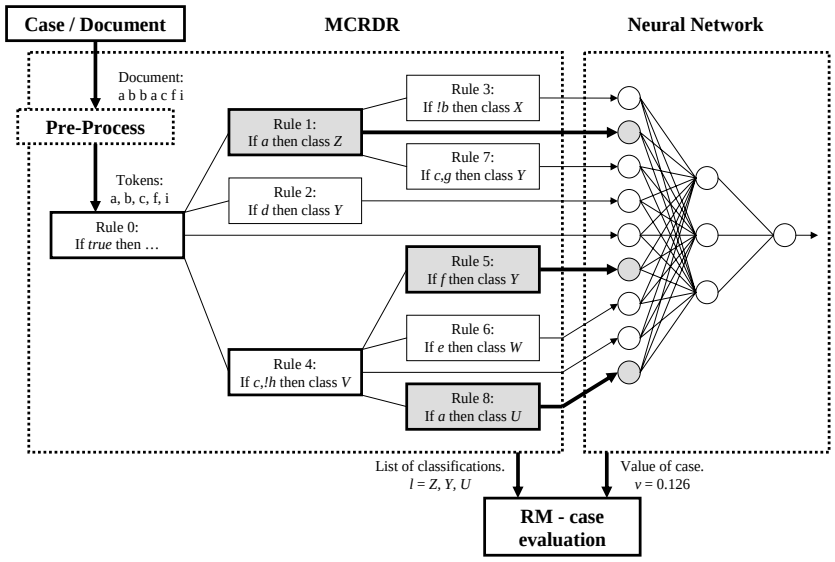


Figure 5-2: RM illustrated diagrammatically. This diagram simply shows the general idea of an attached network. The topology of the network changes according to the type of RM actually used. The integration method shown is the terminating rule association (TRA) method (5.2.4.4).

Figure 5-2 shows a document classification and storage system where documents are also rated to judge their immediate importance. In this example a document with the tokens *{a b b a c f i}* has been pre-processed to a set of unique tokens *{a, b, c, f, i}*[18]. It is then presented to the MCRDR component of the RM system, which ripples the case down the rule tree finding three classifications: *Z*, *Y*, and *U*; from the terminating rules: *1*, *5*, and *8*. In this example, which is using the TRA (*Terminating Rule Association*) method (section 5.2.4.4), the terminating rules then cause the three associated neurons to fire. The input pattern then feeds forward through the neural network producing a single value of 0.126. Thus, this document has been allocated a set of classifications that can be used to store the document appropriately, plus a rating indicating the importance of this document to the user.

5.2.1.1 Learning in RM

Learning in RM is achieved in two ways. Firstly, the rating component receives feedback from the environment concerning the accuracy of its predicted rating. Thus, a system using RM must provide some means of either directly gathering or indirectly estimating the correct rating. For example, in an email application where the system was required to order the documents in the order of importance, the amount of reward given to the network could be based on the order the articles are read by the user or whether the user prints, saves, replies, forwards or deletes the email. How the network actually learns is either in the standard way described in chapter 4, or, as described in the specific section detailing the network used (sections 5.3 to 5.9).

The MCRDR component still acquires knowledge in the usual way; through the user identifying incorrect classifications and creating new rules and occasionally new classifications. Therefore, in the basic RM implementation the expert must still review cases and check classifications are correct. Methods of avoiding this are explored in chapter 7, where prudence analysis is performed using RM.

[18] Repeated tokens could also be totalled and used in rules as well, but for simplicity this example only uses unique tokens.

5.2.2 MCRDR Component

The MCRDR methodology was implemented as a basic module that could have different components added to it using one of the integration methods described in section 5.2.4. The different components are described in detail in the following sections of this chapter. The MCRDR algorithm itself was implemented using the majority of the features described in section 3.3. The primary difference was to only store a single cornerstone case, rather than all new rules creating cases as described in the MCRDR technique. The only case stored in this implementation is the one that is responsible for the creation of that particular rule. It was found during testing that the standard method of storing cornerstone cases, as described earlier, was ideal. However, in many situations it significantly reduces an expert's ability to create rules. It was found that in situations where there is a limit on the number of attributes often the expert cannot create a rule. This is especially the case in the majority of simulated environments as most datasets are too simplified. Only storing a single cornerstone case, however, does not significantly hamper the method's ability to learn. The only real effect is that the natural validation and verification ability of MCRDR is slightly reduced, meaning occasionally the rare additional rule has to be added to correct an incorrect rule.

5.2.3 Artificial Neural Network Component

The neural network components were designed to be plugged on to the end of the MCRDR component using an integration technique described in section 5.2.4. The ANNs used are based on the backpropagation (4.3) and RBF (4.4) techniques. It should be noted that these methods are both traditionally supervised neural networks. However, they have also been used in unsupervised environments such as Tesauro's TD-Gammon (1994). In these environments a separate reinforcement learning method is used to act as the supervisor. In this thesis the networks are also being used in a similarly unsupervised way (except there is no temporal separation). Specifically training is supplied either using the classification of the case or by a calculated class or rating. The specifics of the training rewards used in the various experiments are described fully in section 6.1.

There were seven methods developed and described in detail starting in section 5.3. Each of these methods was designed to test a particular area of the tasks required by the system. Some methods were not expected to work very effectively, but were developed to show that the more advanced methods were required to capture the hidden and dynamic contexts. The primary issue in this project, however, is how to use a neural network in an environment with an ever changing input space. The following section will define this problem mathematically, describing exactly what output can be generated from the MCRDR inferencing process and what is required of the network.

5.2.3.1 The Problem

MCRDR produces an output that can be interpreted in a number of different ways. As the inference process is performed a number of rules fire down the tree and through particular branches. Keeping track of the rules that have fired is one form of output which is highly descriptive of the KB structure. A subset of this is to just consider the final nodes in the path, or *terminating rules*. There is also information within the rules' nodes themselves. For instance, each terminating rule contains an identified classification or action, which is the usual output from the MCRDR inferencing process. However, frequently many terminating rules may have the same classification, effectively meaning that this is a reduction in the useful information from the MCRDR tree. Lastly, each rule also contains a number of attributes that apply to that particular rule. These potentially indicate important global information; however, only using attribute information removes some of the contextual based knowledge in the KB.

Expressing these outputs mathematically, it can be seen that the output from the MCRDR methodology is essentially a set of rules fired, denoted *R*, where $R \in \wp(R^*)$, and R^* is the set of all the rules currently in the knowledge base. Furthermore, the final terminating rules are those that inferencing could not continue past due to them either being a leaf node or because none of their children fired. This output can be given by $R^T \in \wp(R^*)$, where R^T is the set of terminating rules and $R^T \in \wp(R)$. Using the more common classification, denoted *C*, based view, then $C \in \wp(C^*)$, and C^* is the set of all possible classifications. Lastly, the attributes, denoted *A*, identified in the rule conditions

of fired rules could be expressed $A \in \wp(A^*)$, where A^* is the set of all possible attributes currently identified within the MCRDR KB.

Clearly, regardless of which method or combination of methods is used for gathering MCRDR's output, the fundamental mathematical nature of that output is the same. Therefore, for simplicity's sake this is reduced to a common notation. MCRDR can be said to produce a set of features, denoted *F*, as outputs, where $F \in \wp(F^*)$, F^* is the set of all the features currently in the knowledge base, and $F^* = R^* \cup C^* \cup A^*$.

Given the above input to the ANN method, the output is a set of values, $\bar{v}$, which provides one or more varying results in applications where dissimilar tasks may need to be rated differently. For instance, v_0, may identify the desirability or importance of the case presented. Therefore, a mapping must be found from the set $F \rightarrow \bar{v}, \forall F \in \wp(F^*)$. Additionally, RM should be able to learn this mapping for both linear and non-linear sets of features quickly and be able to generalise effectively. Thus, RM needs to identify patterns of features and then associate a value for each pattern through the use of a function-fitting algorithm.

The neural network was integrated into MCRDR by linking individual features used to an input neuron. Thus, for each feature found by the MCRDR system, an associated neuron will fire. The obvious problem with this, as mentioned during the introduction, is that the input space is constantly growing. Every time the expert notices a deficiency in the KB they add a new rule, potentially with attributes never seen before in the conditions and occasionally forming a new conclusion. Therefore, the output feature space, F^*, of MCRDR can grow towards, $F^\#$, where $\|F^\#\| \geq \|F^*\|$ and $F^\#$ is finite but unknown.

To overcome this problem a method for adding input nodes was required. This cannot be easily achieved without significantly influencing the already semi-trained network. As will be detailed later in this chapter, through careful redesign of the ANN algorithms used and selection of initial weights for the new nodes certain effects can be removed or minimised. These, however are unique for each technique developed.

One interesting feature of this problem is that if the expert has just added a new rule it may be possible to assume that they are reasonably sure of the class or value for that case. This is due to having just spent the time reviewing it themselves. Therefore, it may be possible to automatically ensure the network is adjusted accurately for the newly added rule and neuron. This would need to be done by directly calculating a plausible value for any new components added to the network. Once again this calculation is unique for each method developed.

5.2.4 Component Integration

It was identified above that there are four possible types of features that can be derived as outputs from the MCRDR component of the RM algorithm. Integration of the MCRDR and ANN components is carried out by codifying the relevant features taken from MCRDR and converting these into a single input array of values, $\bar{x}$, which is to be provided to the second component for processing. There were five association methods developed for integrating the two portions of the hybrid system. With the exception of the last method, each of them used a discrete *on/off* (*1 and 0*, respectively) input sequence.

5.2.4.1 Class Association (CA)

The *class association (CA)* method analyses every path returned from the MCRDR component and determines the identified classes. Each classification that has been identified by the user is given an index number identifying which input neuron is its associated point of connection. From this information an input array can be formed, where each class currently identified by the expert has a single input into the network. If that class was found in a terminating node from a path taken from the MCRDR inferencing process, then this is set to be *on*, otherwise it is *off*.

The primary advantage of this association method is that it reduces the number of neurons being created in the ANN, thereby, also reducing the size of the input space. However, the reduced input space also means a reduced amount of information about the case, thereby, potentially limiting its learning ability and maybe allowing for less generalization. CA was rarely used in this thesis's final results, as the other methods usually outperformed it in most situations.

5.2.4.2 Attribute Association (AA)

The *attribute association (AA)* method analyses every path returned from the MCRDR component and extracts all the attributes that were used in making each rule fire. Each attribute identified by the user was given an index number identifying which input neuron is its associated point of connection. From this information an input array can be formed, where each attribute currently identified by the expert has a single input into the network. If that attribute was found in a node from a path taken from the MCRDR inferencing process then this is set to be *on*. An added complication can come about from the possibility of a rule containing a '*not*' attribute condition. This can be resolved by either having a *negative on*, -1, or by treating it as a different attribute. If treated as a different attribute then it is given its own index and associated network input. The second option was the only method used in this thesis. A further variation is to only treat an attribute as being *on* if it was used in the terminating rules condition, instead of looking at them all through the path.

One interesting aspect of this method is that it takes contextually located symbols and uses them in a globalised environment. It is like the commonly tried keyword method except the keywords are selected online by the user, which may have interesting applications in some areas, such as information filtering. Another advantage of this method is that if a path, not usually followed, is taken resulting in similar attributes in many of the rules then the system could potentially still achieve a good generalised estimate. However, this association method is subject to the situation being used. The greatest problem area for this method is when the firing of a particular attribute is highly contextually dependant. In such situations inputs would fire globally and, therefore, not when required. Once again it was found in this study that it was rarely better than other methods.

5.2.4.3 Rule Path Association (RPA)

The *rule path association (RPA)* method analyses every path returned from the MCRDR component and determines the rules that fired. Each rule has an index number identifying which is the associated input neuron. From this information an input array can be formed, where each rule currently identified by the expert

has a single input into the network. If that rule is found in the path taken then it must have fired during the inferencing process, therefore, the input neuron is set to be *on*.

This method was consistently one of the better performing association methods. While it does produce significantly more input nodes, it allows significantly more information to be passed on from the first component. One advantage is that if a case only diverges slightly from a previous case, then the input to the network only changes slightly. Therefore, a lot of contextual information is passed on to the network.

5.2.4.4 Terminating Rule Association (TRA)

The *terminating rule association (TRA)* method is essentially a subset of the RPA method. It analyses every path returned from the MCRDR component and determines the rules that fired at the end of the path. Therefore, an input for each neuron is switched *on* only if the associated rule both fired and was either a leaf node or none of its children fired. This method creates neurons at the same time as the RPA integration method but does change the number of inputs that fire significantly. This technique did occasionally work well, however, it does not convey any contextual information. Therefore, it is better suited to situations where the path information inhibits learning. For instance, when many rules have been added as contradictions then the paths may not contribute as well. The RPA method works well when the rules created are specializations of the parent instead.

5.2.4.5 Decreasing Rule Path Association (DRPA)

The *decreasing rule path association (DRPA)* method is the only method used in this thesis that does not use the *on/off* approach. This method could be viewed as a combination of the RPA and TRA association methods. For each rule that is encountered in a classification path an input is given for the associated neuron. The value of the neuron's input, however, is measured by the distance from the terminating rule. Generally, this involves giving the terminating rule's neuron an activation of *1*, and a decreasing activation for each input that connects a rule higher up in the MCRDR tree. The amount of decrease per level can either be static or relative to the depth of the path. For instance, in

the static approach you may simply subtract some amount, such as *0.25*, for each level. The relative approach alters the decrease according to how many rules are in the path. One simple relative approach is a linear decrease from 1 to 0 from the terminating rule to the root using equation 5-1.

$$r_i^v = \frac{r_i^d}{\|P\|}, r \in P, P \in \wp(R), R \in \wp(R^*) \qquad (5\text{-}1)$$

Where P is the set of rules in an individual path; r_i^d is the depth of the i^{th} rule, r, in the path ($r^d=0$ at the root node and $r^d=||P||$ at the terminating rule) and r_i^v is the activation value for the i^{th} rule. Alternatively, a non-linear method could be used such as a sigmoidal or exponential function. In this thesis only the linear method was used.

The original idea behind this method was to introduce a means of removing some degree of discreteness from the inputs. This was expected to help the ANNs develop; as such algorithms often perform better with continuous inputs. This method was occasionally found to produce a slightly better response than the RPA method; however, generally it did not significantly aid learning. One problem was the varied level of a particular node relative to the terminal node. Thus, sometimes a node would strongly contribute to the input space and other times it would have little effect at all. A second issue occurred when a node was the parent of multiple firing branches. If one branch was short and the other was long then the node could have either a high or low activation value. In this implementation it was always set to the highest possible value.

5.3 RM_w: Weighted Technique

The simplest approach, referred to as RM_w, does not use a neural network. Instead, a value is associated either with the rule, class or attribute and after inferencing these values are averaged. This system is implemented very simply. The primary difficulty is deciding what value to assign a new feature identified by the expert (5.3.1). This method was implemented as one of the benchmark methods for the other RM methods to be tested against. This was to ensure that the other methods were performing better than what was expected to be the worst performer, as well as to verify that the trouble of developing the more elaborate methods later on was warranted.

5.3.1 Implementation

The basic implementation developed for RM_w used equation 5-2 to calculate the appropriate RM output value, v, for each case. In situations where more than one RM output was required per case, then additional weights, w_i, were stored for each relevant output. Essentially, equation 5-2, is simply averaging the weights of the features, where n is the number of MCRDR features that fired and $n \leq \left\| F^* \right\|$, where $\left\| F^* \right\|$ is the total number of MCRDR features currently being used.

$$v = \frac{\sum_{i=0}^{n-1} w_i}{n} \tag{5-2}$$

Weights assigned to newly identified features were calculated using equation 5-3. In circumstances where more than one RM output was required for each case, then additional weights were calculated for each. The idea is to calculate a value which allows an identical case in the future to get the correct output. This is achieved by subtracting the average from the *value of the reward*, v^r, received by the system. It is possible, and frequently occurs, that the expert can add multiple new rules for the one case. In these situations the calculated weight is divided by the number of new features, m. This effectively distributes the weight between the created rules giving each new feature the same weight.

$$w_n = \frac{\zeta\left[v^r - \left(\sum_{i=0}^{n-1} w_i \Big/ n \right) \right]}{m} \tag{5-3}$$

The additional *step-distance* function modifier, *Zeta (ζ)*, should always be set in the range $0 \leq \zeta \leq 1$. It is included to allow adjustments to whether a full step or partial steps should be taken for the new features. For instance, if ζ is one then the new weights will provide a full step and any future identical cases will give the exact correct answer. A lesser value for ζ causes new features to only receive a portion of their value. It was found in testing that the inclusion of the ζ modifier allows better performances in some situations, especially in generalisation.

5.3.2 Discussion

This method however, does not meet many of the previously stated requirements. For instance, it does not allow for the discovery of non-linear relationships. While such relationships may not always exist, it is a problem feature that may need addressing. More importantly, the method doesn't allow for any form of learning. If the philosophical ideas stated in chapter 2 turn out to be *false* and there are no hidden or dynamic contexts then this method should perform as well, if not better, than the subsequent techniques. This is because the immediate value received should not need to find non-linear relationships. Nor should it need to find dynamically changing values.

5.4 $RM_{l(\varepsilon)}$: Basic Linear Technique

This similarly simple method takes the other approach, where it allows for learning but no immediate weight is assigned to new features. This method is identified by $RM_{l(\varepsilon)}$, where *l* stands for *linear method* and the ε identifies a *random* initialisation for new neurons. This method uses the simplest of ANN implementations. Basically, it uses a single layer network to capture linear relationships. This allows for new nodes to be added with little concern for the initial weight. As with RM_w, this method was also implemented to show the value in the later methods.

5.4.1 Implementation

The implementation developed for $RM_{l(\varepsilon)}$, simply used the standard feedforward weighted sum, Equation 4-1, which is passed through the symmetric sigmoid function, Equation 4-3. New neurons were added by simply selecting a small initial, positive or negative, random number. A value other than zero is required to ensure some learning occurs. This method allowed the learning process of the network to appropriately adjust the weight on the new neuron's connection.

In this method there is no reason to be concerned about the new node hampering the learning of the other, older and better trained connections. This is because the new node will only affect the output when it receives an activating input. Therefore, given sufficient training time this network will learn a reasonable linear value for the relationship between features.

5.4.2 Discussion

This method would be expected to work well in situations where relationships between features are linearly separable. However, it would have no ability to learn non-linear solutions. Also, while it would be expected that the extra information the network receives from the MCRDR component would speed the network's learning over a standard network, this would only be a minor improvement. Therefore, while this method allows learning, which improves on the RM_w method, it fails in two regards: being able to learn non-linear relationships and being able to learn fast enough to be used in an online environment.

5.5 $RM_{l(\Delta)}$: Advanced Linear Technique [19]

The problem of faster learning was partially solved by the first implementation. RM_w initialised the weights by calculating the correct value when the weight was created. This is possible because the expert had just created a new rule and, depending on the task at hand, a value can generally be derived for the new feature which is much closer to the final required weight. Therefore, such a feature allows us to learn an initial weight that is reasonably accurate and significantly faster. While the mathematics is slightly more complicated, this same idea is certainly applicable in the networked situation. This method, denoted $RM_{l(\Delta)}$, where the Δ indicates the use of the *single-step-Δ_l-initialisation-rule* (5.5.1.1), combines this idea with the learning ability of the $RM_{l(\varepsilon)}$ method.

5.5.1 Implementation

The implementation for this method is essentially the same as the previous $RM_{l(\varepsilon)}$ method. The only exception is that when a new node is added a flag is set so that on the subsequent update the system can use the *single-step-Δ_l-initialisation-rule* to set the new connection to a more appropriate weight than the random number generation allowed.

[19] This version of *RM* was previously published in *Weighted MCRDR: Deriving Information about Relationships between Classifications in MCRDR* by *Dazeley and Kang* in *2003* with the notation *WM*.

5.5.1.1 The Single-Step-Δ_l-Initialisation-Rule

The *single-step-Δ_l-initialisation-rule* directly calculates the required weight for the network to step to the correct solution immediately. This is done in fundamentally the same way as in the RM_w method. However, because the feedforward process passes through the sigmoid function, rather than the simple averaging used in RM_w, then we must go back through the inverse of the sigmoid when calculating our new weight.

Equation 5-4, calculates the weight needed for the new input connection, w_{no} by first calculating the error in the *weighted-sum*, δ_{ws}. This is then divided by the input at the newly created input node, x_n, which is always *1* in this implementation, where there are $n>0$ input nodes and $o>0$ output nodes. In a similar manner to the RM_w implementation this value is also divided by the number of new features added for the current case and multiplied by the step-distance modifier, ζ.

$$w_{no} = \zeta\left(\frac{\delta_{ws}}{mx_n}\right) \tag{5-4}$$

δ_{ws} is calculated, using Equation 5-5, by first deriving the *target weighted-sum*, T_{ws}, from the known error, δ, and subtracting the *actual weighted-sum* at a particular output node, denoted by net_o.

$$\delta_{ws} = T_{ws} - net_o \tag{5-5}$$

The value for *net* for each output node, *o*, was previously calculated by the network during the feedforward operation, and is shown in Equation 5-6, where there are $n>1$ input nodes, where the n^{th} input node is our new input.

$$net_o = \sum_{i=0}^{n-1} x_i w_{io} \tag{5-6}$$

T_{ws}, can be found for each output node, by reversing the thresholding process that took place at the output node when initially feeding forward. This is calculated by finding the inverse of the asymmetric sigmoid function, Equation 4-2, and is shown in Equation 5-7, where *f(net)* is the original thresholded value that was outputted from that neuron.

$$T_{ws_o} = \frac{\log\left[\frac{f(net)_o + \delta_o}{1-\left(f(net)_o + \delta_o\right)}\right]}{k} \tag{5-7}$$

Alternatively, using the symmetric sigmoid, shown in Equation 4-3, as used in this thesis's implementation, T_{ws} is calculated similarly and is shown in Equation 5-8.

$$T_{ws_o} = \frac{\log\left[\frac{f(net)_o + \delta_o + 0.5}{0.5-\left(f(net)_o + \delta_o\right)}\right]}{k} \tag{5-8}$$

Thus, the full *single-shot-Δ_l-initialisation-rule*, used for each of the new input's connections to all of the output nodes is given in equation 5-9. Due to the use of the asymmetric sigmoid function, it is clear that at no time can the system try and set the value of the output to be outside the range $0 > (f(net) + \delta) > 1$ as this will cause an error.

$$w_{no} = \zeta\left[\overbrace{\left(\log\left[\frac{f(net)_o + \delta_o}{1-\left(f(net)_o + \delta_o\right)}\right]\Big/k\right)}^{T_{ws}} - \overbrace{\left(\sum_{i=0}^{n-1} x_i w_{io}\right)}^{net}\Big/ mx_n\right] \tag{5-9}$$

Similarly, when using the symmetric sigmoid, Equation 5-10 is used, with the requirement that the system does not attempt to set the value of the output outside the range $-0.5 > (f(net) + \delta) > 0.5$ as this will cause an error.

$$w_{no} = \zeta\left[\overbrace{\left(\log\left[\frac{f(net)_o + \delta_o + 0.5}{0.5-\left(f(net)_o + \delta_o\right)}\right]\Big/k\right)}^{T_{ws}} - \overbrace{\left(\sum_{i=0}^{n-1} x_i w_{io}\right)}^{net}\Big/ mx_n\right] \tag{5-10}$$

This updating method is best understood by seeing what is occurring diagramatically. Figure 5-3 shows an input pattern that had a weighted sum of 3.0 calculated at the output node. This was fed through the symmetric sigmoid function, finding the output value 0.47. The correct output, however, was -0.358. The correct weight for the new input node is calculated by feeding this target back through the inverse of the sigmoid function finding the value -1.8. Therefore, the weight on the new node is the difference of the correct value and the actual weighted sum, -4.8, which is the required weight for the new input.

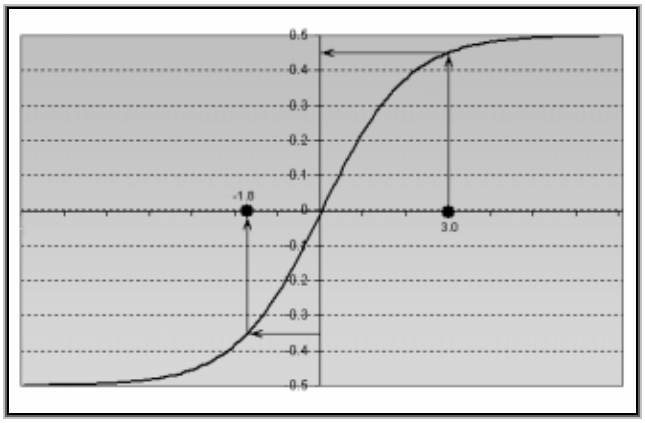


Figure 5-3: Example of the single-step-Δ-initialisation-rule shown diagramatically. This shows the principle behind what the learning rule is calculating. The dot at 3.0 is the incorrect weighted sum producing the output from the network of 0.47. The reward for this pattern was -0.358 which means that the inverse of the sigmoid function finds the correct weighted sum of -1.8.

5.5.2 Discussion

This is the first method to seriously address the learning outcomes of this project. It is able to step to the correct output immediately upon receiving new expert knowledge, allowing the system to learn initial estimates significantly faster. This allows for online learning. Additionally, unlike RM_w, it can also continue to learn a finer representation of a relationship between features. However, it is still limited to only being capable of finding linear relationships. While many problems are linearly separable, it cannot be assumed that all MCRDR knowledge maps will only contain such relationships. Therefore, further work was required to develop a method for finding all types of dynamic contexts.

5.6 $RM_{bp(\varepsilon)}$: Basic Non-Linear Technique

To address the problem of finding non-linear solutions, a system using multiple layers of neurons is required. This, however, is a distinctly more difficult problem, due to the full interconnection of inputs in the hidden layer. Therefore, when new inputs are added they need to be connected to all the existing hidden nodes. Therefore, the new node will alter already learned patterns held in those older hidden nodes. A number of '*tricks*' were developed to reduce the interconnection effects, which have been incorporated into the following method, $RM_{bp(\Delta)}$. This method, $RM_{bp(\varepsilon)}$, is the basic version using the standard backpropagation algorithm.

5.6.1 Implementation

In the experiments used in this thesis the network developed comprised of a single fully connected hidden layer. As with the other perceptron based methods it is using a symmetric sigmoidal thresholding function with a bias term. When new input nodes are added to the system this method simply assigns small random weights, symbolised by ε, for the new connections. If these connections altered existing correct hidden nodes then this is simply accepted.

It should also be noted that as new input nodes are added then new hidden nodes are also required. This is because the additional inputs are increasing the size of the input space, thus, the number of potential patterns that need to be learned. Therefore, in this implementation a new parameter representing the *percentage ratio*, *p*, of hidden nodes to input nodes is used to determine when and how many hidden nodes are added. For instance, a ratio of 100% means a new hidden node is created for every additional input node.

In this implementation, when new hidden nodes are added, the connections from the old input nodes and any new input nodes are given a small random value. Likewise, the connections from the new hidden node to each of the output nodes are also given a small random value. Just as with the adding of input nodes, this process can affect all the previously learned patterns and temporarily reduce the system's effectiveness. These issues are addressed more fully in the subsequent system, $RM_{bp(\Delta)}$.

5.6.2 Discussion

This method is still expected to learn, just more slowly than the linear based approach. The slower learning is especially expected during the early stages of training when many new input nodes are added. Once the MCRDR KB is nearing a more stable size the system should start learning nonlinear relationships quite effectively. This method is clearly incapable of learning in an online environment due to its slow initial learning. Even though it is not expected to learn quickly it has been included as a means of comparing the more refined version, $RM_{bp(\Delta)}$, to show the effectiveness of the adjustments made.

5.7 $RM_{bp(\Delta)}$: Advanced Non-Linear Technique[20]

This method is one of the primary techniques developed in this thesis. It aims at addressing the combined problems from the previous implementations. Firstly, it needs to allow for non-linear relationships which $RM_{l(\Delta)}$ was unable to accomplish, without losing its fast initial learning. Secondly, it is designed to eliminate the effect new input and hidden nodes have on already trained hidden nodes. The resulting methodology is a powerful amalgamation, allowing for fast online learning and generalisation.

5.7.1 Implementation

The approach taken in this implementation captures the initial information by directly calculating the required weight that provides us with the correct *weighted-sum* without the need for a training period in a similar way as $RM_{l(\Delta)}$. When applying this weight, however, it must be done in a way that does not affect any of the already learnt cases. Therefore, no changes can be made to weights connecting the hidden layer to the output layer. This only leaves the connections from the input to hidden layers to carry the weight adjustments. However, due to the thresholding that is performed at the hidden layer it is entirely possible that weights on the arcs from the input layer to hidden layer cannot be increased enough to fully reduce the error and give the required result.

These restrictions meant that the network structure needed to be altered by adding shortcut connections from any newly created input node directly to each output node and using these connections to carry the entire weight adjustment required to gain the desired set of results. Figure 5-4 shows the network topology used. This system allows the network to adjust immediately when a new feature is created and yet still provide the ability to learn adjustments to the new node's relationships with other nodes through the underlying network.

[20] This version of *RM* was previously published in *Rated MCRDR: Finding non-Linear Relationships Between Classifications in MCRDR*, by *Dazeley and Kang* in *2003* and *An Augmented Hybrid System for Document Classification and Rating*, by *Dazeley and Kang* in *2004* with the notation RM.

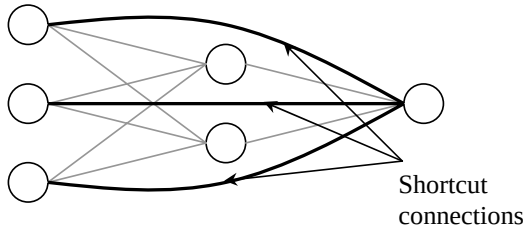


Figure 5-4: Network structure of $RM_{bp(\Delta)}$ showing a single hidden layer network with shortcut connections directly connecting the input nodes to the output nodes.

One potentially enabling result of using a topology such as this, is that we effectively now have two networks. The shortcut network is essentially a single layer network capable of learning linear functions, and thus, linear relationships, very quickly, while the two layer network underneath allows the system to still be able to learn non-linear relationships. Thus, the combination of these networks allows the system to learn a linear function quickly while still, over time, being able to derive the more complex non-linear functions.

Like with $RM_{bp(\varepsilon)}$, additional hidden nodes still need to be added in parallel with the added input nodes. Likewise, these need to be assigned an initial weight. Therefore, when adding a new input and, zero or more, hidden nodes, new connections must also be added in the following places:

- from the new input node to all of the old hidden nodes.
- from all input nodes, new and old, to each of the new hidden nodes, if any.
- from each of the new hidden nodes, if any, to all of the output nodes.
- the shortcut connections from the new input node to all of the output nodes.

The process for adding nodes and connections is illustrated in Figure 5-5. Each of these different groups of new connections requires particular start up values. First, the new connections from the new input node to all the old hidden nodes should be set to zero so that they have no immediate effect on existing relationships. If the new input node is included in particular patterns that already exist then this will be learned over time. If new hidden nodes have also been added then the connections from them to the output nodes should also be set to zero for the same reason. In order for these connections to be trained, the output from the new hidden nodes must be non-zero. Thus, the new connections from all the input nodes to the new hidden nodes, and their biases, are given random values. Finally, the new shortcut connections are given a value calculated using the *single-step-Δ_{bp}-initialisation-rule*.

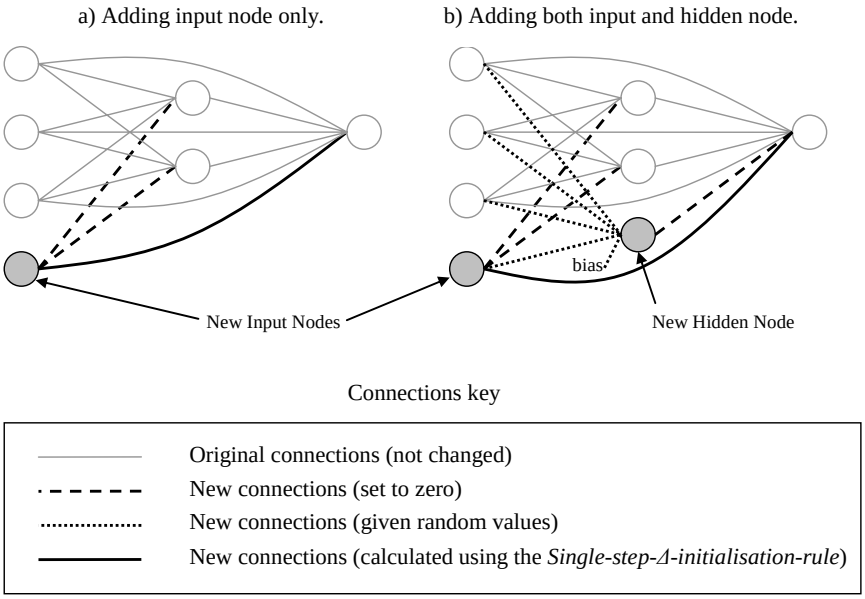


Figure 5-5: Process used for adding new input and hidden nodes in $RM_{bp(\Delta)}$. (a) shows how inputs are added by themselves. (b) shows how input and hidden nodes are added simultaneously

5.7.1.1 The Single-Step-Δ_{bp}-Initialisation-Rule

The *single-step-Δ_{bp}-initialisation-rule* is essentially the same as the *single-step-Δ_{l}-initialisation-rule* (5.5.1.1), both in derivation and use. The only difference is the weighted-sum calculation for the actual weighted-sum. In the Δ_{bp} version this must also incorporate the weighted sum from the shortcut connections and is shown in Equation 5-11: where, *i* is the individual input layer nodes and *n* is the number of inputs; *h* is the individual hidden layer nodes and *q* is the number of hidden nodes; and, *o* is the individual output layer nodes.

$$net_o = \left(\sum_{i=0}^{n-1} x_i w_{io}\right) + \left(\sum_{h=0}^{q} x_h w_{ho}\right) \tag{5-11}$$

Therefore, the final Δ_{bp} rule can easily incorporate the new *actual weighted sum* and it is shown in its general form in Equation 5-12.

$$w_{no} = \zeta\left[\overbrace{\left(\log\left[\frac{f(net)_o + \delta_o}{1-\left(f(net)_o + \delta_o\right)}\right] \Big/ k\right)}^{T_{ws}} - \overbrace{\left[\left(\sum_{i=0}^{n-1} x_i w_{io}\right) + \left(\sum_{h=0}^{q} x_h w_{ho}\right)\right]}^{net} \Big/ mx_n\right] \tag{5-12}$$

Due to the use of the symmetric sigmoid in this thesis the actual calculations were made using Equation 5-13.

$$w_{no} = \zeta \left[\overbrace{\left(\log \left[\frac{f(net)_o + \delta_o + 0.5}{0.5 - \left(f(net)_o + \delta_o \right)} \right] \Big/ k \right)}^{T_{ws}} - \left[\overbrace{\left(\sum_{i=0}^{n-1} x_i w_{io} \right) + \left(\sum_{h=0}^{q} x_h w_{ho} \right)}^{net} \right] \Big/ m x_n \right] \quad (5\text{-}13)$$

As can be seen in the above equations, this method has also incorporated the use of the step-distance modifier, ζ.

5.7.2 Discussion

This is one of the main methods developed that is expected to learn an effective non-linear solution quickly, while maintaining the ability to generalise. It incorporates the stepping function combined with a multilayer network. Additionally, it carefully sets connection weights to significantly reduce any effect from new nodes on older already semi-trained nodes. One advantage of the shortcut connections is that the system can learn a linear function as quickly as $RM_{l(\Delta)}$, method and then learn a refined nonlinear value via the non-linear portion of the network. It is expected that there is no reason the same idea cannot be further extended to a two hidden layer network. However, this was not tested in this thesis.

The primary problem with this method would be in successful generalisation during prediction tasks, where only a few inputs fire and these are all strongly excitory or inhibitory instead of a combination of both. In such a situation the system can produce a large positive or negative value, well outside the range of a reasonable result. This was mostly solved by using the RPA integration method. This meant that significantly more inputs fired considerably reducing the risk of a compounded result. Other techniques involving variations to reward structures can also limit this problem. This issue is explored further in the following initial results chapter.

5.8 RM_{rbf}: Basic Radial Basis Function Technique[21]

The second approach to finding nonlinear solutions was to use the radial basis function ideas discussed in section 4.4. It is important to recall here that when adding new inputs for the backpropagation method a weight needed to be selected that minimised the effect on already semi-learned hidden nodes. The advantage, in this regard, when using an RBF, is that when we add a new input node we can also add a hidden node which can be used to form a hyperellipsoid function representing the new input pattern in a similar method as that of the RAN network (4.5.5). This significantly simplifies the issues of altering values for already learned hidden nodes as the new input will not be recognised as part of their inputs. This method, along with the following algorithm, are the main alternatives to the backpropagation methods described previously. RBFs can traditionally learn faster solutions but run the risk of over fitting, and thus, may have poor generalisation.

5.8.1 Implementation

Fundamentally, this method is a standard RAN based RBF network. It uses the symmetric sigmoid thresholding function, Equation 4-3, for the output nodes and the hidden layer uses the Gaussian function, Equation 4-15, with a Euclidean distance measure, Equation 4-16. Like in Platt's (1991) RAN network, new hidden nodes can be added to capture new and unique patterns.

In this implementation, however, hidden nodes were only added when a new input was added. The new hidden node was given the hyperellipsoid function containing all the inputs for this new pattern. Therefore, the weight of a connection between each input node and the new hidden node is set to the same value as the activation value that was received. All the connections from the new input to the old hidden nodes are given the weight of *0*. Thus, the new node will only have a small effect on any already learned nodes. Finally, the coefficient, λ, of the newly created basis function is calculated in an identical way as the new weights in the $RM_{l(\Delta)}$ method using the *single-step-Δ_l-initialisation-rule*.

[21] This version of *RM* was previously published in *Detecting the Knowledge Frontier: An Error Predicting Knowledge Based System*, by *Dazeley and Kang* in *2004* and *An Online Classification and Prediction Hybrid System for Knowledge Discovery in Databases*, by *Dazeley and Kang* in *2004* with the notation *RM*.

5.8.2 Discussion

This method, as will be discussed in chapter 6, has shown a lot of ability at learning online. Through adjustment to the width of the Gaussian function, it has also been able to generalise effectively. Due to this method only adding hidden nodes when input nodes are added the usual problems of over fitting are prevented. However, it also prevents hidden nodes from being added for unique patterns that still may occur even when a new input was not added. This was found to significantly limit the system's ability to learn the relationships between features on many problems.

5.9 RM_{rbf+}: Advanced Radial Basis Function Technique

The final method developed is really an extension to the previous technique and uses the notation, RM_{rbf+}. The primary aim of this method was to improve on some of the issues that arose from the previous implementation, such as being able to allocate resources when inputs are not added and more directly addressing the issues that arise from the changing input space. This method was developed in the expectation that it would achieve fast and generalised learning and could outperform the backpropagation methods previously developed.

5.9.1 Implementation

Beginning with the implementation described for the RM_{rbf} method, this technique addresses two main issues that arose during testing. The first is the need to be able to occasionally add new hidden nodes even though no input has been added. A subsequent issue concerning the prevention of the leaking effect caused when new inputs are added, which can still have a small affect on previously identified patterns, through the implementation of input keys. The solutions for these issues are detailed in the following subsections.

5.9.1.1 Handling Resource Allocation

The approach taken for the allocation of hidden nodes when no input was added was relatively simple. A threshold, θ, was selected that would indicate when an error was too large for the system to simply train towards. Once fired, the system investigated each hidden node to find any that fitted the input pattern

exactly, in other words a hidden node with a Euclidean distance of zero. If none were found, then it is known that this is a new previously unseen pattern that due to being outside the threshold deserves its own hidden neuron.

The difficulty in this method is that the value for the threshold needs to be selected carefully. A threshold set too small will result in over fitting and, therefore, very poor generalisation, while a threshold set too large will have little effect on the method's learning ability. The best setting for this parameter was found through repeated testing.

5.9.1.2 Input Keys

The original advantage in using an RBF network was that we could more easily reduce the effect of a newly discovered pattern on the already trained hidden neurons. However, they do still have some effect in certain circumstances. Imagine a situation where *case A* is correctly classified and rated. Then *case B* is received, which produces exactly the same result, but, this time it is not correct. A new rule is added creating a new input and hidden nodes in the network. Later still, *case A* happens to be re-presented. This time it still produces the same classification and input sequence to the network, however, the new hidden node created to cater for case B will also fire because the pattern is only wrong by one input. Obviously, it does not fire at full strength but it does still produce a value which perturbs the current result for case A. Therefore, a case that was fully trained now produces an incorrect value. Although, generally a small error, it can still cause problems.

This was resolved by using what has been referred to as input keys. Basically, when a new hidden node is created in parallel with a new input node then a record is kept of the pair's association. Now when future cases are processed hidden nodes can only fire if either their associated key input or an input created after its key input fired. Therefore, each hidden node cannot fire and disturb any patterns learned prior to its creation; instead it can only affect nodes added after it was created. This is possible because the ones created after it, were done with a full understanding of what was currently in the network. This was found to significantly aid the system's stability as new nodes no longer had an affect on older ones.

5.9.2 Discussion

This method represents an interesting application of an RBF network and has been quite effective in many of the experiments performed as part of this thesis. This method is expected to learn quickly with a reasonable generalisation. It is also expected to be able to capture non-linear relationships quickly. However, prior to testing it was unknown whether it would outperform the $RM_{bp(\Delta)}$ method developed. One of the primary difficulties with this method is the number of parameters that are now required, many of which have a significant effect on the system's performance. This difficulty has made testing time consuming and difficult.

5.10 Other Network Types and Function Fitting Methods

The above implementations have been based primarily on the standard backpropagation, RBF, and RAN neural networks. From discussion with colleagues at the university and at the conferences attended, many have asked about other possible methods such as Cascade Correlation (CC) or self organising networks, such as Kohonen networks, and even non-network based methods such as Support Vector Machines (SVM). Many of the methods that have been considered were briefly described in chapter 4. However, these investigations confirmed that these systems were not suitable.

The primary problem for all of these methods is that any method that could be devised for assigning additional input neurons resulted in an unwieldy algorithm at best and completely impossible at worst. For instance, CC trains a hidden neuron until stable then sets it in stone, freezing its weights, and adds a new hidden node for further refinement. However, a given hidden node will never be stable if the input space is constantly changing. Additionally, many cases are only seen once preventing the training of subsequent hidden nodes. No means of adjusting the algorithm could be found that could handle the required environment and still keep any of the CC algorithm intact. Another issue which occurred with methods such as the *upstart, divide-and-conquer* and *tiling algorithms* were that they required the entire dataset at the outset and could not operate in an online environment.

A version of the resource allocating network (RAN), called adaptive response function neurons (ARFNs), described in section 4.5.5.1, had been a particularly promising method for application to the RM system. This was because if the ARFNs were used as input nodes then the inputs could be trained to only respond to particular input levels. Basically, it was the only existing method found that allowed for the adding of input nodes. However, the ARFN requires continuous inputs and MCRDR could only provide discrete values. This was why the DRPA integration method had been developed, to find a means of providing a more continuous-like input. This, however, failed because the fractional values are always a portion of *on* as opposed to a sensor type reading for which the neurons were created.

5.11 Summary

This chapter was the primary methodology chapter, describing the RM system developed in detail. Initially it described the problem of a lack of cohesion with the MCRDR methodology and that inferencing occurs down separate paths with no reference to what may be the conclusion of other paths. Also, ideas concerning the philosophy of knowledge were revisited, identifying clearly the aims of the algorithm being developed in this thesis.

The main methodology was developed over a number of sections. The system overview described the overall design of the system and a brief description of the two components of the hybrid system. The section also described how the two components could be integrated in five possible ways. These are by no means the only integration methods possible but they do represent the methods tested in this thesis. The following sections described the seven different methods developed. Some of these techniques, such as RM_w and $RM_{l(\varepsilon)}$ were very simplistic. These were used to show, in the following chapter, that the more complicated techniques, such as $RM_{bp(\Delta)}$ and RM_{rbf+} were required.

6

Classification and Prediction

It sounded to him like a riddle, and he was never much good at riddles, being a Bear of Very Little Brain (Milne 1926, p67).

The previous chapter presented the justification and development of the RM algorithm. In developing the algorithm, a number of alternative neural network implementations were presented. This chapter details a large number of test results designed to accomplish two tasks. Firstly, the chapter aims to find which of the presented methods developed produces the optimal learning and generalisation abilities for both classification and prediction tasks. Secondly, the chapter intends to illustrate how RM compares against MCRDR and neural networks by themselves. Thus, the aim is to show how RM learns as fast as MCRDR on presented cases, yet maintains the generalisability of a neural network on unseen cases.

This chapter has been divided into three distinct sections. First, it will discuss the classification and prediction experiments being performed. This will consist of a discussion of the simulated experts previously used in RDR based research and the development of the experts used for the experiments performed as part of this chapter, along with the datasets used. The second section will present a collection of results that provide a comparison of the proposed algorithms presented in the previous chapter. This comparison will then be discussed and used for the selection of the primary methods used for the remainder of this thesis. The final section will provide additional results aimed at showing how the selected RM method compares against MCRDR and a neural network in the classification domain and a neural network in a prediction domain. Additionally, due to RM specifically targeting an online environment, test results will also be shown indicating how they progress as cases are presented individually over time.

6.1 Experimental Method

There is a large collection of results presented in this chapter, and all were carried out in similar ways. Each test used 10 different randomisations of the relevant datasets with the average presented in this chapter. In many of the tests there are too many parameters to define them all. Therefore, Appendix D has a table giving all the relevant parameters used on all the experiments. This chapter consists of two primary types of tasks. The first is a standard *classification* task, while the second investigates the *prediction* capability of the methods. Each task consists of two types of tests: *generalisation* and *online learning*. The first, *generalisation* test, divides each dataset into ten equal sized groups. Results are presented where $9/10^{ths}$ of the dataset are used for training and $1/10^{th}$ for testing. The size of the training set is then reduced incrementally in steps of $1/10^{th}$, down to $1/10^{th}$. The same $1/10^{th}$ set is always used as the test set. The second, *online-classification* test, investigates how the methods can correctly classify cases or predict values over time. In this test the entire dataset is broken up into smaller blocks, each $1/50^{th}$ of the original dataset, and passed through the system one group at a time. The system's performance is recorded after each group, showing how fast the system learns for each new batch of cases.

This section will briefly discuss the use of simulated expertise in RDR research. This will be followed by a detailed description of the simulated experts used in this thesis and how they were used in generating the MCRDR and RM results. Next, a detailed description is provided of the two types of experiments run for each type of task. Finally, this section will describe the datasets used to evaluate the RM methods developed.

6.1.1 Simulated Expertise

One of the greatest difficulties in KA and KBSs research is how to evaluate the methodologies developed (Compton 2000). This is because any evaluation requires the use of people to actually build the system. Furthermore, the same experts would ideally be used to compare two systems. Clearly, for this comparison to be meaningful they should also be built around the same domain. Therefore, the expert will have accrued some experience when building the first system and would provide better quality knowledge for the second system built.

Even if the results of such a comparison could be gained in a meaningful way then finding an expert to provide their time to build two expert systems is next to impossible and would be prohibitively expensive.

One attempt to develop a method of comparatively testing KA development tools is based on the Sisyphus projects (Linster 1992). These projects consist of a complete specification on paper of a problem. KBS developers can then trial their systems on the projects providing a base for comparison. However, the results are far from conclusive. Due to all the required knowledge being made available, no actual KA is required. Therefore, such test beds are more likely to only compare a methodology's problem solving method development ability.

The solution to this evaluation problem taken by the majority of RDR based research has been to build a simulated expert, from which knowledge can be acquired. Generally, this has been accomplished by first building a KB using another KBS. The KA tool being tested can then use the simulated expert as its source of knowledge. This allows the method being tested to build a new KBS which should have the same level of competence as the original KBS. Clearly, this approach has certain advantages, such as the experimenter having complete control over the variables used and being able to perform multiple tests, allowing the method's further refinement (Compton et al. 1995). Such a method also allows the direct comparison of different methods more easily.

While there are issues with this approach, such as the data model defined in the original KBS is easily extracted in the secondary KBS. Nevertheless, the problems are generally minor. Obviously, testing with a simulated expert can never provide the same level of analysis that testing with real human experts can supply, but they do provide an excellent initial testing ground. Such testing can be used during methodological development, and then once the method is fully refined, it can then be tested in a real world domain.

This is the approach that has been taken in this thesis. For the majority of testing, a set of simulated experts has been used to both refine and justify the majority of claims in this thesis. The resulting method is then used in the *MonClassifier* monitoring and classification system to illustrate that the claims made using simulated experts hold when using human expertise. This section will discuss the four simulated experts created for the tests performed in this chapter.

6.1.1.1 C4.5 Simulated Expert

The first simulated expert created is fundamentally very similar to what has been described in other RDR research such as the one used by Compton et al's (1996). The only purpose of the simulated user is to select which differences in a difference list are the primary ones. In other words, it uses its KB to select the symbols that will make up the new KB. In this thesis, the first simulated expert uses C4.5 (Quinlan 1993) to generate the simulated expert's knowledge base. This is done by presenting the dataset to the C4.5 decision tree generation software. This was done with no modifiers or tree pruning, because we are not after the most efficient decision tree just an effective one. The decision tree generated is then loaded into the RM testing environment.

The loaded tree views each case presented and classifies it, just like our KB under development. If the KB being constructed evaluates an incorrect classification then the simulated expert KB tree is used, by finding rules that led to the correct classification. Any attributes that appear in the different list and are in the path of rules from root to concluding leaf node in the simulated expert, can potentially be selected.

There are numerous ways this can be done. In order to limit the quantity of testing required to a manageable level, this project only used settings that have been classed as reasonable in other studies. Firstly, attributes were selected from the top down, that is, attributes closest to the root were always selected first, and those nearest the leaf were selected last. Secondly, the simulated expert always tried to select two attributes for each rule. In situations where there is only one possible attribute then only one was selected. This is similar to the *'smartest'* expert created by Compton et al (1995).

This expert was found to work particularly effectively with any dataset that can first be run through the C4.5 method. It also allows a direct comparison of work done on prudence analysis in chapter 7, against Compton et al's (1996) prudence system. It should be noted; however, that the performances of the RM methods vary a little with changes to the number of attributes selected just as the performance of MCRDR is altered. These differences were generally relatively small, and irrelevant when comparing RM against MCRDR because the same settings were always used for both.

6.1.1.2 Linear Multi-Class Simulated Expert

The fundamental problem with the above simulated expert is that it requires an induction system, such as C4.5, to generate a complete KB prior to its use. This is a problem because no such system is available that can create such a tree for a multiple classification domain. In C4.5, while there can be many conclusions, each case can only have a single associated class. However, the system being developed in this thesis is primarily targeting domains with multiple classification domains. Therefore, a second simulated expert was created specifically designed for handling a particular multiple classification based dataset (6.1.4).

This heuristic based simulated expert uses a randomly generated table of values, representing the level that each possible attribute, $a \in A$, in the environment contributes to each possible class, $c \in C$. Two rules were used in generating the expert: each attribute contributes to one class positively (1 to 3) and one class negatively (-1 to -2) and the remaining classes are given a value of zero; secondly, each class has one positive and one negative attribute from every $|C|$ number of possible attributes. Thus, if there are 6 class and 12 attributes then each class will have two attributes providing a positive and two suppling a negative influence, the rest have no effect and have the value zero. An example of an expert's attribute table used is shown in Table 6-1.

	a	b	c	d	e	f	g	h	i	j	k	l
C1	0	0	-1	3	0	0	0	0	0	0	-1	3
C2	0	0	0	-2	2	0	0	-2	0	0	1	0
C3	0	-2	1	0	0	0	0	0	0	1	0	-1
C4	-1	3	0	0	0	0	1	0	-1	0	0	0
C5	0	0	0	0	-2	2	-2	0	2	0	0	0
C6	2	0	0	0	0	-2	0	1	0	-2	0	0

Table 6-1: Example of a randomly generated table used by the linear multi-class simulated expert. Attributes a - l are identified across the top, and the possible classes C1 – C6 down the left side.

When a case is presented to the expert it is tested to see which class it belongs to by adding all the associated values for each attribute in each class. The expert will then classify the case according to which classes provided a positive total. The reason for setting up the expert in this way was to ensure that every case presented to the expert would be classified in at least one or more

classes. When creating a new rule, the expert selects the attribute from the difference list that distinguishes the new case from the cornerstone case to the greatest degree. This was achieved by locating the most significant attribute, either positively or negatively, that appeared in the difference list.

Table 6-2, gives two example cases each with 4 attributes where the method for calculating the case's appropriate classification can be seen. Each attribute contributes a value for the class. The simulated expert's resulting classification for both of these cases are *{1, 4, 6}* for *case A* and *{2, 3, 6}* for *case B*. This expert was used for the multiple conclusion dataset.

Case A = *{a, b, c, d}*							Case B = *{a, c, e, g}*						
Attributes	Classifications						Attributes	Classifications					
	1	2	3	4	5	6		1	2	3	4	5	6
a	0	0	0	-1	0	2	*a*	0	0	0	-1	0	2
b	0	0	-2	3	0	0	*c*	-1	0	1	0	0	0
c	-1	0	1	0	0	0	*e*	0	2	0	0	-2	0
d	3	-2	0	0	0	0	*g*	0	0	0	1	-2	0
Total	2	-2	-1	2	0	2	Total	-1	2	1	0	-4	2
Classified	✓	✗	✗	✓	✗	✓	Classified	✗	✓	✓	✗	✗	✓

Table 6-2: Two example cases being evaluated by the linear multi-class simulated expert. Each case has 6 classes in which they can be classified. They are only classed as being in that class if the total for the case's attributes are greater than zero.

6.1.1.3 Non-Linear Multi-Class Simulated Expert

The above simulated expert was effective but only tested RM in linearly separable problems. It was linear because each attribute always affected a particular attribute in the same way. It cannot be assumed that all problems encountered are as simplistic. Therefore, a second multi-class simulated expert was developed, to test the method's ability in a non-linear domain. A non-linear expert needs the attributes' contribution to classifications to vary according to what other attributes were in the case. This was achieved by selecting an even number of attributes and pairing them together for each of the classes. Once paired, they were given an increasing absolute value. Additionally, alternate pairs had their sign changed. This can best be understood by investigating the example shown in Table 6-3. Here it can be seen that for the class *C1*, the attribute pairs *{b, j}, {f, h}* and *{d, j}* have a positive influence, while *{a, l}, {h, i}* and *{a, k}* have a negative influence.

	1		-1		2		-2		3		-3	
C1	b	j	a	l	f	h	h	i	d	j	a	k
C2	g	b	c	f	e	h	a	b	k	d	g	k
C3	i	d	e	b	g	i	k	l	j	a	c	f
C4	l	a	c	i	j	a	i	l	f	h	j	a
C5	k	g	b	f	d	g	j	f	b	c	a	e
C6	c	l	h	j	a	c	j	b	g	k	d	e

Table 6-3: Example of a randomly generated table of attribute pairs. The top numbers represent the positive or negative values for the pairs. Each class in this example has six pairs.

Now when a case is presented to the expert it is tested to see which class it belongs to in much the same way as in the linear expert, by adding all the associated values for each attribute and attribute-pairs in each class. The expert will, once again, classify the case according to which classes provided a positive total. When creating a new rule, the expert selects the attribute or pair of attributes, from the difference list that distinguishes the new case from the cornerstone case to the greatest degree.

Table 6-4, gives the same two example cases, as with the linear expert, where each case has 4 attributes. As before each attribute contributes a base value for the class, and attribute pairs add or subtract the extra values. The simulated expert's resulting classification for both of these cases are *{1, 4, 5, 6}* for *case A* and *{2, 3, 6}* for *case B*. This expert was found to be quite versatile allowing both simple and more complicated tests to be performed.

Case A = *{a, b, c, d}*							Case B = *{a, c, e, g}*						
Attributes	Classifications						Attributes	Classifications					
	1	2	3	4	5	6		1	2	3	4	5	6
a	0	0	0	-1	0	2	*a*	0	0	0	-1	0	2
b	0	0	-2	3	0	0	*c*	-1	0	1	0	0	0
c	-1	0	1	0	0	0	*e*	0	2	0	0	-2	0
d	3	-2	0	0	0	0	*g*	0	0	0	1	-2	0
{a, c}						2	*{a, c}*						2
{a, b}		-2					*{e, a}*					-3	
{b, c}					3								
Total	2	-4	-1	2	3	4	Total	-1	2	1	0	-7	4
Classified	✓	✗	✗	✓	✓	✓	Classified	✗	✓	✓	✗	✗	✓

Table 6-4: Two example cases being evaluated by the non-linear multi-class simulated expert. Each case has 6 classes in which they can be classified. They are only classed as being in that class if the total for the case's attributes are greater than zero.

6.1.1.4 Multi-Class-Prediction Simulated Expert

While one of the most common application domains for neural networks and KBSs is in classification, ANNs can also be applied in *value prediction* problems as well. This, however, is not a common application of symbolic based AI methods. Early work on this project found, however, that the RM method also introduced a means of using expert knowledge specifically for value prediction. Testing RM in such a domain, however, was found to be problematic when using traditional datasets. This was because the datasets available do not give both symbolic knowledge and a target value instead of a classification. This could be partially resolved by assigning each classification a value. However, fundamentally this is still a classification type problem.

The approach taken in this thesis was to extend the linear multi-class simulated expert allowing a value to be provided instead of the classification. By using the Multi-Class dataset (6.1.4.2) it was possible to incorporate both symbolic expert knowledge and a rating. The symbolic knowledge could still be used to correct rules in the same way that was used by the linear multi-class simulated expert. Thus, the expert still maintains an understanding of classification at least as a means of grouping similar cases. The rating is a concept of overall importance. This can represent many different things in a real world application.

One example situation where we may have a classification and a rating could be an email system. The classification may be the folder a received email should be placed. While the rating could be an indication of how important the email would be to the user. For instance, it may learn that an email from the boss is important and give it a high rating. However, if the email was sent to a social group address then it is much less important. The application may then use this rating as an indication of whether to alert the user to the email's arrival with a dialog or just place it in a list for when the user is ready to check.

In such a system the rating would have to be learnt from watching user behaviour. For instance, the system may start by always giving alerts but if the user doesn't immediately read the email then the rating for that email would be low. Also, if they print, save or reply to the email quickly then that may raise the rating. There are many other possible applications for this approach where a

single method can both classify and calculate a rating of importance. In the experiments performed in this thesis the rating is calculated and supplied directly by the simulated expert.

To fully push the system's abilities, the rating calculated by the simulated expert would be required to generate a non-linear value across the possible classifications. The eventual implementation used for prediction tests throughout this thesis, generates an energy space across the level of class activations, giving an energy dimensionality the same as the number of classes possible. Each case is then plotted on to the energy space in order to retrieve the case's value.

First, the strength of each classification found is calculated. As previously discussed, in the basic Multi-Class simulated expert, a case was regarded as being a member of a class if its attribute value was greater than *0*. However, no consideration was made to what was the degree of membership. In this expert the degree of the case's membership is calculated as a percentage, *p*, of membership using Equation 6-1.

$$p = t^a / t^m \tag{6-1}$$

This is simply the actual calculated total, t^a, divided by the maximum possible total, t^m, for that particular class. Extending the example from Table 6-2 for case A, classification C1, the total *2* is divided by the best possible degree of membership *6,* from Table 6-1, thereby, giving a percentage, *p*, membership of *33%*. This calculation is performed for each class. Each class then has a randomly selected point of highest value, or centre, *c*, which is subtracted from the percentage and squared, Equation 6-2. This provides a value which can be regarded as a distance measure, *d*, from the centre. This distance measure can be *stretched* or *squeezed*, widening or contracting the energy patterns around a centre, by the inclusion of a width modifier, *w*.

$$d = (pw - c)^2 \tag{6-2}$$

The classes' centres are combined to represent the point of highest activation for the expert, referred to as a *peak*. Therefore, if the square root of the sum of distances is taken then the distance from this combined centre can be found. This distance can then be used to calculate a lesser value for the case's actual rating. Therefore, as a case moves away from a *peak* its *value* decreases. Any function can be used to calculate the degree of reduction in relation to distance.

In this thesis a Gaussian function was used. Equation 6-3 gives the combined function for calculating a value for each possible peak, v^p, where n is the number of classes in the dataset.

$$v^p = \frac{1}{1+e^{-0.5\left(1\Big/\sqrt{\sum_{j=0}^{n}\left(\left(\frac{t_j^a}{t_j^m}\right)w_j - c_j\right)^2}\right)}} - 0.5 \qquad (6\text{-}3)$$

Finally, it is possible to have multiple peaks in the energy space. In such a situation each class has a centre for each peak. Each peak is then calculated in the same fashion as above, resulting in a number of values, one for each peak. The expert then simply selects the highest value as the case's actual rating. This rating method is best understood by looking at a three dimensional representation shown in Figure 6-1.

The third dimension, shown by the height, illustrates the value at any particular point in the energy space. This figure shows a dataset with only two possible classes, C1 and C2, and two peaks. A three class dataset cannot be represented pictorially. The advantage of this approach is that it generates an energy pattern that is nonlinear. At no location can a straight line be drawn where values are all identical.

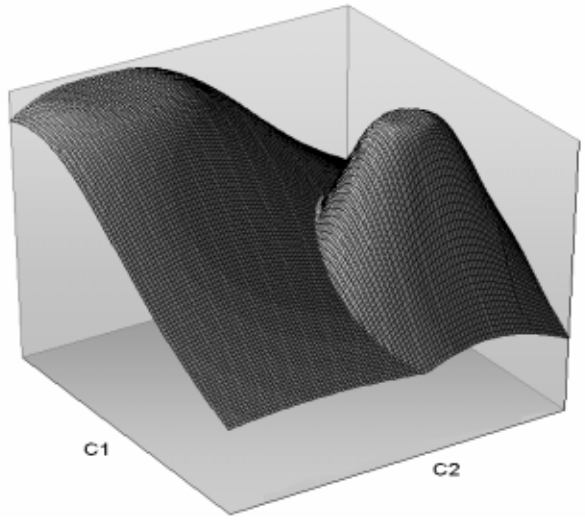


Figure 6-1: Example of a possible energy pattern used in the Multi-Class-Prediction simulated expert. This would be used for a dataset with two possible classifications. This energy pattern contains two randomly located peaks.

6.1.2 Generalisation Experiment

The first collection of experiments performed is aimed at finding how well the different methods generalise. This experiment has used the standard approach of presenting a portion of the dataset for the algorithm to use for training purposes. Then at certain intervals the remaining section of the dataset, which has never been seen, is shown to the method for testing. The procedure for this experiment is described in the Figure 6-2.

Preparation

1. Load the dataset and generate the simulated expert.
2. Divide the dataset into 10 equal sized pieces. Discard any left over cases.

Processing

3. Present a preset number of dataset segments (either 1 to 9 segments) once.
   a. Load the next case.
   b. Classify or rate using the method being tested.
   c. If correctly classified
      i. Present a reward to the method
         (MCRDR and RMw simply ignore the reward value).
      ii. Return to 3.a until all cases have been used.
   d. Otherwise, if incorrectly classified
      i. Either, present a negative reward, if performing the classification task or the expert's value if doing the prediction task, to the method. Then use the simulated expert to add the required exception rules.
         (no reclassification performed with the ANN based methods).
      ii. Return to 3.a until all cases have been used.
4. Present the final dataset segment only.
   a. Load the next case.
   b. Classify or rate using the method being tested.
   c. Either, record whether the classification found is correct, or maintain a progressive average-error between the predicted rating and the expert's rating in the prediction tasks.
   d. Return to 4.a until all cases have been used.
5. Log, either the amount of correct and incorrect classifications found in step 4, or the average difference between the predicted rating and the expert's rating in the prediction tasks.
6. Return to 3 for the next training iteration for a preset number of times.

Figure 6-2: Step-by-step description of the generalisation experiment.

Each experiment was run 10 times and averaged. Furthermore, each dataset was run using $9/10^{ths}$, down to $1/10^{th}$ for training. Regardless of the amount of training they received, they were only tested on the final $1/10^{th}$ of the dataset. This allows an analysis to be made of how a method's generalisation tapers off, as the amount of training received decreases.

6.1.3 Online Learning Experiment

The second type of experiments, were aimed at discovering how well the different methods could learn in an online environment. The purpose of these experiments was to show the speed with which a method learns a reasonable solution. This experiment, described in Figure 6-3, involved first breaking the given dataset up into a number of equal sized blocks of cases then presenting each block in turn to the system. After all of the blocks were shown, the cases were randomly reordered and partitioned again ready for the next iteration.

Preparation

1. Load the dataset and generate the simulated expert.
2. Divide the dataset into 50 equal sized pieces. Temporarily discard any left over cases.

Processing

3. Present each of the 50 data segments in turn.
    a. Load the next case.
    b. Classify or rate using method being tested.
    c. If correctly classified
        i. Present a reward to the method
            (MCRDR and RMw simply ignore the reward value).
        ii. Either, increment the number of correct classifications if performing the classification task, or keep a running average of the difference between the rating given and the expected rating if performing the prediction task.
        iii. Return to 3.a until all cases have been used.
    d. Otherwise, if incorrectly classified
        i. Either, present a negative reward if performing the classification task, or the expert's value if doing the prediction task, to the method. Then use the simulated expert to reclassify the case
            (no reclassification performed with the ANN based methods).
        ii. Either, increment the number of incorrect classifications if performing the classification task, or keep a running average of the difference between the rating given and the expected rating if performing the prediction task.
        iii. Return to 3.a until all cases have been used.
    e. Log, either the amount of correct and incorrect classifications found in step 4, or the average difference between the predicted rating and the expert's rating in the prediction tasks.
4. Re-randomise the whole dataset and then repartition into 50 segments.
5. Return to 3 for the next dataset segment for a preset number of times.

Figure 6-3: Step-by-step description of the online learning experiment.

6.1.4 Datasets

The purpose of the testing in this chapter was to select the best RM method and to show how RM performs against the existing MCRDR and ANN methodologies. It does not aim to show how RM would perform specifically in any particular area. The six datasets used were selected to provide different types of problems. The first five are standard datasets retrieved from the University of California Irvine Data Repository (Blake and Merz 1998). These five methods have used the *C4.5* based simulated expert (6.1.1.1). The third dataset is a purpose designed randomised set and is used only with the *multi-class* simulated experts (6.1.1.2, 6.1.1.3 and 6.1.1.4).

6.1.4.1 Standard Datasets

Below is a list describing each of the five dataset used from the University of California Irvine Data Repository (Blake and Merz 1998). Each has a brief description and details the number of attributes each case has, as well as how many cases and how many classifications can be determined from the dataset. Also, described is how many cases appear in each $1/10^{th}$ and $1/50^{th}$ segments.

- **CHESS** – Using the Chess end game of *King+Rook (black) vs King+Pawn (white)* on *a7*.

  36 attributes
  3196 cases
  Producing a binary classification.
  In each $1/10^{th}$ group there are 319 cases and 63 cases in each $1/50^{th}$ group.

- **TIC-TAC-TOE** (TTT) – This dataset uses the complete collection of possible terminating board configurations for Tic-Tac-Toe.

  9 attributes
  958 cases
  Producing a binary classification.
  In each $1/10^{th}$ group there are 95 cases and 19 cases in each $1/50^{th}$ group.

- **NURSERY DATABASE** – This dataset was derived from a hierarchical decision model developed to rank applications for nursery school.

  8 nominal attributes
  12960 cases
  5 classifications.
  In each $1/10^{th}$ group there are 1296 cases and 259 cases in each $1/50^{th}$ group.

- **AUDIOLOGY** – Contains a standard set of attributes used in audiology medical diagnostics.

  70 nominal-valued attributes
  200 cases
  17 classifications.
  In each $1/10^{th}$ group there are 20 cases and 4 cases in each $1/50^{th}$ group.

- **CAR EVALUATION** – Derived from a simple hierarchical decision model.

  6 attributes
  1728 cases
  4 classifications.
  In each $1/10^{th}$ group there are 172 cases and 34 cases in each $1/50^{th}$ group.

6.1.4.2 Multi-Class Dataset

The *multi-class* dataset was generated specifically for testing RM in its ideal environment. The actual dataset builds cases by randomly selecting attributes from the environment. For instance, an environment setup for the example simulated expert used in section 6.1.1.2 and 6.1.1.3 would allow for 12 possible attributes. In the tests carried out in this chapter each case selected 6 attributes, giving a possible 924 different cases. There were also 6 possible conclusions that could happen singly or together. Therefore, in each $1/10^{th}$ group there are 92 cases and 18 cases in each $1/50^{th}$ group. This dataset is used for both classification and prediction experiments. The actual dataset does not change for these two tasks but a different simulated expert is used.

6.2 Comparison of Proposed Methods

In the previous chapter seven methods RM_w, $RM_{l(\varepsilon)}$, $RM_{l(\Delta)}$, $RM_{bp(\varepsilon)}$, $RM_{bp(\Delta)}$, RM_{rbf} and RM_{rbf+} were developed. These represent a range of possible approaches to the task at hand; many are expected to perform better than others. This section is aimed at investigating each of these seven proposed algorithms to decide which performs classification and prediction tasks best. It will do this in two sub-sections: classification and prediction.

6.2.1 Classification

In the classification task we are concerned with the algorithm's ability to correctly identify which category or categories each case belongs. Each algorithm has been tested on each of the seven datasets for both their ability to generalise and to perform in an online environment. The results for these tests are discussed across the following subsections. In the tests using the multi-class, nursery, audiology and car evaluation datasets each method is assigned an output for each of the possible classes. Thus, with six classes each method has six outputs. During testing if the network delivers a value above zero then the current case is given that classification, otherwise it is not. In the tests using the chess and tic-tac-toe datasets each method only has one output. If the value is above zero then '*win*' is the identified class, otherwise the selected class is '*no-win*'.

6.2.1.1 Generalisation Overview

The ability of a method to generalise is measured by how well it can correctly classify cases during testing that it did not see during training. A method that cannot generalise will be unable to recognise anything about a new unique case. The performance of each method at generalisation in this thesis is gauged by its ability to classify unseen cases both after training and during training. The results shown in Figures 6-4 through to 6-10 show how each of the seven methods performed on each of the four datasets. Each chart shows the percentage of correct classifications, averaged over 10 trials, for each of the nine tests. To reduce the complexity of the charts, error bars have been omitted. Instead, they are shown in later figures when specific results are discussed.

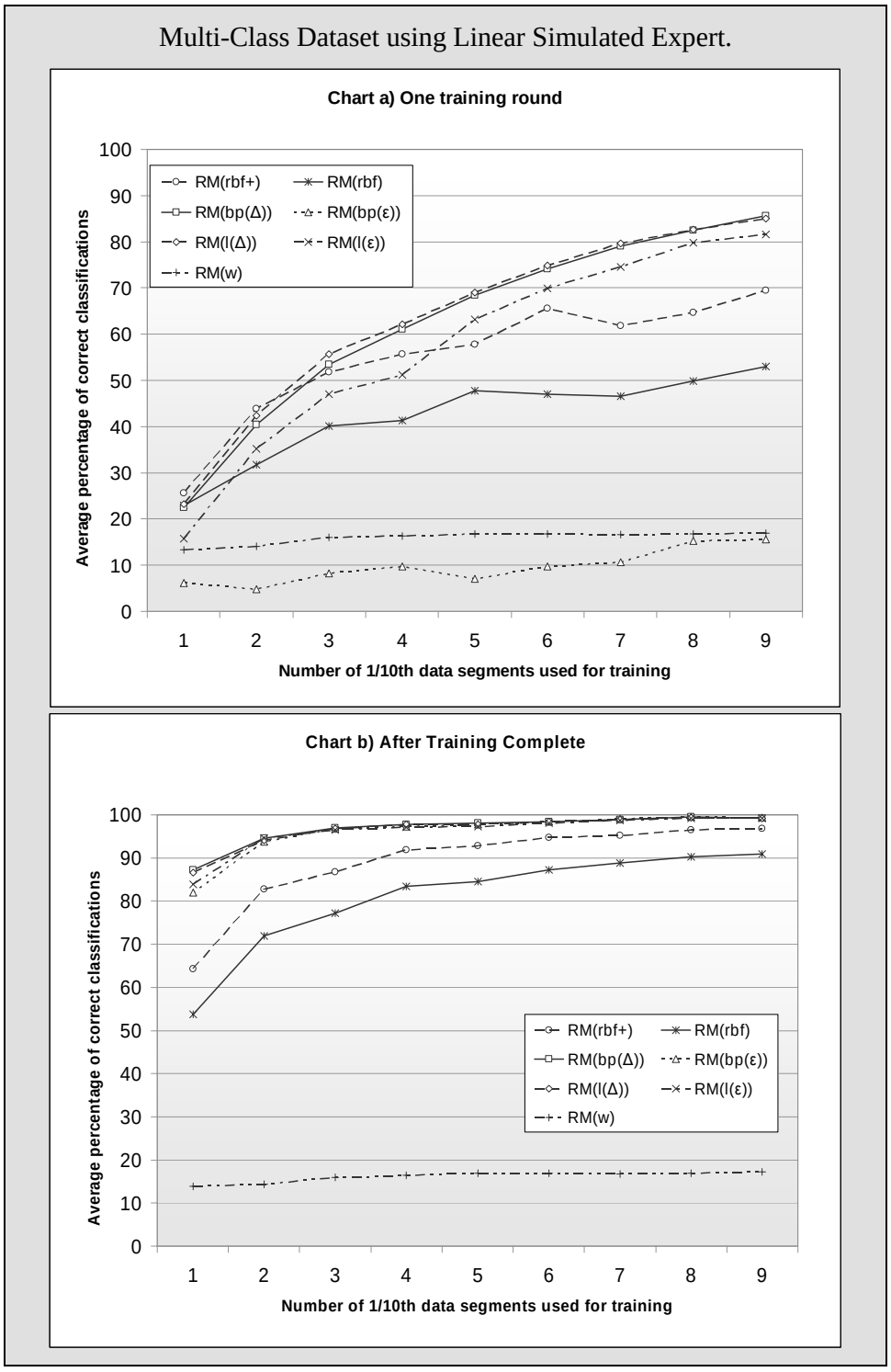


Figure 6-4 Two charts comparing how each of the seven proposed methods RM_w, $RM_{l(\varepsilon)}$, $RM_{l(\Delta)}$, $RM_{bp(\varepsilon)}$, $RM_{bp(\Delta)}$, RM_{rbf} and RM_{rbf+}, perform on the Multi-Class dataset using the Linear Multi-Class simulated expert. Chart a) shows how the methods compare after only one viewing of the training set. Chart b) shows how the methods compare after training has completed. The X-axis shows how many tenths of the dataset were used for training. The Y-axis shows the percentage of cases that were classified correctly (must get all six classes correct). All results used the last tenth for testing.

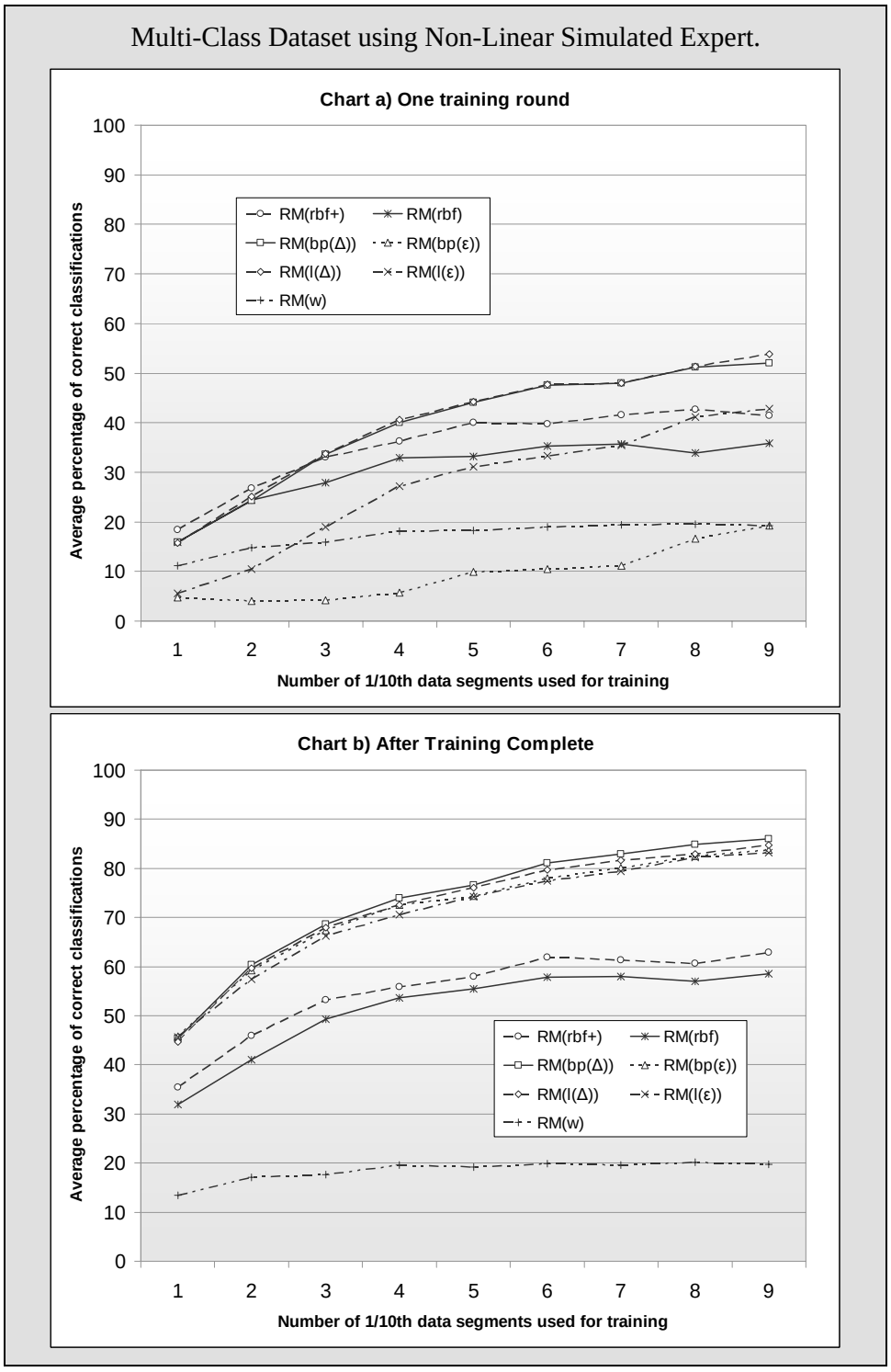


Figure 6-5 Two charts comparing how each of the seven proposed methods RMw, RMl(ε), RMl(Δ), RMbp(ε), RMbp(Δ), RMrbf and RMrbf+, perform on the Multi-Class dataset using the Non-Linear Multi-Class simulated expert. Chart a) shows how the methods compare after only one viewing of the training set. Chart b) shows how the methods compare after training has completed. The X-axis shows how many tenths of the dataset were used for training. The Y-axis shows the percentage of cases that were classified correctly (must get all six classes correct). All used the last tenth for testing.

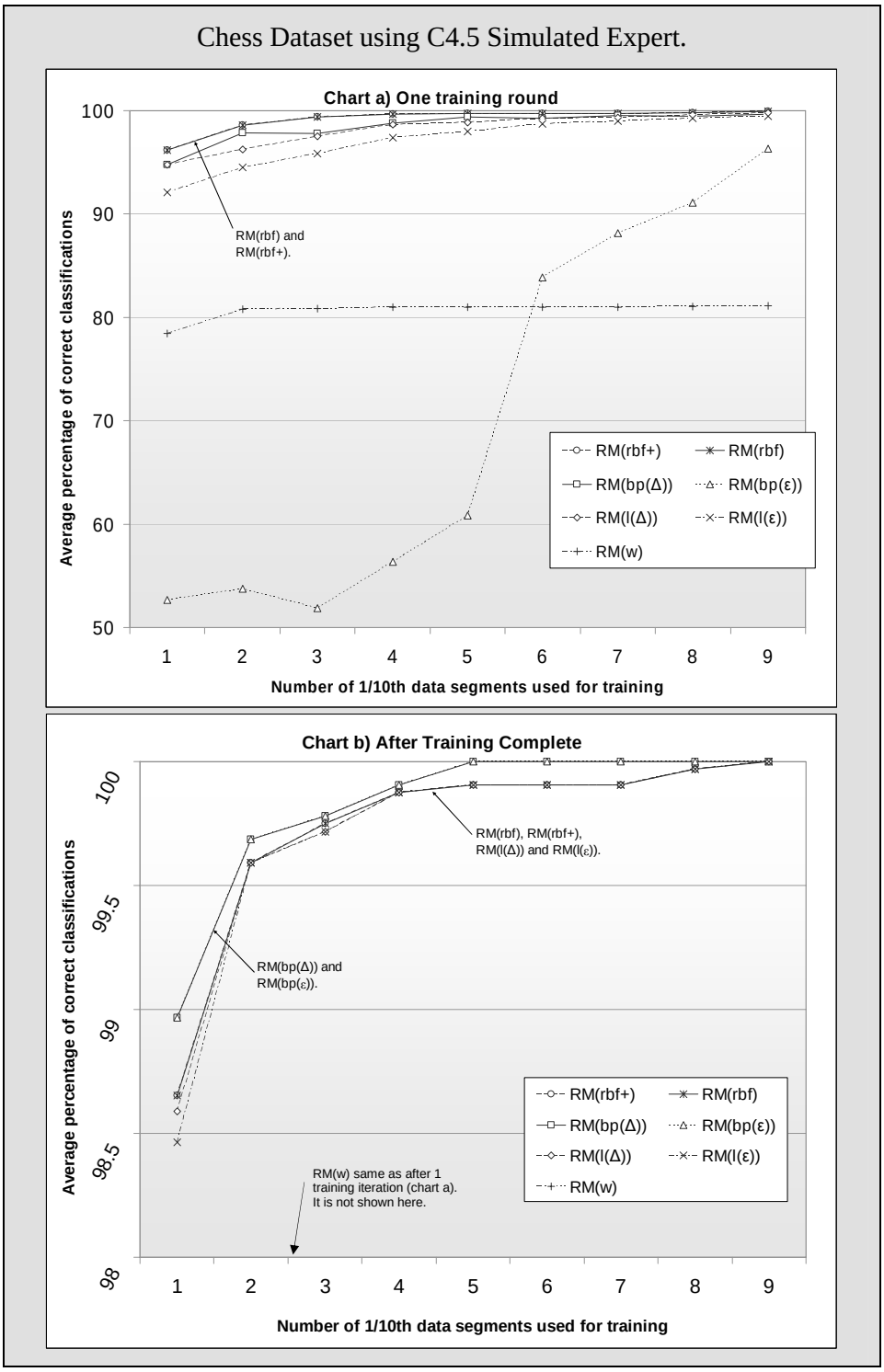


Figure 6-6: Two charts comparing how each of the seven proposed methods RM_w, $RM_{l(\varepsilon)}$, $RM_{l(\Delta)}$, $RM_{bp(\varepsilon)}$, $RM_{bp(\Delta)}$, RM_{rbf} and RM_{rbf+}, perform on the Chess dataset using the C4.5 simulated expert. Chart a) shows how the methods compare after only one viewing of the training set. Chart b) shows how the methods compare after training has completed. The X-axis shows how many tenths of the dataset were used for training. The Y-axis shows the percentage of cases that were classified correctly. All used the last tenth for testing.

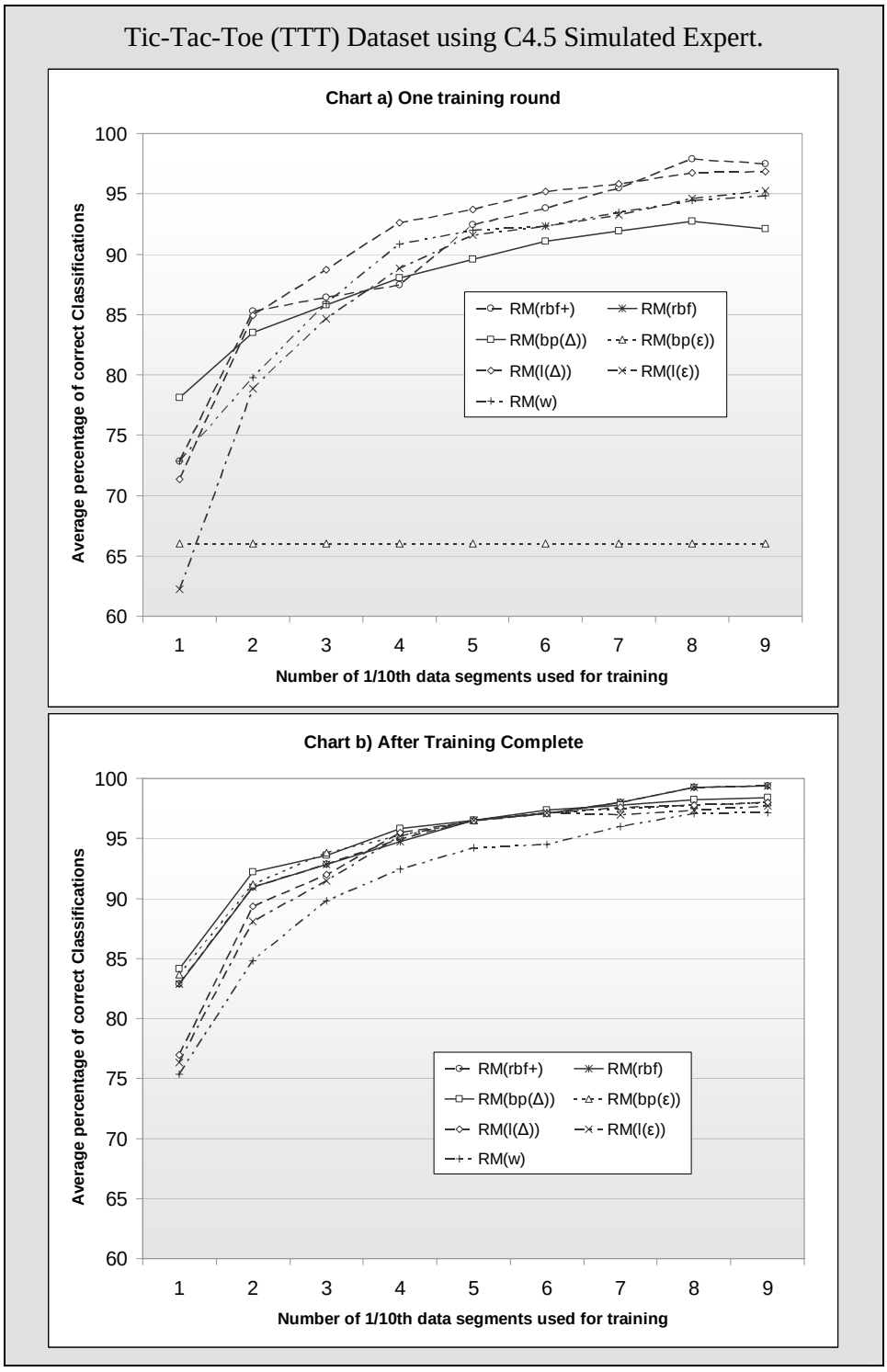


Figure 6-7: Two charts comparing how each of the seven proposed methods RM_w, $RM_{l(\varepsilon)}$, $RM_{l(\Delta)}$, $RM_{bp(\varepsilon)}$, $RM_{bp(\Delta)}$, RM_{rbf} and RM_{rbf+}, perform on the Tic-Tac-Toe dataset using the C4.5 simulated expert. Chart a) shows how the methods compare after only one viewing of the training set. Chart b) shows how the methods compare after training has completed. The X-axis shows how many tenths of the dataset were used for training. The Y-axis shows the percentage of cases that where classified correctly. All used the last tenth for testing.

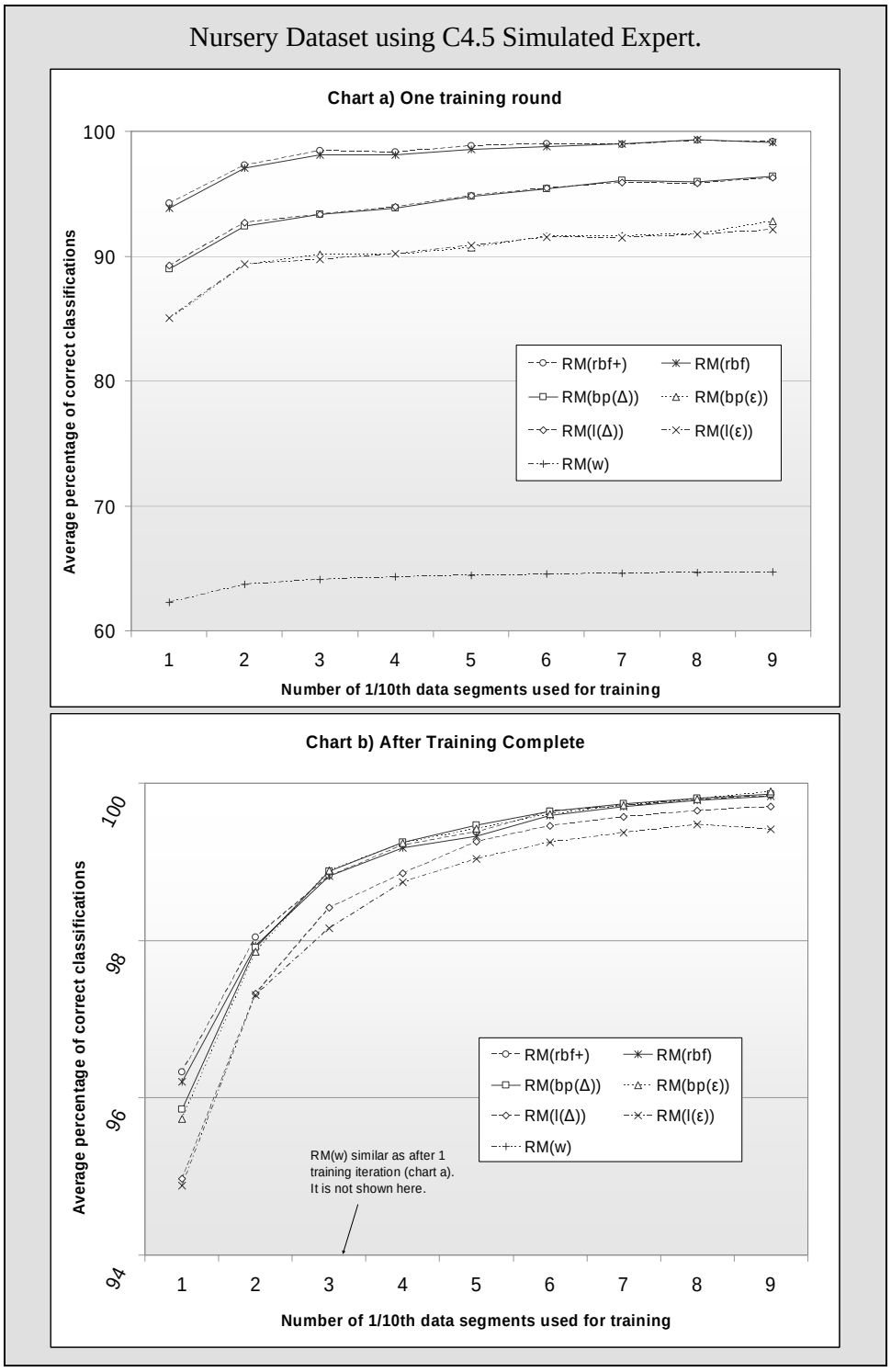


Figure 6-8 Two charts comparing how each of the seven proposed methods RM_w, $RM_{l(\varepsilon)}$, $RM_{l(\Delta)}$, $RM_{bp(\varepsilon)}$, $RM_{bp(\Delta)}$, RM_{rbf} and RM_{rbf+}, perform on the Nursery dataset using the C4.5 simulated expert. Chart a) shows how the methods compare after only one viewing of the training set. Chart b) shows how the methods compare after training has completed. The X-axis shows how many tenths of the dataset were used for training. The Y-axis shows the percentage of cases that where classified correctly. All used the last tenth for testing.

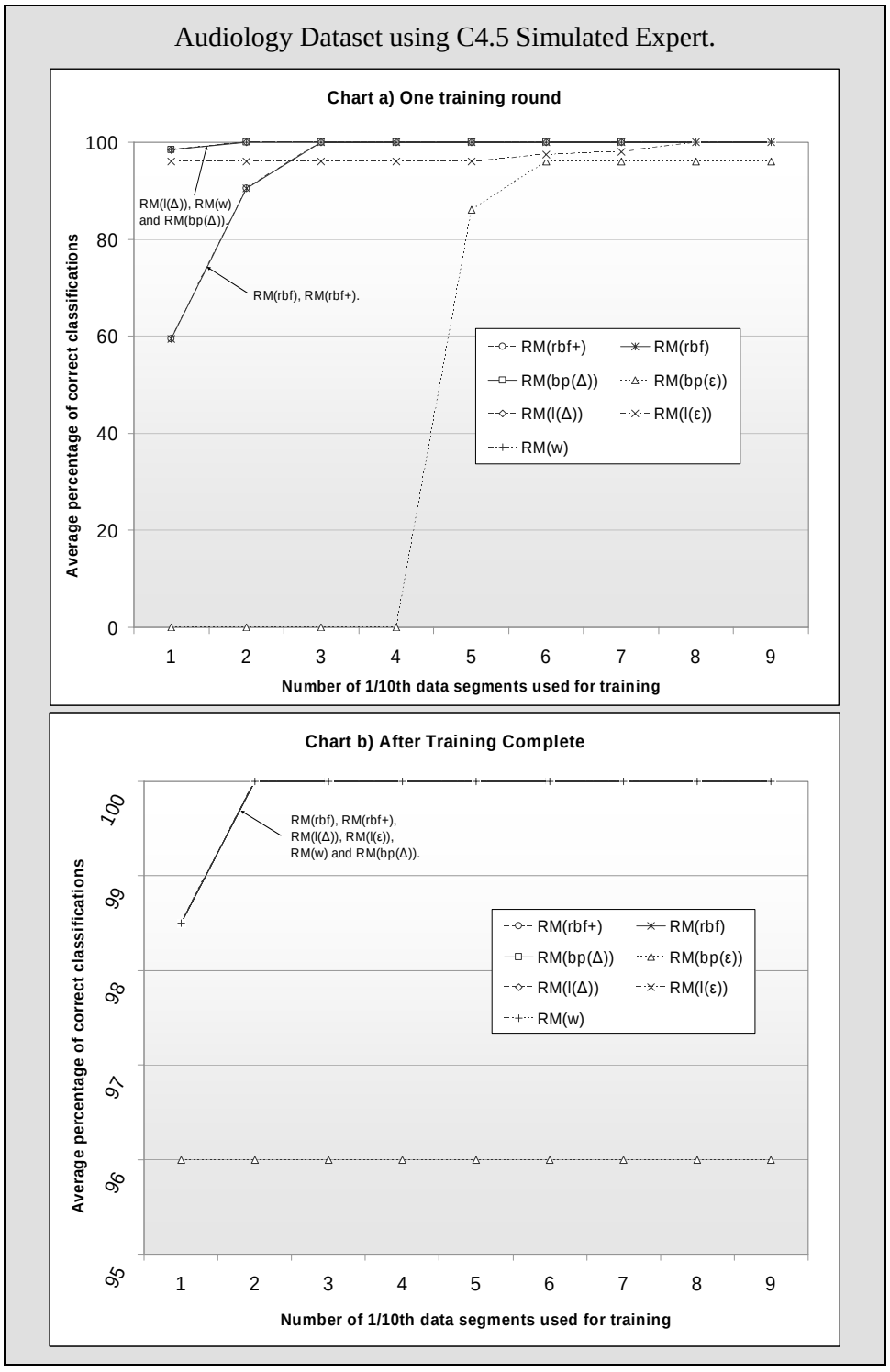


Figure 6-9 Two charts comparing how each of the seven proposed methods RM_w, $RM_{l(\varepsilon)}$, $RM_{l(\Delta)}$, $RM_{bp(\varepsilon)}$, $RM_{bp(\Delta)}$, RM_{rbf} and RM_{rbf+}, perform on the Nursery dataset using the C4.5 simulated expert. Chart a) shows how the methods compare after only one viewing of the training set. Chart b) shows how the methods compare after training has completed. The X-axis shows how many tenths of the dataset were used for training. The Y-axis shows the percentage of cases that where classified correctly. All used the last tenth for testing.

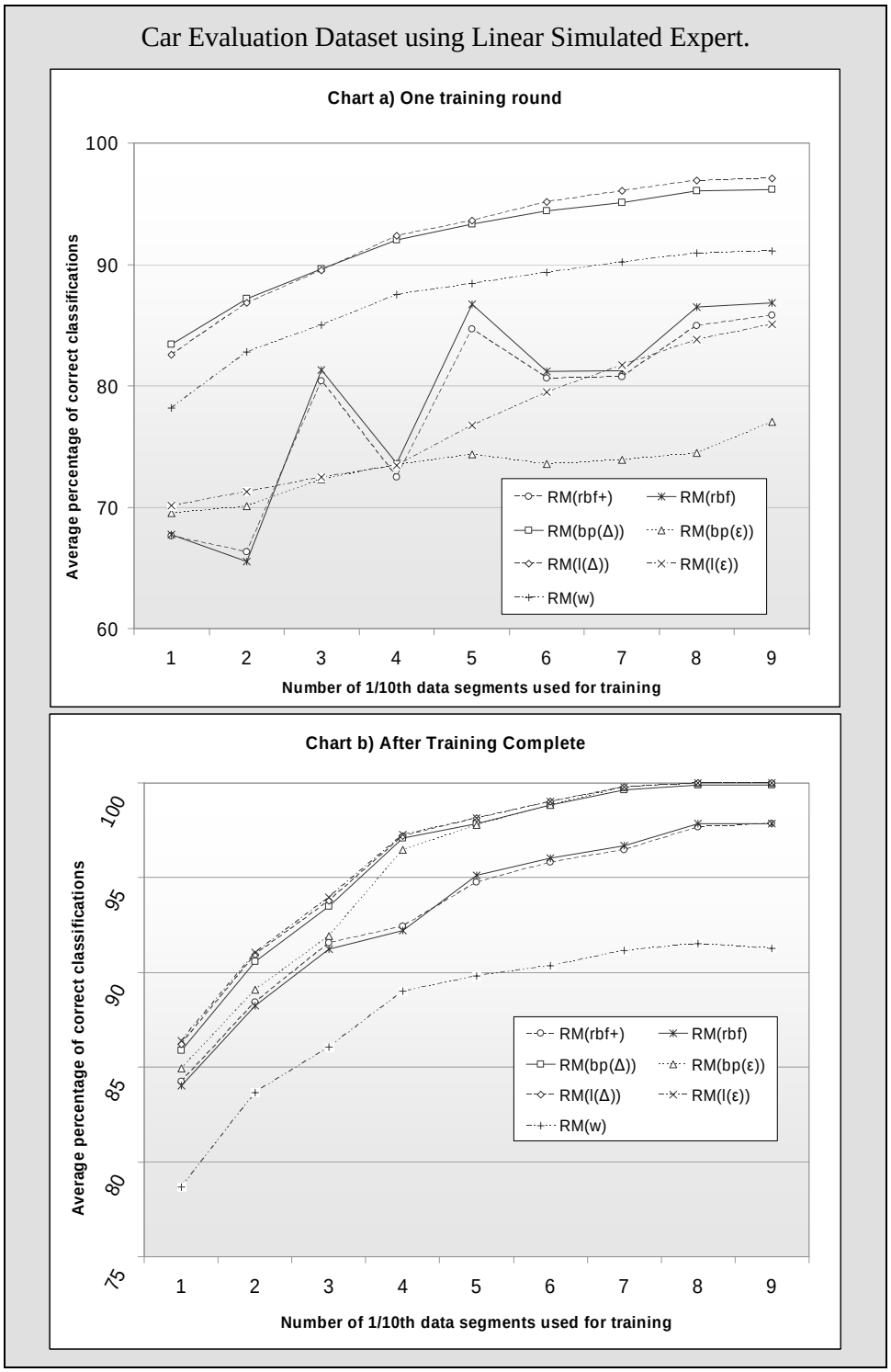


Figure 6-10 Two charts comparing how each of the seven proposed methods RM_w, $RM_{l(\varepsilon)}$, $RM_{l(\Delta)}$, $RM_{bp(\varepsilon)}$, $RM_{bp(\Delta)}$, RM_{rbf} and RM_{rbf+}, perform on the Car Evaluation dataset using the C4.5 simulated expert. Chart a) shows how the methods compare after only one viewing of the training set. Chart b) shows how the methods compare after training has completed. The X-axis shows how many tenths of the dataset were used for training. The Y-axis shows the percentage of cases that where classified correctly. All used the last tenth for testing.

These charts show an overview of all the results after the first iteration and after training has completed. Methods were regarded as being complete after they showed little sign of further improvement. There is a lot of information in these graphs that will be discussed in this section. Some of the information will be shown again in a more digestible manner as each point is discussed. The areas of discussion below relate to:

- The performance of the basic relationship weighting of RM_w.
- How well the single-step-Δ-initialisation-rules performed.
- The difference between linear and non-linear methods.
- The greater effectiveness of the *RAN* style *RBF+* network compared to the basic *RBF*.
- Which network type, backpropagation or RBF, was best for RM.

6.2.1.2 RM_w Performance

As previously mentioned, RM_w calculated a simple weight for each relationship in the MCRDR tree. This weight was calculated at the time of each new rule being added to the system. As expected, the above results show this was not very effective. However, what was surprising was that without any learning it was able to give correct classification occasionally. Even more astonishing was its performance in the Chess, TTT and especially the Audiology datasets. For instance, in the TTT experiments it averaged over 97% accuracy, and was the equal best performer in the Audiology dataset. These results essentially show that the two binary and the Audiology datasets are relatively simple, with a high degree of linear separation between the possible conclusions. These results show that the idea of calculating an immediate value does give a meaningful value initially, which could be improved upon during a learning phase.

It may also be noticed that generally the method did not improve its performance significantly when viewing more segments of the dataset before testing on the last tenth. This is because during the first cases seen the method has accrued a rating for the collection of relationships and the subsequent viewing of cases offers little possibility of refinement without an associated learning rule. When reviewing these results it was evident that the value calculated was most relevant during the early stages of knowledge acquisition.

6.2.1.3 The Single-Step-Δ-Initialisation-Rules

Probably the most important algorithm decision made was the inclusion of the *single-step-Δ-initialisation-rule*. When included it was hoped that it would provide significantly faster learning capabilities while not adversely affecting the technique's ability to learn and generalise over the long term. In order to justify the inclusion of this unique initialisation rule, a comparison between $RM_{l(\Delta)}$ and $RM_{bp(\Delta)}$, and their randomised initialisation partners, $RM_{l(\varepsilon)}$ and $RM_{bp(\varepsilon)}$ respectively, must be made. Due to large amounts of information presented in the previous figures, a number of additional charts are shown in Tables 6-5 and 6-6, comparing each method pair on each dataset.

A quick glance down the first column of results clearly shows that the methods using the *Δ-rules* always out performed the randomly initialised method. A *T-Test* was performed comparing all results, finding that the $RM_{l(\Delta)}$ method performed significantly better than $RM_{l(\varepsilon)}$, with the worst *p-value* being *0.04*. While the comparison between the $RM_{bp(\Delta)}$ and $RM_{bp(\varepsilon)}$ showed an even stronger statistically significant improvement, with the worst *p-value* of only *0.001*. This difference was due to the $RM_{bp(\varepsilon)}$ method learning very slowly initially. It can also be seen that the random method had much greater error ranges on some datasets, such as the chess set. This was because sometimes it received a lucky number during the random initialisation, which aided learning.

One interesting result was $RM_{bp(\varepsilon)}$*'s* performance on the TTT dataset. Here it can be seen that the method was unable to noticeably improve its performance when it saw more of the dataset, and was only able to improve with protracted training. This was because each time a new rule is added it receives a new input and hidden nodes with random weights. These random weights prevent the system from learning until a series of cases are seen that do not create new nodes. Basically, the new weights are so small that when combined with the affect of the second layer sigmoid, they are unable to alter the resulting conclusion. Therefore, they require a significant amount of training before the weights change enough to have an affect. The effect of this is more evident in the TTT dataset because the expert creates a very high number of rules with only a small number of cases exacerbating the problem more.

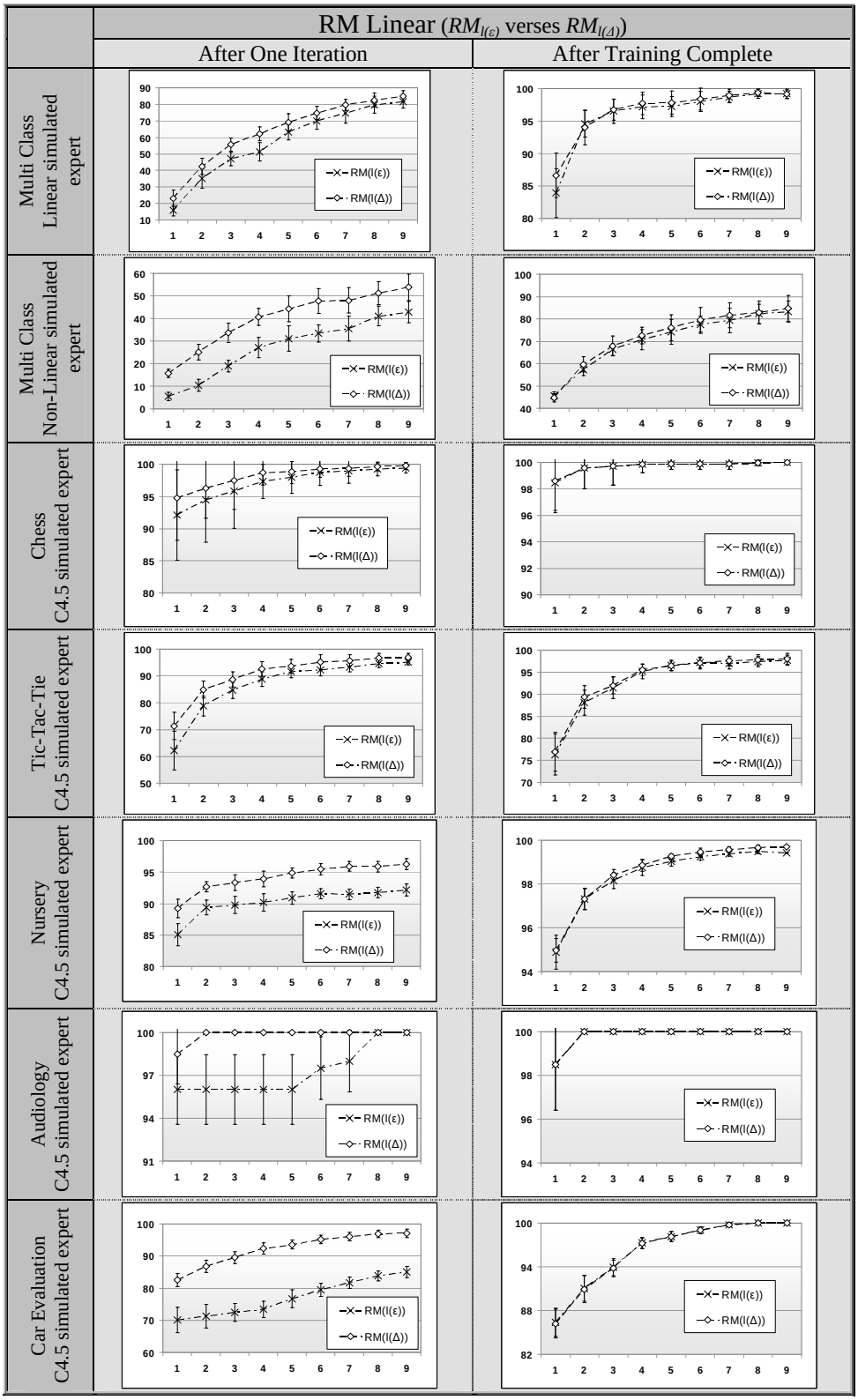


Table 6-5: Compares the linear versions of RM ($RM_{l(\varepsilon)}$ and $RM_{l(\Delta)}$) for each of the datasets and types of experts tested. Each chart shows each technique's average percentage of correct classifications across each of the nine tests performed. The charts also show error bars set at the 95% confidence mark.

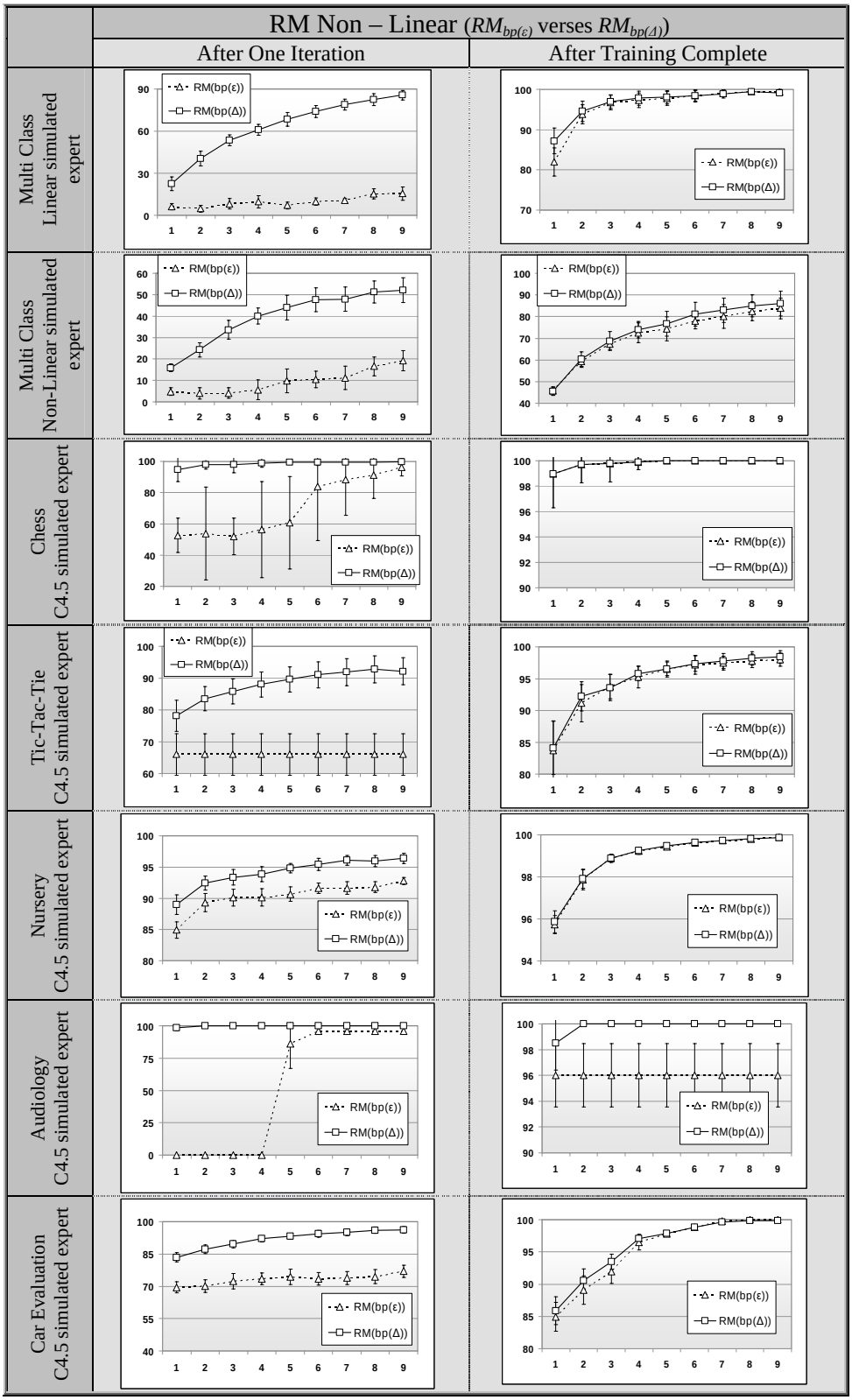


Table 6-6: Compares the non-linear versions of RM ($RM_{bp(\varepsilon)}$ and $RM_{bp(\Delta)}$) for each of the datasets and types of experts tested. Each chart shows each technique's average percentage of correct classifications across each of the nine tests performed. The charts also show error bars set at the 95% confidence mark.

The improved performance because of the *single-step-Δ-initialisation-rules* was largely expected, although the degree of improvement in the non-linear methods was impressive. The primary concern, however, at the inclusion of the *Δ-rules* in the methodology was whether or not the targeted initialisations would damage the method's ability to learn and generalise over time. The second column presents the results after training is complete. Here the *Δ-rule* based methods were not expected to do better than the random initialised methods and the fear was that they would be significantly worse. However, their performance was if anything marginally better in all tests for all datasets. This was not caused through the *Δ-rule* based methods getting a head start, because training was continued until the randomised methods showed no discernable learning.

Instead, the slightly improved performance of the *Δ-rule* based methods is believed to be because of the momentum effect. Previously, as discussed in section 4.3.4, ANN researchers have found that adding a momentum factor can force an ANN to jump over local minima. It is possible that the use of the *Δ-rule's* ability to take a large step towards the value a node requires, allows it to step over a number of possible local minima. These minima missed still have the potential to trap the randomly initialised methods. This result, while unexpected, means the *Δ-rule* can be used in RM to vastly improve learning initially, through directly representing new knowledge in the ANN without any adverse affect on the trainability of the method and may even offer a slight overall advantage.

6.2.1.4 Linear verses Non-Linear

In ANN based methods developers face a significant decision on whether a linear or non-linear network should be used. Generally, a linear ANN will learn any task significantly faster, because error information does not need to be fed back through two sigmoidal thresholds. However, the linear ANN will never solve problems with a non-linear component. In RM it was unclear during methodological development whether it could be assumed that a purely linear approach would suffice, or if a non-linear approach was used, how much that would affect the trainability of the system. Therefore, methods were developed for both. Table 6-7 shows a set of charts comparing $RM_{l(\Delta)}$ and $RM_{bp(\Delta)}$ *directly.*

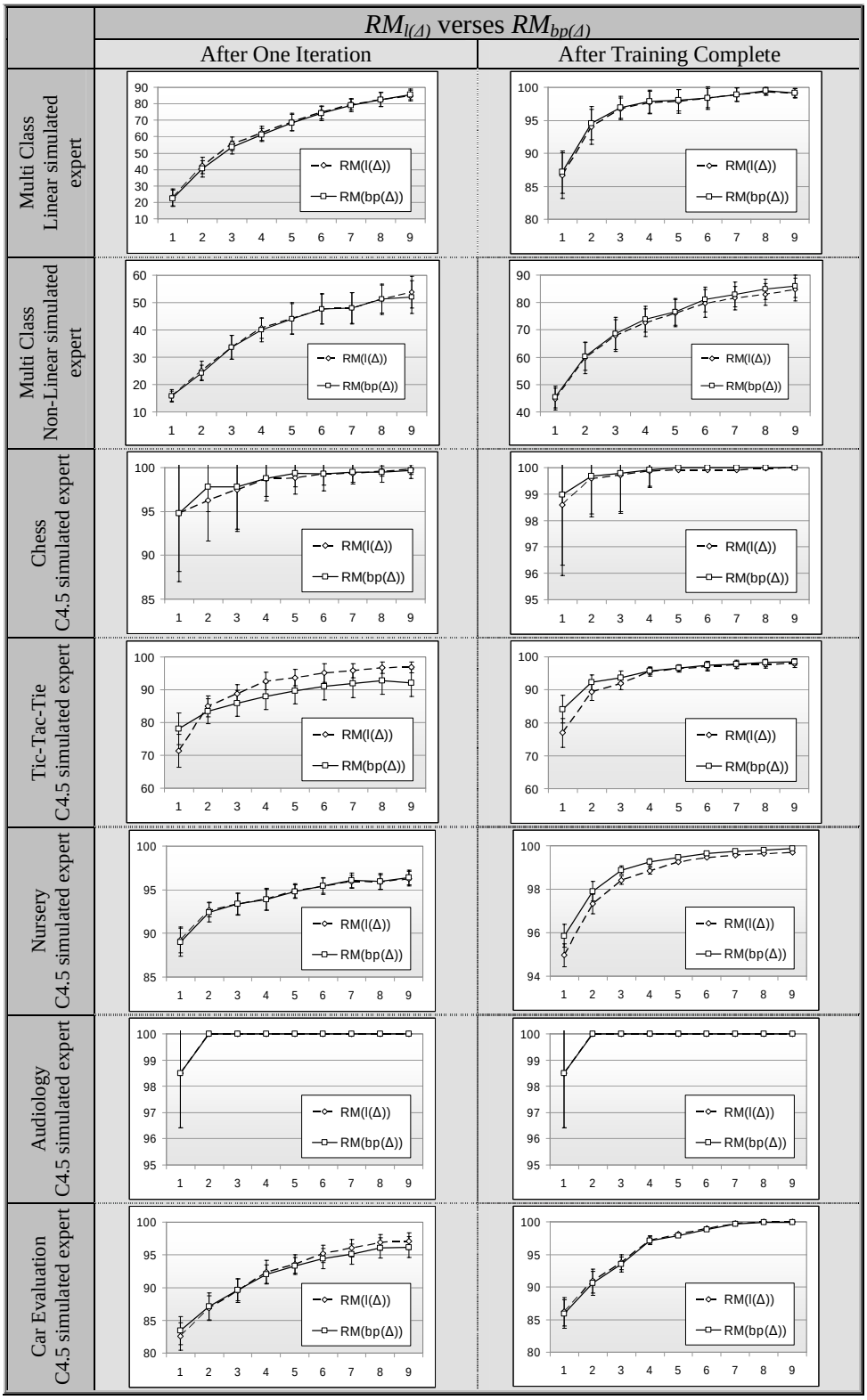


Table 6-7: Compares $RM_{I(\Delta)}$ and $RM_{bp(\Delta)}$ for each of the datasets and types of experts tested. Each chart shows each technique's average percentage of correct classifications across each of the nine tests performed. The charts also show error bars set at the 95% confidence mark.

Looking at the first column of results from Table 6-7, it can be seen that, with the exception of the TTT dataset, there is no statistically relevant advantage to the linear method. The significant advantage on the TTT dataset (average p-value of just under 0.05 over the last 6 tests (4-9 tenths for training)) is because it had quickly learned adjustments to the initial step made by the *Δ-rule*. This training is visible in the TTT experiment because of the simplicity of the dataset. The poor performance of the linear method on the one-tenth tests is because it had not seen a sufficient number of cases for even the linear technique to have started learning. The non-linear version having done better in the one tenth tests is most likely a statistical anomaly caused by the small set of results.

The advantage in favour of the linear method is extremely small, especially if compared to the usual difference between a linear and non-linear ANN. In the chess dataset it can be seen that the non-linear method actually performed better. However, this is not statistically significant and the large error bars suggests that it most likely is not correct.

While non-linear methods do not learn as quickly, they usually do learn a better solution in the long run. The second column of results compares the $RM_{l(\Delta)}$ and $RM_{bp(\Delta)}$ methods to see if that holds in this methodology. While in the Nursery dataset the non-linear approach did marginally better, there is very little statistical evidence in these results indicating the non-linear approach performed better after training. Most of the results indicate that, if there is any advantage at all, it is only extremely small. It can also be seen that even in the multi-class dataset using the non-linear simulated expert, which was specifically designed to have a major nonlinearity in the classifications was only marginally better.

The performance of $RM_{bp(\Delta)}$ in the TTT and Nursery dataset when only training on a small percentage of the dataset are the only results showing any significance. This result suggests that the linear method could not build a sufficiently general function from the sparse amount of examples, whereas, the non-linear approach could use its non-linear nature to capture extra information. Some interesting questions arise from these results. First, why would the non-linear approach generally offer only a small advantage? Second, why would it offer significantly more advantage in information sparse environments?

While the answer to these questions may need further study, the answer appears to be due to MCRDR providing a means for reducing dimensionality of

the problem domain. This would be similar, in principle, to the way Support Vector Machines (SVM) or the hidden layers in a neural network behave. Basically, it appears that the expert when creating rules has created them in such a way that the problem for the neural network is actually only linear, even though the original problem was non-linear. Therefore, it suggests that the relationships between rules in the MCRDR tree are primarily linear, regardless of the complexity of the dataset.

This explanation can be extended to potentially answer the second question as well. In situations where the MCRDR tree is incomplete, due to an information sparse domain or just because it is still incomplete, then it is not able to fully reduce the dimensionality. Therefore, as the MCRDR tree grows it is reducing the dimensionality of the domain. This could be further extended to suggest that a knowledge base is complete when the hidden context is linearly separable.

These ideas are interesting, but not the focus of this thesis and so are not further investigated. The reason for performing this analysis was to find the best method of hybridising MCRDR and an ANN. The trouble is that these results do not fully justify selecting either approach. However, the potential advantage of the non-linear approach in some situations, and with little significant learning penalty, meant that it was potentially the most versatile approach to be chosen for further experimentation.

6.2.1.5 Radial Basis Function Comparison

Often in ANN research RBF networks have been found to be very effective. Their primary drawback can be poor generalisation when over fitting of hidden nodes occurs. In chapter 5, two methods were developed using RBF style networks. The first method, RM_{rbf} added hidden nodes whenever a new input node was added. The second method, RM_{rbf+} also did this but also investigated a number of other possible improvements, such as adding hidden nodes even when rules are not added, dynamically changing the size of the hyperellipsoid functions and the use of input keys. Table 6-8 below, compares these two RBF style methods on each of the datasets both before and after training.

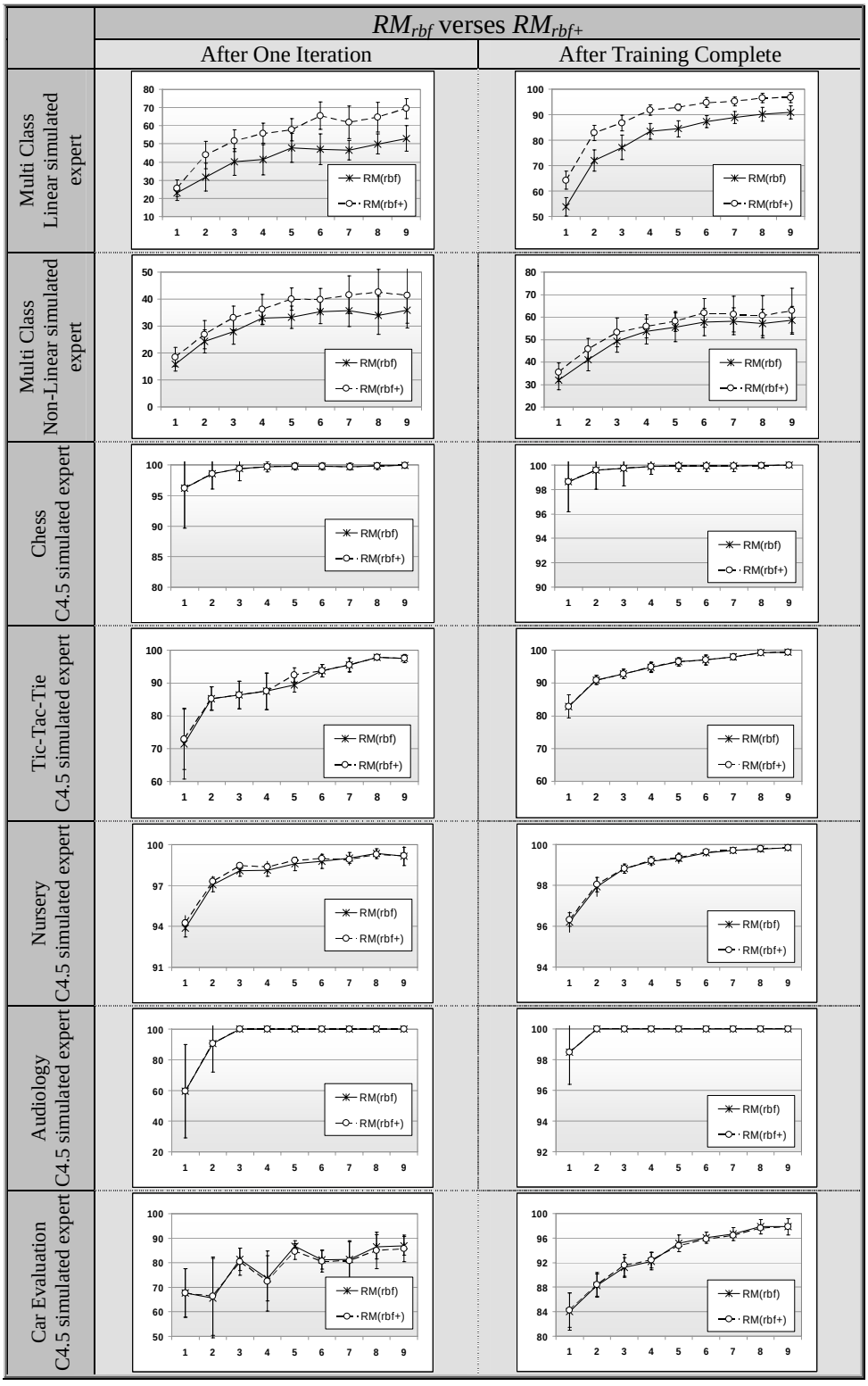


Table 6-8: Compares RM_{rbf} and RM_{rbf+} for each of the datasets and types of experts tested. Each chart shows each technique's average percentage of correct classifications across each of the nine tests performed. The charts also show error bars set at the 95% confidence mark.

In the more complex multi-class datasets it can be seen that the RM_{rbf+} approach has provided some advantage during both the first iteration and after training has completed. However, only the results with the linear simulated expert have any statistical significance (p-value of 0.05 or less for all except the $1/10^{th}$ test after the first iteration). Interestingly, the more complex method performed virtually identically on the chess, TTT, Nursery and Audiology datasets and slightly worse on the Car Evaluation dataset. This was because the amount of hidden nodes added when new rules were added, was sufficient to learn the task. Therefore, the requirements for adding additional hyperellipsoid functions occurred very infrequently. Tightening the threshold for node creation only resulted in a worse performance due to over fitting, such as with the Car evaluation dataset.

6.2.1.6 Comparison of Non-linear Approaches

The final comparison needs to be made between the best of the backpropagation and RBF approaches. These two methods represent the main methods developed. The RBF based approaches were developed as they generally perform better in most ANN applications. However, this ANN application is vastly different to the general network usage and it was unclear how the changing input space for the network would affect performance. Table 6-9 below, compares the $RM_{bp(\Delta)}$ and RM_{rbf+} non-linear methods on each of the datasets both before and after training.

This set of charts presents an interesting dichotomy of results. On the one hand, the backpropagation method has clearly outperformed the RBF approach on the Multi-class, Audiology, and Car Evaluation dataset. This was even more pronounced on the Multi-class dataset after training completed, with a strong statistical significance across all tests. Yet the RBF approach was able to perform marginally better during the early stages of training on the Chess and TTT datasets and significantly better on the Nursery data.

The RBF's poor performance on the multi-class dataset can be put down to the large amount of contradictory rules created. Generally, it was observed that there were many *'stop rules'*, as well as rules that significantly changed the conclusion in the multi-class datasets. This caused many hidden nodes to be added into the RBF that were later not used. This caused significant over

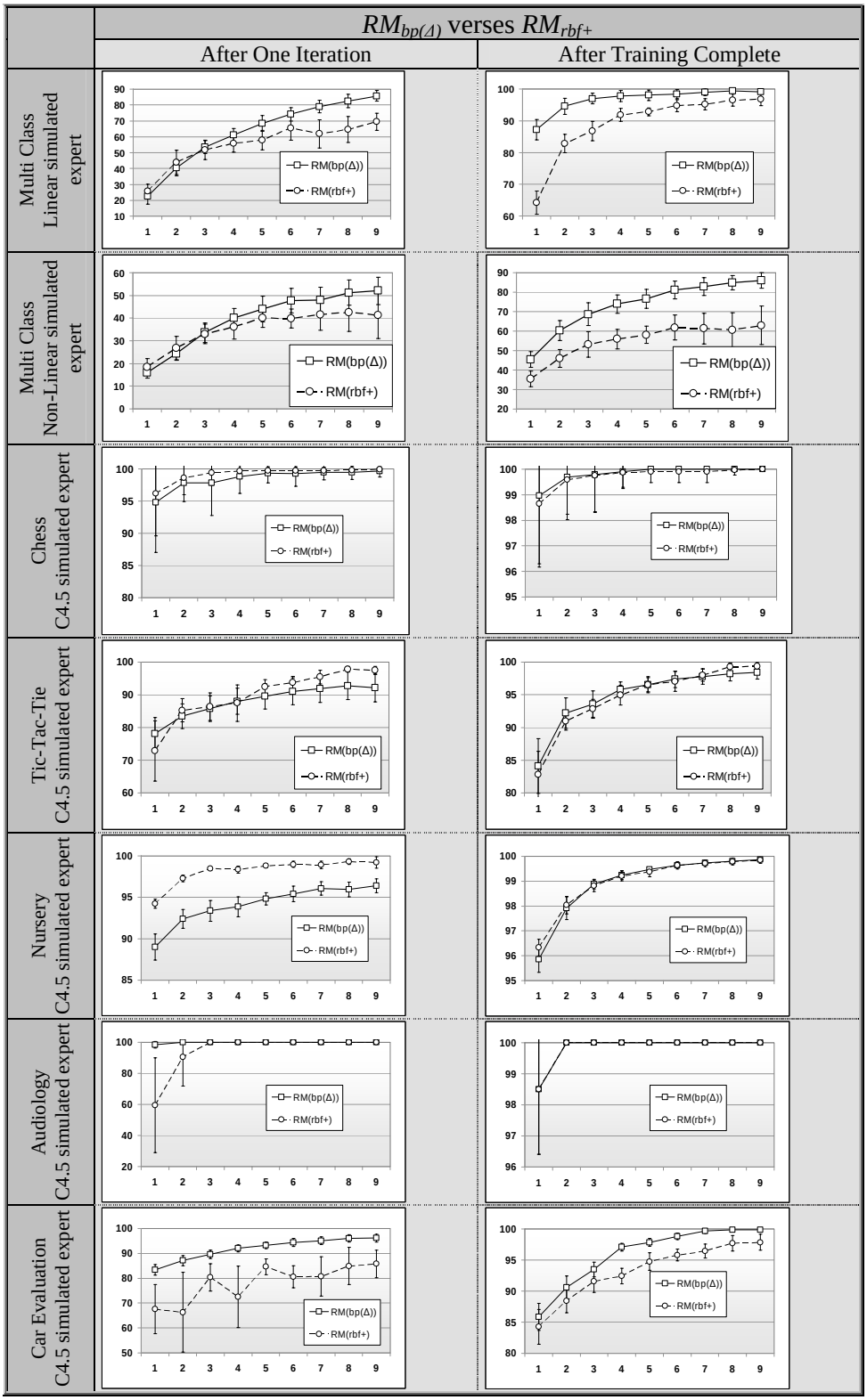


Table 6-9: Compares $RM_{bp(\Delta)}$ and RM_{rbf+} for each of the datasets and types of experts tested. Each chart shows each technique's average percentage of correct classifications across each of the nine tests performed. The charts also show error bars set at the 95% confidence mark.

fitting where nodes were not needed, plus poor coverage where nodes were required. Likewise the complexity in the Audiology and Car Evaluation datasets also resulted in over fitting of the RBF based approach. In the Chess, TTT and especially the Nursery datasets the rules where fewer and more straight forward with a smaller range of conclusions, allowing less hidden nodes to be created and, therefore, causing less over fitting.

The most significant problem with the RBF approach, however, which is not clear in these results, is its volatility (although the large error bars in some results are an indication). It was found during experimentation that the smallest change to a single parameter could have a significant detrimental effect. Additionally, the occasional experiment would produce a result vastly different to the others. Thus, the method was very reliant on the order the cases were presented. This volatility combined the above results show that in general the backpropagation method, $RM_{bp(\Delta)}$, would be the preferred choice in most situations. However, there are occasions where the RBF style system could be used effectively.

6.2.1.7 Online Classification

One of the main features RM was hoping to gain from the use of the ANNs was the ability to generalise well, an ability not usually attributed to KBSs. It was also hoped that the KBS portion of the hybrid system would contribute the ability to learn quickly from only seeing cases a few times, preferably only once, which is not a typical feature of ANNs. The collection of results in this section is aimed at investigating the method's ability to learn quickly in an online environment. If successful, then it is RM's ability to capture and combine the two underlying system's abilities that potentially makes it a powerful methodology.

This section provides Figures 6-11, 6-12, 6-13, 6-14, 6-15 and 6-16 showing how each of the seven methods developed and performed at classification in an online learning environment for each dataset. To reduce the number of results the multi-class dataset using the linear simulated expert was not used. Each point on the charts is an average of the previous 10 data segments (over 2 data segments on the small Audiology dataset), over the ten randomised runs. Each

segment contains a random selection of cases, 1/50th the size of the whole dataset. Error bars were omitted, allowing greater readability.

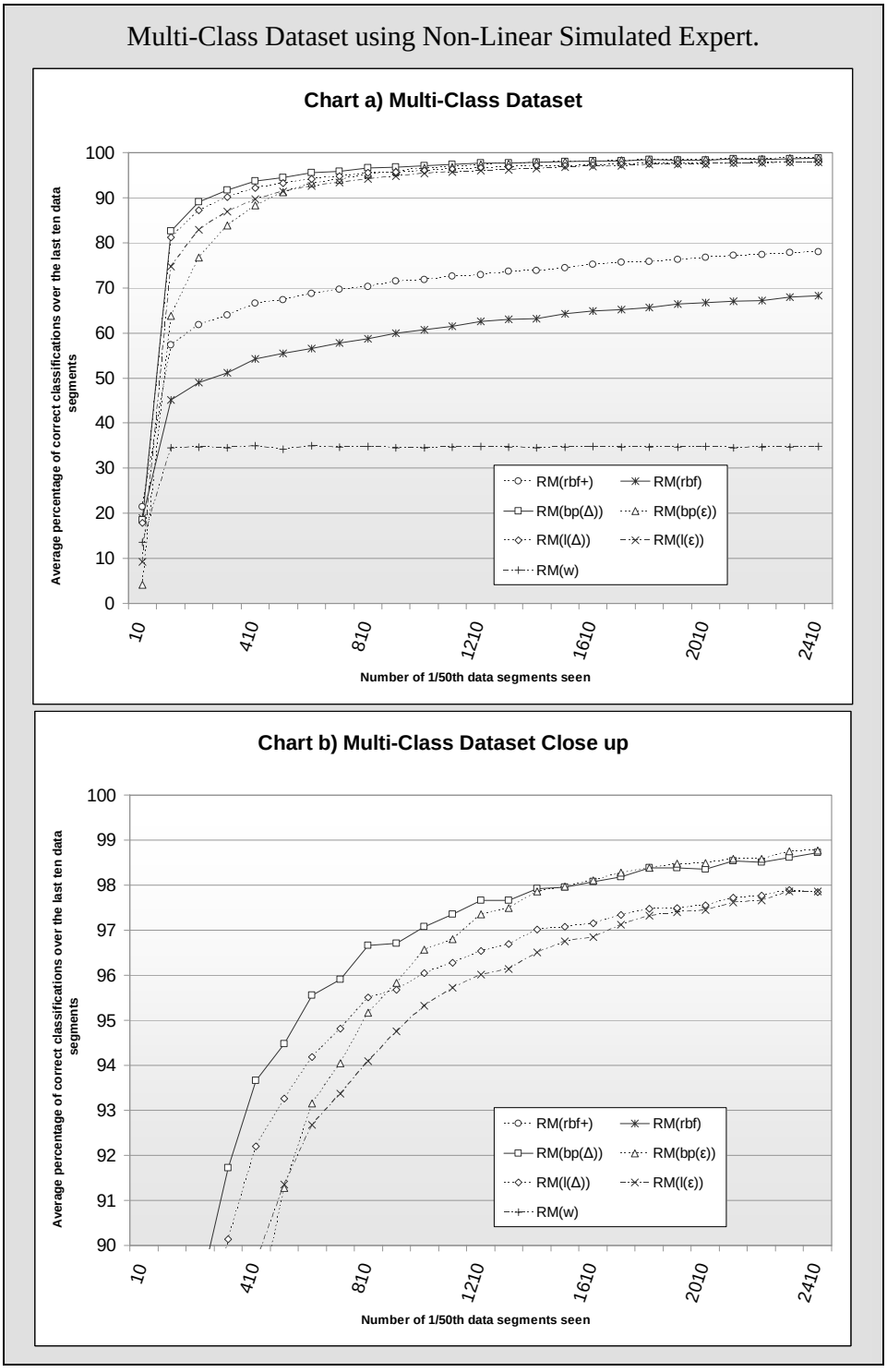


Figure 6-11: Two charts comparing how each of the seven proposed methods RM_w, $RM_{l(\varepsilon)}$, $RM_{l(\Delta)}$, $RM_{bp(\varepsilon)}$, $RM_{bp(\Delta)}$, RM_{rbf} and RM_{rbf+}, perform on the Multi-Class dataset using the non-linear simulated expert. The x-axis shows the amount of 1/50 data segments that have been seen. The y-axis shows the percentage correct over the last 10 data segments. Each point is an average across ten runs.

Chart a) shows the overall result. Chart b) shows a close up of the four methods that all performed well.

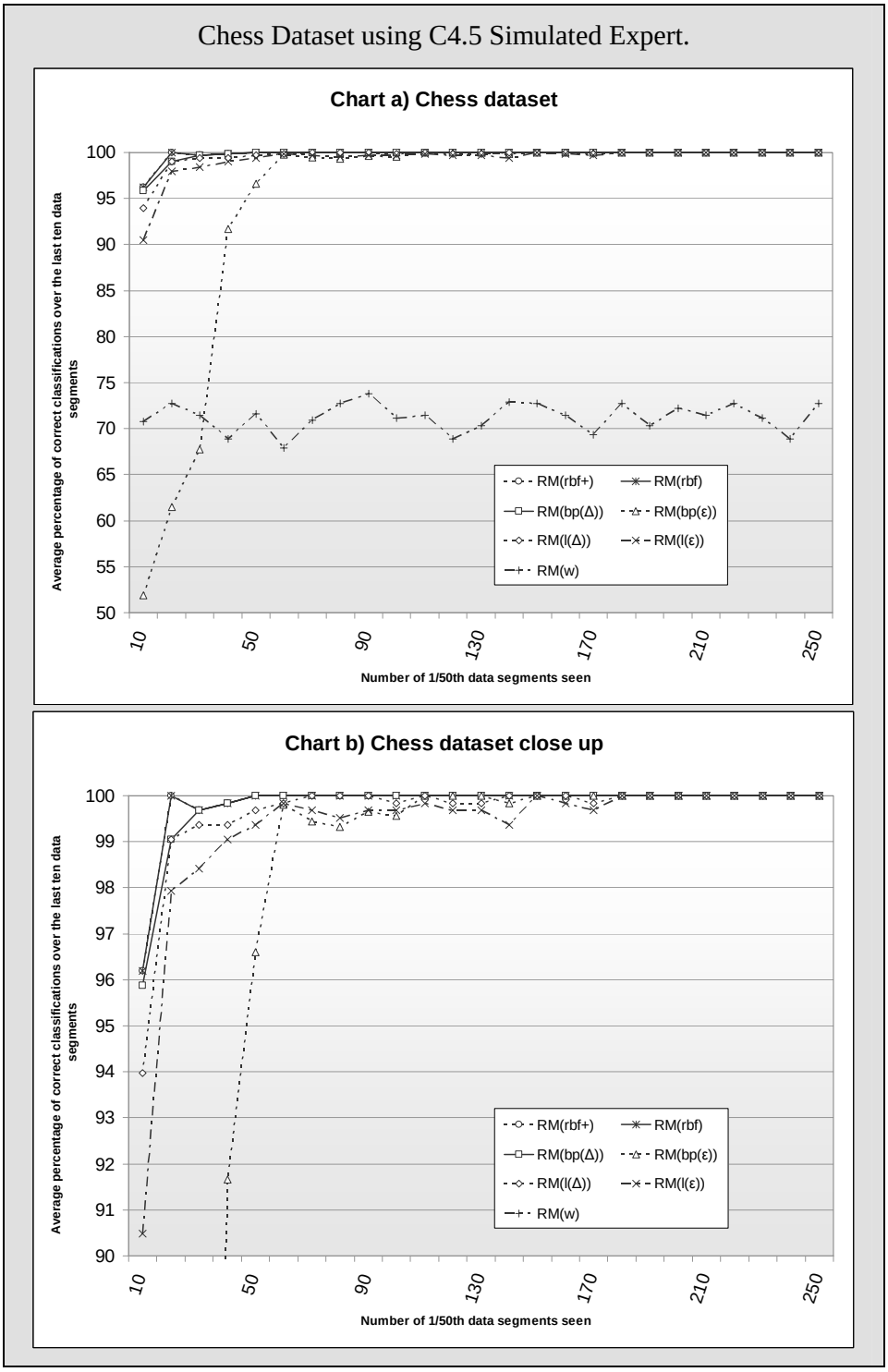


Figure 6-12: Two charts comparing how each of the seven proposed methods RM_w, $RM_{l(\varepsilon)}$, $RM_{l(\Delta)}$, $RM_{bp(\varepsilon)}$, $RM_{bp(\Delta)}$, RM_{rbf} and RM_{rbf+}, perform on the Chess dataset using the C4.5 simulated expert. The x-axis shows the amount of 1/50 data segments that have been seen. The y-axis shows the percentage correct over the last 10 data segments. Each point is an average across ten runs. Chart a) shows the

overall result. Chart b) shows a close up of the six methods that all performed well.

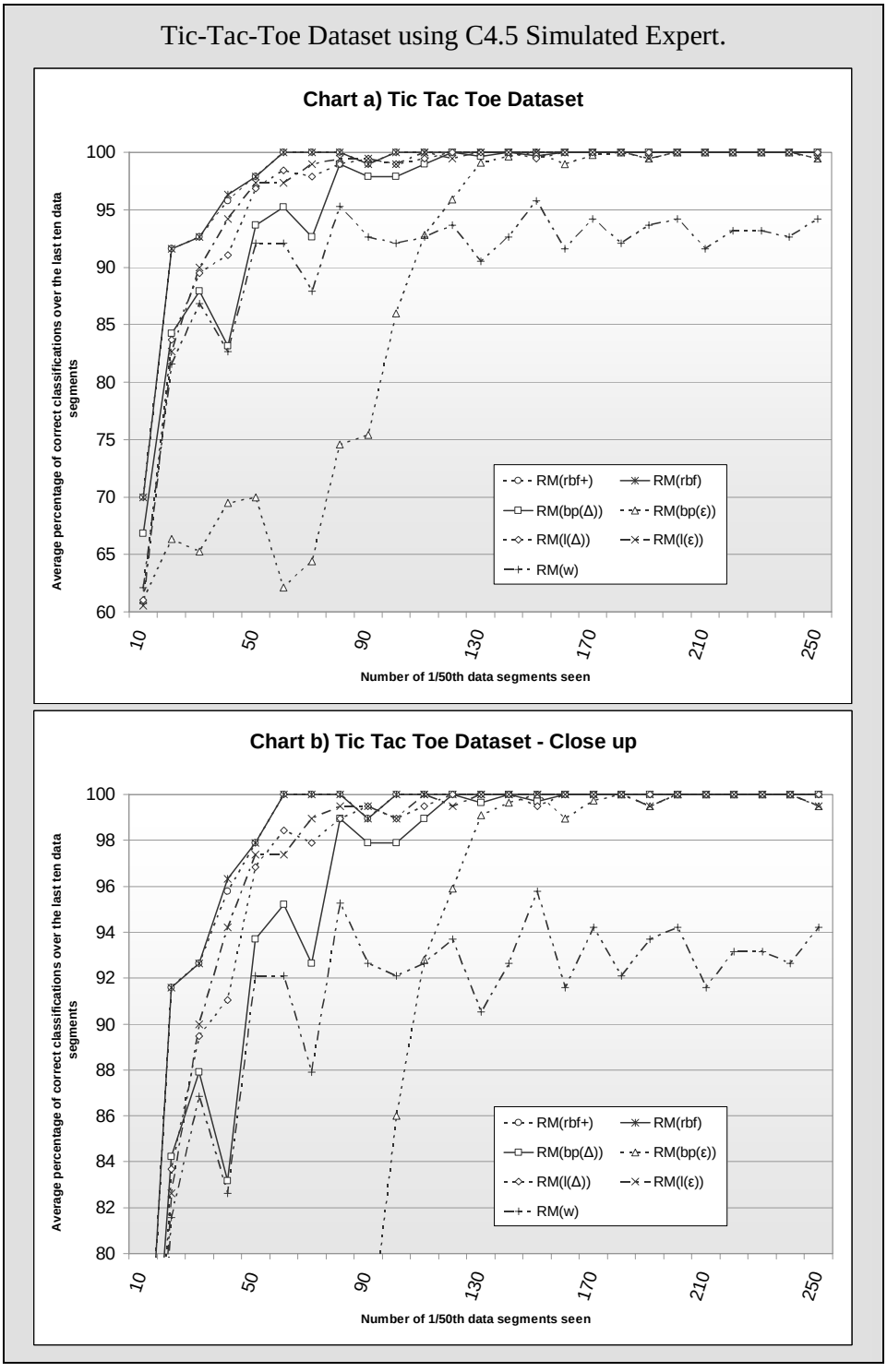


Figure 6-13: Two charts comparing how each of the seven proposed methods RM_w, $RM_{l(\varepsilon)}$, $RM_{l(\Delta)}$, $RM_{bp(\varepsilon)}$, $RM_{bp(\Delta)}$, RM_{rbf} and RM_{rbf+}, perform on the TTT dataset using the C4.5 simulated expert. The x-axis shows the amount of 1/50 data segments that have been seen. The y-axis shows the percentage correct over the last 10 data segments. Each point is an average across ten runs. Chart a) shows the overall result. Chart b) shows a close up of the seven methods.

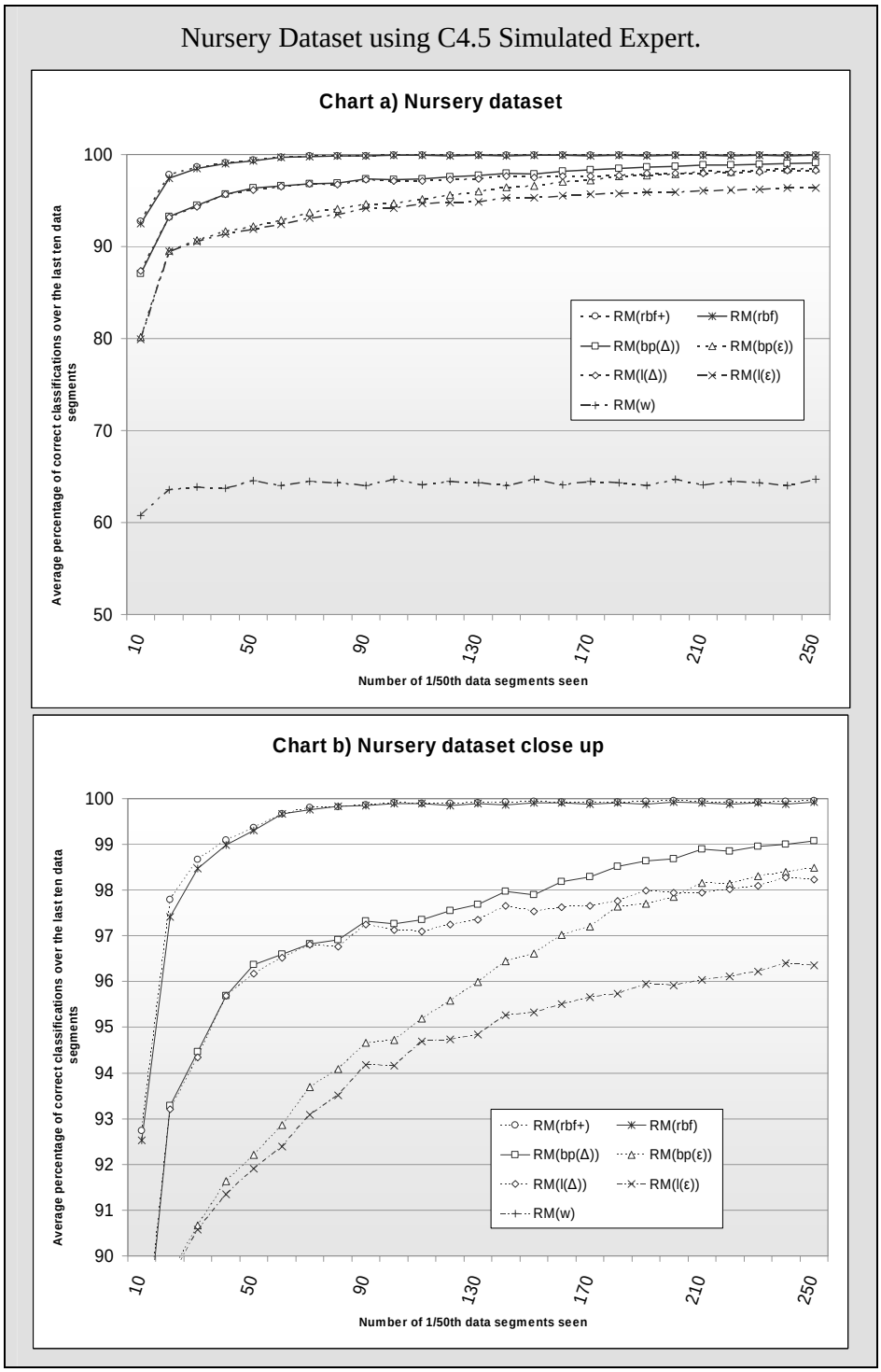


Figure 6-14: Two charts comparing how each of the seven proposed methods RM_w, $RM_{l(\varepsilon)}$, $RM_{l(\Delta)}$, $RM_{bp(\varepsilon)}$, $RM_{bp(\Delta)}$, RM_{rbf} and RM_{rbf+}, perform on the Nursery dataset using the C4.5 simulated expert. The x-axis shows the amount of 1/50 data segments that have been seen. The y-axis shows the percentage correct over the last 10 data segments. Each point is an average across ten runs. Chart a) shows the overall result. Chart b) shows a close up of the seven methods.

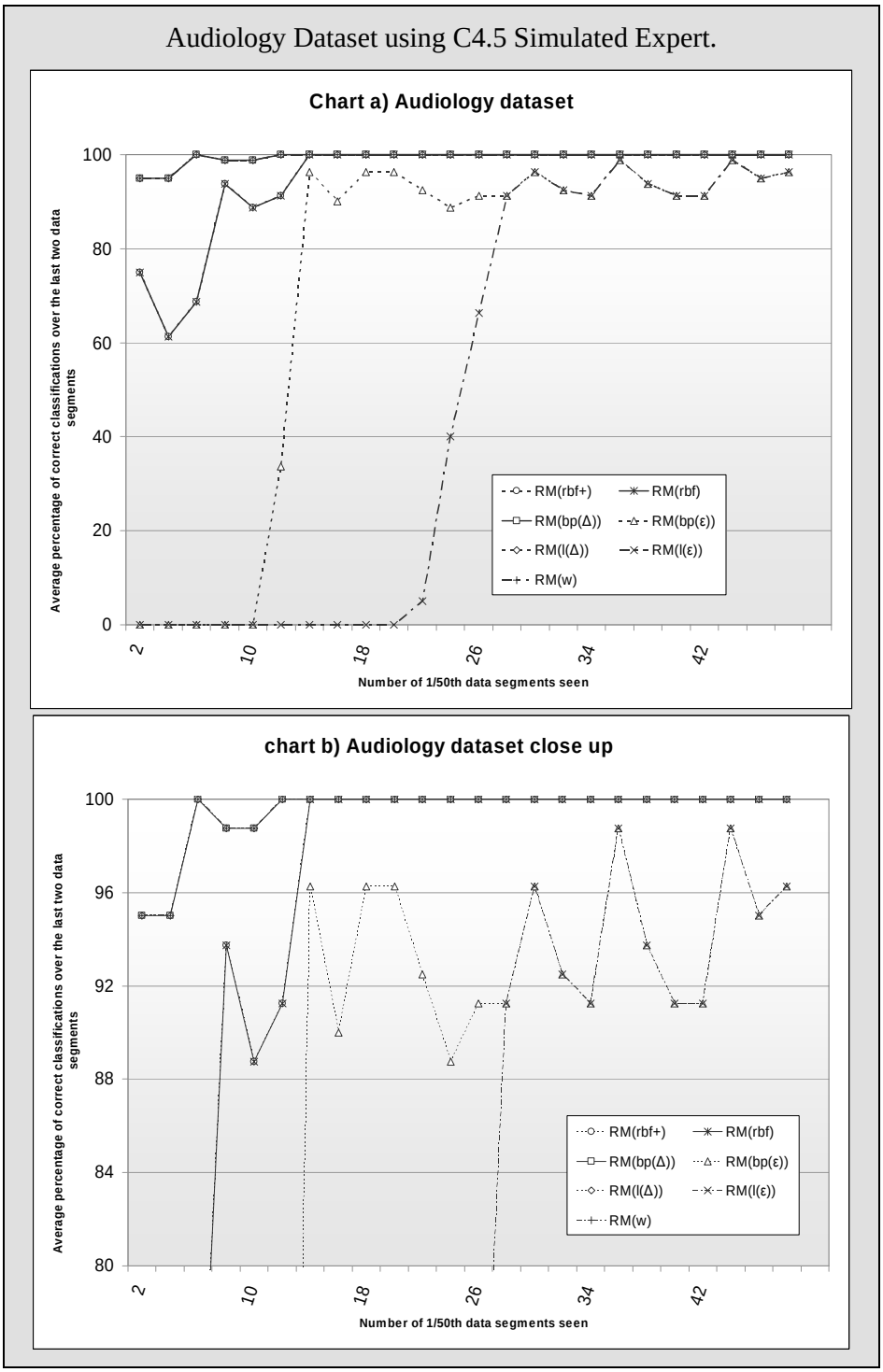


Figure 6-15: Two charts comparing how each of the seven proposed methods RM_w, $RM_{l(\varepsilon)}$, $RM_{l(\Delta)}$, $RM_{bp(\varepsilon)}$, $RM_{bp(\Delta)}$, RM_{rbf} and RM_{rbf+}, perform on the Audiology dataset using the C4.5 simulated expert. The x-axis shows the amount of 1/50 data segments that have been seen. The y-axis shows the percentage correct over the last 2 data segments. Each point is an average across ten runs. Chart a) shows the overall result. Chart b) shows a close up of the seven methods.

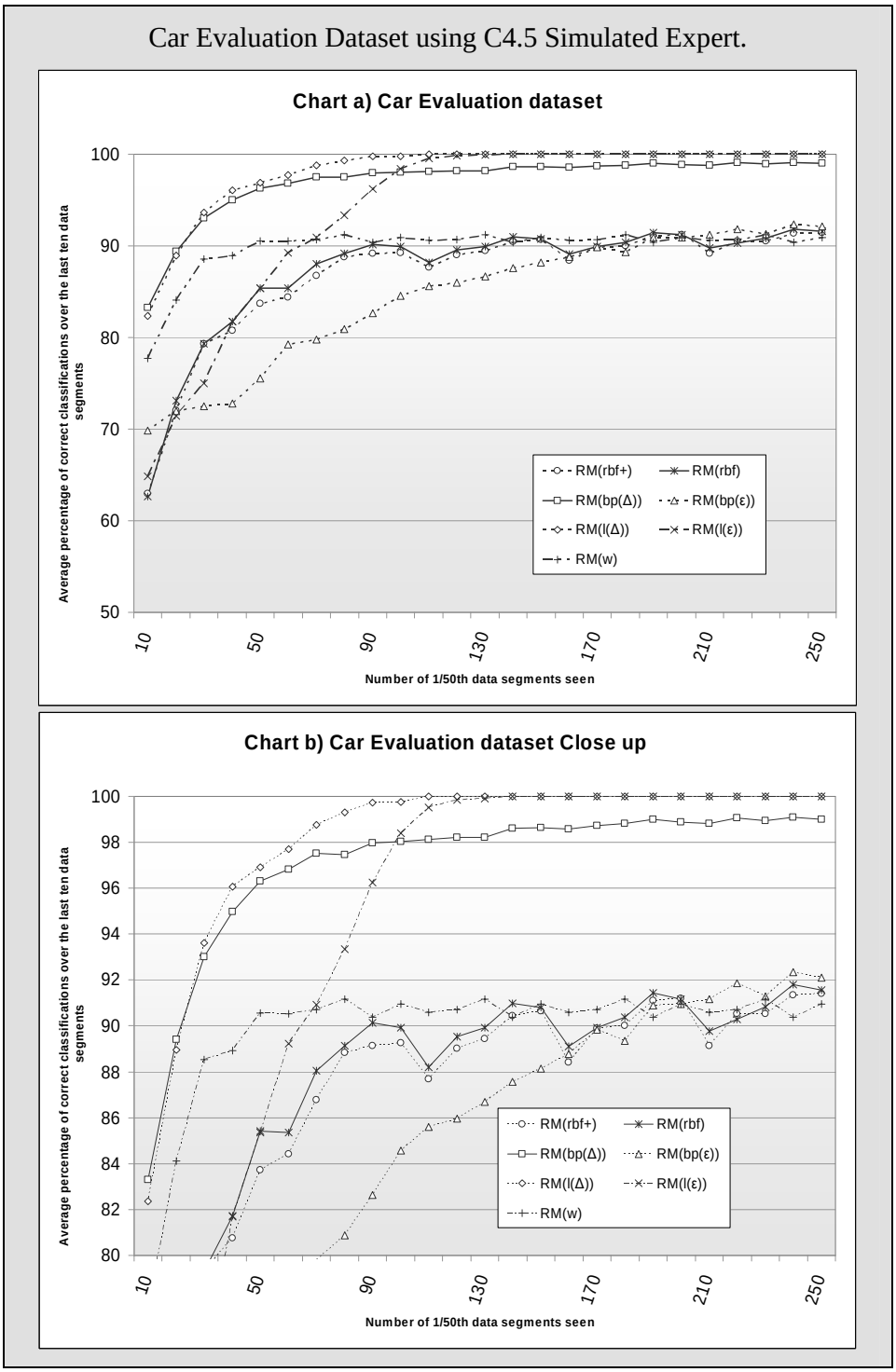


Figure 6-16: Two charts comparing how each of the seven proposed methods RM_w, $RM_{l(\varepsilon)}$, $RM_{l(\Delta)}$, $RM_{bp(\varepsilon)}$, $RM_{bp(\Delta)}$, RM_{rbf} and RM_{rbf+}, perform on the Car Evaluation dataset using the C4.5 simulated expert. The x-axis shows the amount of 1/50 data segments that have been seen. The y-axis shows the percentage correct over the last 10 data segments. Each point is an average across ten runs. Chart a) shows the overall result. Chart b) shows a close up of the seven methods.

Firstly, looking at the results from the multi-class dataset it can be seen that the non-linear based backpropagation approach learned faster and sustained that advantage over the long term. The other backpropagation methods did eventually come close but did not learn as fast at the outset. It can be seen in the close up view that while the random initialisation approach, as expected caught up, the non-linear approach always maintained an advantage over the linear method. It can also be seen that both the methods using the *single-step-Δ-initialisation-rules* achieved a much faster initial learning boost at the outset. This is not a standard feature of ANNs. Unexpectedly though, even the randomly initialised method, $RM_{bp(\varepsilon)}$, was able to learn very quickly.

Likewise, in the Nursery dataset the non-linear based backpropagation approach outperformed the linear approach over a period of time. However, like in the generalisation approach the RBF based approaches did significantly better due to fewer contradictions and conclusions in the dataset. Contrasting the Nursery dataset, the Car evaluation dataset results show the linear version achieved an advantage. Although it appears that the non-linear approach was closing the gap slowly.

The results from the chess, TTT and audiology datasets are less clear. The problem with these was that all the main methods did extremely well making it more difficult to single one out as performing the best. It can be seen though that in the chess experiment the non-linear approach using backpropagation still did better than its linear counterpart and significantly better than the randomly initialised version. However, this was not duplicated in the TTT results where the linear method performed marginally better during the early stages.

What is clear from all these results is that the *Δ-rules* allowed much faster learning in the online environment. Generally, there is also an advantage in using the non-linear technique, because it usually performs better and rarely incurs any penalty in learning. Deciding the best approach between the RBF and backpropagation methods is, however, more difficult. It is expected that the selection should almost certainly be dependant on the problem domain in which RM is being applied. Therefore, in applications with simpler linear contextual relationships the RBF based approach offers advantages in both speed and long term learning. However, in the more complex and less linear domains the non-linear backpropagation method shows clear advantages.

6.2.2 Prediction

Traditionally, MCRDR and other KBSs can usually only be applied to classification problems. Even when used for prediction they usually still use the same basic classification style but each classification gives a predictive value instead. One advantage found in the hybrid system developed in this thesis is that it can also be applied in a prediction environment. This can be achieved by the network being setup to output just a single value, representing the system's prediction for the task at hand.

The expert, however, is still required to *'classify'* each case, as well as judge its accuracy in predicting the required value. This classification does not need to be a class as such, but can simply be a grouping of cases to similar cases in the expert's mind. Basically, s/he needs to determine if a case is different from those that previously reached the same concluding rule. If it is different then a new rule needs to be defined. Due to the restriction that our expert would be needed to both determine a form of classification, as well as a value, the C4.5 based experts could not be used and, therefore, neither could the chess and TTT datasets. Instead, only the multi-class prediction expert was used (6.1.1.4). This section is once again broken into two sections. The first deals with generalisation in a prediction environment and the second looks at how each method performs in predicting a value in an online environment.

6.2.2.1 Prediction Generalisation

The tests performed for prediction generalisation are essentially identical to those performed for the classification problem, except that instead of producing an output for each class, RM produces a single value. The value returned is then compared to the simulated expert's correct value. The absolute difference between these two values (*error*) is then averaged over all the cases in the data segment and logged. The results shown in Figure 6-17 show how each of the seven methods developed perform on the multi-class-prediction simulated expert. Each point is the average error for the test data segment averaged over ten randomised runs of the experiment, for each of the nine tests. To reduce the complexity of the charts shown, error bars have been omitted.

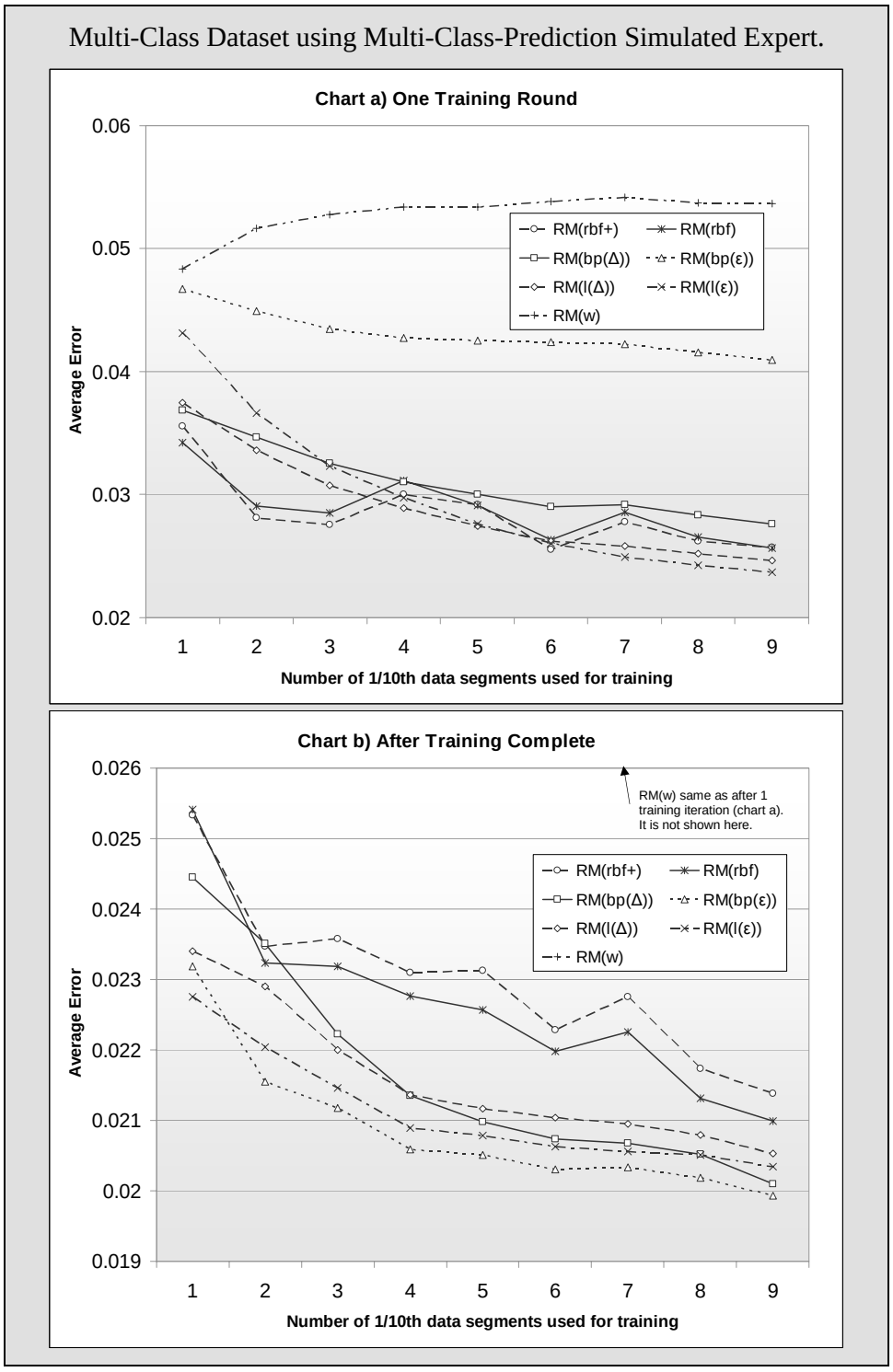


Figure 6-17 Two charts comparing how each of the seven proposed methods RM_w, $RM_{l(\varepsilon)}$, $RM_{l(\Delta)}$, $RM_{bp(\varepsilon)}$, $RM_{bp(\Delta)}$, RM_{rbf} and RM_{rbf+}, perform on the Multi-Class-Prediction dataset using the Multi-Class-Prediction simulated expert. Chart a) shows how the methods compare after only one viewing of the training set. Chart b) shows how the methods compare after training has completed. The X-axis shows how many tenths of the dataset were used for training. All used the last tenth for testing. The Y-axis shows the average error.

For RM, the act of predicting a value is significantly different to simply classifying a case. When RM was used for classification it was only required to find a value above or below zero for each class. However, when predicting, it must learn a precise value. Therefore, if the *Δ-rules* are going to cause problems, it is most likely going to occur during the prediction task. This is because the initial value can sometimes be different to the true value, and the system, therefore, may be pushed further away from the target rather than closer.

This concern has materialised in the unusual performance of the RM_w method. It can be seen in chart (a) that its simple value assigning actually performed marginally worse as it saw more of the dataset. This is clearly counter intuitive. It indicates a flaw in the value assigning rule, which to some degree may also occur in the more advanced *Δ-rule* based methods. It illustrates that when new rules are added to MCRDR, and a value is assigned, that the value calculated can cause already correct values to become incorrect when averaged with the new values assigned. Therefore, the more knowledge added causes more values to be included in the averaging function. When this is in a significantly non-linear domain it actually detracts from the overall performance.

The *Δ-rule* based initialisation methods would also be expected to suffer with this same difficulty. Interestingly, however, the above results indicate that both $RM_{l(\Delta)}$ and $RM_{bp(\Delta)}$ gained significant advantages over their randomly initialised counterparts, $RM_{l(\varepsilon)}$ and $RM_{bp(\varepsilon)}$, during the early stages of training. This contradiction is due to the RM_w method relying on a higher zeta, ζ, value (0.8) to compensate for the lack of learning capability. The *Δ-rule* methods can reduce the zeta constant (0.4, 0.2 for $RM_{bp(\Delta)}$ and $RM_{l(\Delta)}$) to prevent this difficulty, yet still have a high enough rate to maintain faster initialisation.

The improved learning initially, however, did not carry through to the eventually trained systems. The poorer performance after training is, however, extremely small. The results for the non-linear results are reproduced in Figure 6-18 with error bars. It can be seen in this chart that the small difference is even less relevant due to the large range of variation in the results. In fact, the p-value is only 0.05 – 0.4 across the nine tests, indicating there is only some weak statistical advantage in not using the *Δ-rule* on the fully trained system. There is, however, a very significant advantage to using the *Δ-rule* during the early stages of learning (p-values in the range 0.009 – 0.02).

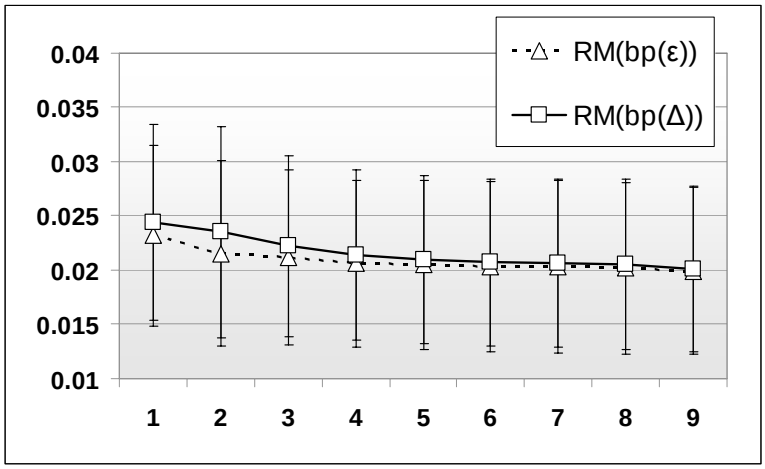


Figure 6-18: Comparison of $RM_{bp(\Delta)}$ and $RM_{bp(\varepsilon)}$ after training was completed on the Multi-Class-Prediction test. The X-axis shows how many tenths of the dataset were used for training. All used the last tenth for testing. The Y-axis shows the average error. Error bars are included for both data series.

The performance of the non-linear method against the linear version shows that the linear method has performed marginally better in the majority of the tests. There is some significance to this in the early stages of learning (p-values range from 0.002 – 0.6). This is due to the speed of learning of the linear method and allows it to gain a lead beyond the advantage already provided by the *Δ-rule*. This advantage does not continue after training. In fact there is an advantage to the non-linear approach after training, for the tests where more of the dataset was seen, which is approaching weak significance (p-value 0.17). These results appear to show that the non-linear approach may have an advantage in non-linear domains but it requires most of the dataset to have been viewed, in order to achieve this advantage.

The results in Figure 6-17 also show that the RBF method, while performing exceptionally well during the early stages, struggles during training to learn as effectively as the backpropagation methods. This poor learning is due to over fitting of hidden nodes during later training, which may be improved marginally by refining the hidden node creating rules. Another concern with the RBF method, however, is the variation in results, indicating a strong correlation between its performance and the type and order of the dataset. Also, additional data examples can harm learning rather than allowing for improvement.

6.2.2.2 Prediction Online

The tests performed for prediction online are essentially identical to those performed for the classification problem, except that instead of producing an output for each class it produces a single value. Like in the above generalisation experiments, the value returned is compared to the simulated expert's correct value. The absolute difference between these two values (*error*) is then averaged over all the cases in the data segment. The results in Figure 6-19, show how each of the seven methods perform on the multi-class-prediction dataset. Each point is the average error over the last 10 data segments for the 10 runs performed. The first point shows the average after 10 data segments followed by the subsequent points being after every 50 data segments.

These results once again reinforce earlier findings that the better performing methods are the linear methods, $RM_{l(\Delta)}$ and $RM_{l(\varepsilon)}$, and the non-linear method, $RM_{bp(\Delta)}$, using the *Δ-rule* initialisation. They also suggest that during the early stages the $RM_{l(\Delta)}$ method performed better than the $RM_{l(\varepsilon)}$ (only in the first group of iterations) and $RM_{bp(\Delta)}$. Although after some training, the three methods all converged to similar results. Certainly, there is little to suggest in these results that the extra complexity of the $RM_{bp(\Delta)}$ is warranted and the speed of learning of the linear method gives it an early advantage. With a much closer inspection of the last few hundred iterations it can be seen that the non-linear approach always performs, as well and on some of the results performed marginally better, than the other linear methods. While there is no statistical significance in this difference, the trend does continue through the following 3000 iterations (not shown). It is possible that the non-linearity could only be captured, or was only useful, in some groups of data segments.

It can also be seen that the RBF based methods once again started exceptionally well, but were unable to continue this with training afterward. Likewise the result for RM_w followed the same trend observed in the generalisation experiment. It can be seen that as it saw more cases its performance level decreased. As suggested previously, this is a result of its high zeta rate causing new values to cause older values to shift from their already partially correct value to one that is more incorrect. Due to it receiving no further training it was unable to correct the problem.

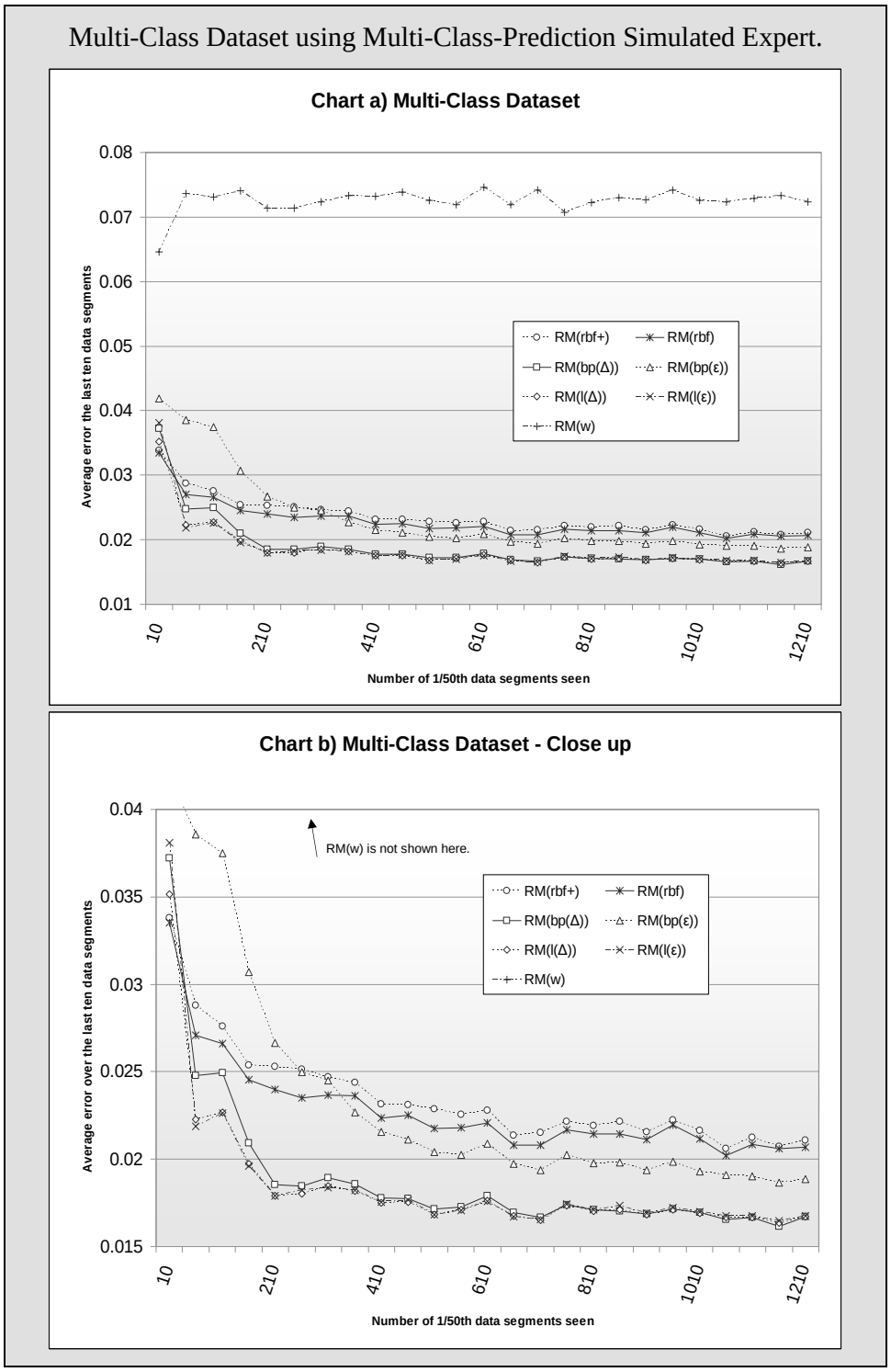


Figure 6-19 Two charts comparing how each of the seven proposed methods RM_w, $RM_{l(\varepsilon)}$, $RM_{l(\Delta)}$, $RM_{bp(\varepsilon)}$, $RM_{bp(\Delta)}$, RM_{rbf} and RM_{rbf+}, perform on the multi-class dataset using the multi-class-prediction simulated expert. The x-axis shows the amount of 1/50 data segments that have been seen. The y-axis shows the average error over the last 10 data segments. Each point is an average across ten runs. Chart a) shows the overall result. Chart b) shows a close up of the main six methods.

6.2.3 Further Discussion

This section has presented and discussed numerous results. This has been done to analyse the seven methods developed to verify which methods perform best in which situations. Across all of the results gathered it has been clear, that the most effective methods are those using backpropagation with the single-step-Δ-rule for initialisation. Although the RBF based methods did occasionally do as well, they only occasionally performed better. The primary difficulty with the RBF approach was the volatility of the method. It was extremely inconsistent in its performances and was very dependant on the order cases were presented. Furthermore, while the evidence was seldom significant the non-linear method, $RM_{bp(\Delta)}$ usually showed more potential.

For the purposes of this thesis one of these methods needed to be selected for further experimentation. This is because it would be unreasonable and unnecessary to test all the methods when they are not as effective. The issue in this section, was which one? It seems apparent, from the previous discussion, that the best methods to use would be $RM_{bp(\Delta)}$ and $RM_{l(\Delta)}$. While the linear method was tested in the later experiments, it was not ever the best performer (although it was often an equal performer). For these reasons the results in the remainder of this thesis will use $RM_{bp(\Delta)}$.

Initially, the first method developed and applied to the experiments in the later chapters had been the RM_{rbf+} technique. This was because, prior to being proved otherwise, it was expected to perform the best. After observing the results in this chapter many of the experiments were redone. This resulted in the majority of results being significantly improved upon. Therefore, while the method selected from the above results is the $RM_{bp(\Delta)}$, other methods have been tested but not presented to minimise the amount of results. Furthermore, it should also be noted that some results previously published (Dazeley and Kang 2004b; 2004c) have been updated in this thesis and so are no longer current. Where this situation occurs a footnote will be provided detailing the relevant changes.

6.3 Comparing RM against MCRDR and ANNs

The first hypothesis being addressed in this thesis was concerned with whether a hybridised technique combining MCRDR with an ANN would result in an improved learning system that was greater than the sum of its parts. So far we have only tested the different hybrid methods developed against themselves. In this section, this analysis will be extended to compare how the selected algorithm performs in comparison to its two original core techniques. This will be done by repeating the earlier experiments for both MCRDR and a neural network. In order to positively answer the hypothesis RM would need to learn as fast as MCRDR but also be able to apply this in a generalised manner like an ANN. This section has been broken up into three subsections. The first will discuss the ANN used for testing. This includes a description of what input is provided. The second section will discuss the classification abilities of the methods being investigated, while the third will look at the prediction abilities. It should be noted that MCRDR cannot be used for prediction tasks in any meaningful sense and so is not included in the last section's results.

6.3.1 Overview of ANN used

During this section the ANN used is a standard backpropagation network. This network of course suffers the same difficulties of all ANNs when parameters are selected. The parameters used for the results in this section are listed in Appendix D. Furthermore, decisions had to be made concerning the network topology and what input should be provided. All these decisions can affect the network's performance and it should be noted that while the best effort was made to make the most appropriate selection there is no guarantee that they are the most effective.

When using the multi-class dataset the network simply provided a single input node for each attribute. Any relationships between the attributes would need to be found by the network. This was used as it is the most intuitive and any neural network test on a dataset, such as this, would use this input method initially. The inputs for the network on the chess, tic-tac-toe, nursery, audiology and car evaluation datasets are slightly different. For these experiments the network had an input node for each possible value of each attribute. For instance,

each of the nine attributes in the TTT dataset can have an x, o or b (blank), therefore the network had 27 inputs. Therefore, only one input would fire out of every three representing that attribute. This is also a fairly standard input representation for these types of datasets.

The outputs for the network were identical to the method used for RM. Therefore, in the tests using the multi-class, nursery, audiology and car evaluation datasets the network is assigned an output for each of the possible classes. Thus, with six classes each method has six outputs. Then during testing, if the network delivers a value above zero then the current case is given that classification, otherwise it is not. In the tests using the chess and tic-tac-toe datasets each method only has one output. If the value is above zero then '*won*' and '*positive*' respectively is the identified class, otherwise the selected class by the systems is '*no-win*' and '*negative*'.

6.3.2 Classification

In the classification task we are concerned with the algorithm's ability to correctly identify in which category or categories each case belongs. Each algorithm has been tested on each of the seven datasets for both their ability to generalise and their performance in an online environment. These tests are separated into two sub sections: generalisation and online classification.

6.3.2.1 Generalisation

The ability of the methods to generalise is measured by how well they can correctly classify cases during testing. A method that cannot generalise will be unable to recognise anything about a new unique case. The performance of each method at generalisation in this thesis is gauged by its ability to classify unseen cases both after training and during training. In other words, the performance will be classified according to how fast it learns enough about the domain to be able to generalise. The results shown in Figures 6-20, through to 6-26 show how RM compares against MCRDR and an ANN on each of the seven datasets. Each chart shows the percentage of correct classifications, averaged over 10 trials, for each of the nine tests.

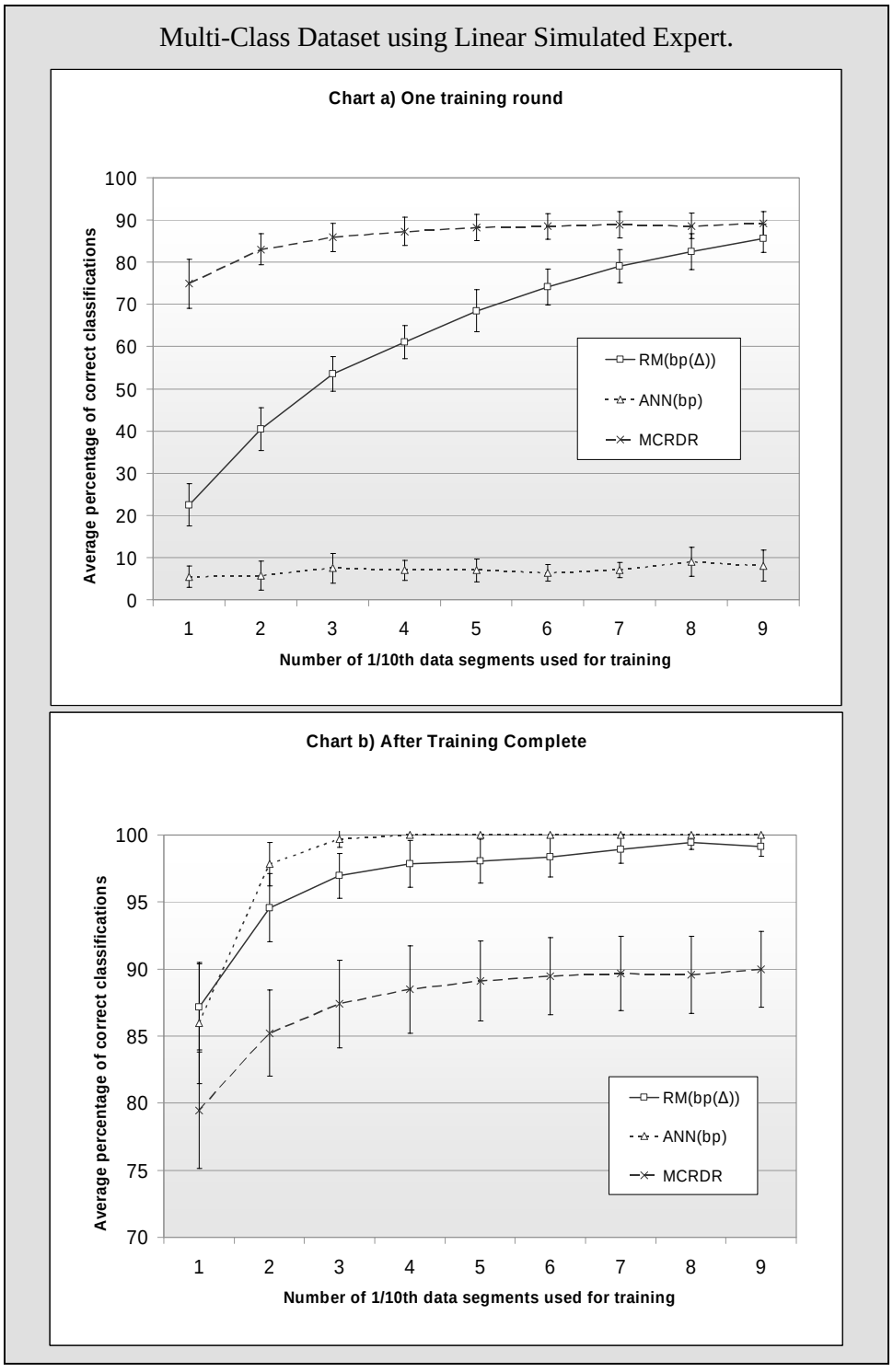


Figure 6-20 Two charts comparing how each method $RM_{bp(\Delta)}$, $ANN_{(bp)}$ and MCRDR, perform on the Multi-Class dataset using the Linear Multi-Class simulated expert. Chart a) shows how the methods compare after only one viewing of the training set. Chart b) shows how the methods compare after training was completed. The X-axis shows how many tenths of the dataset were used for training. All results used the last tenth for testing. The Y-axis shows the percentage of cases that were classified correctly (must get all six classes correct). Error bars are included showing the 95% confidence range.

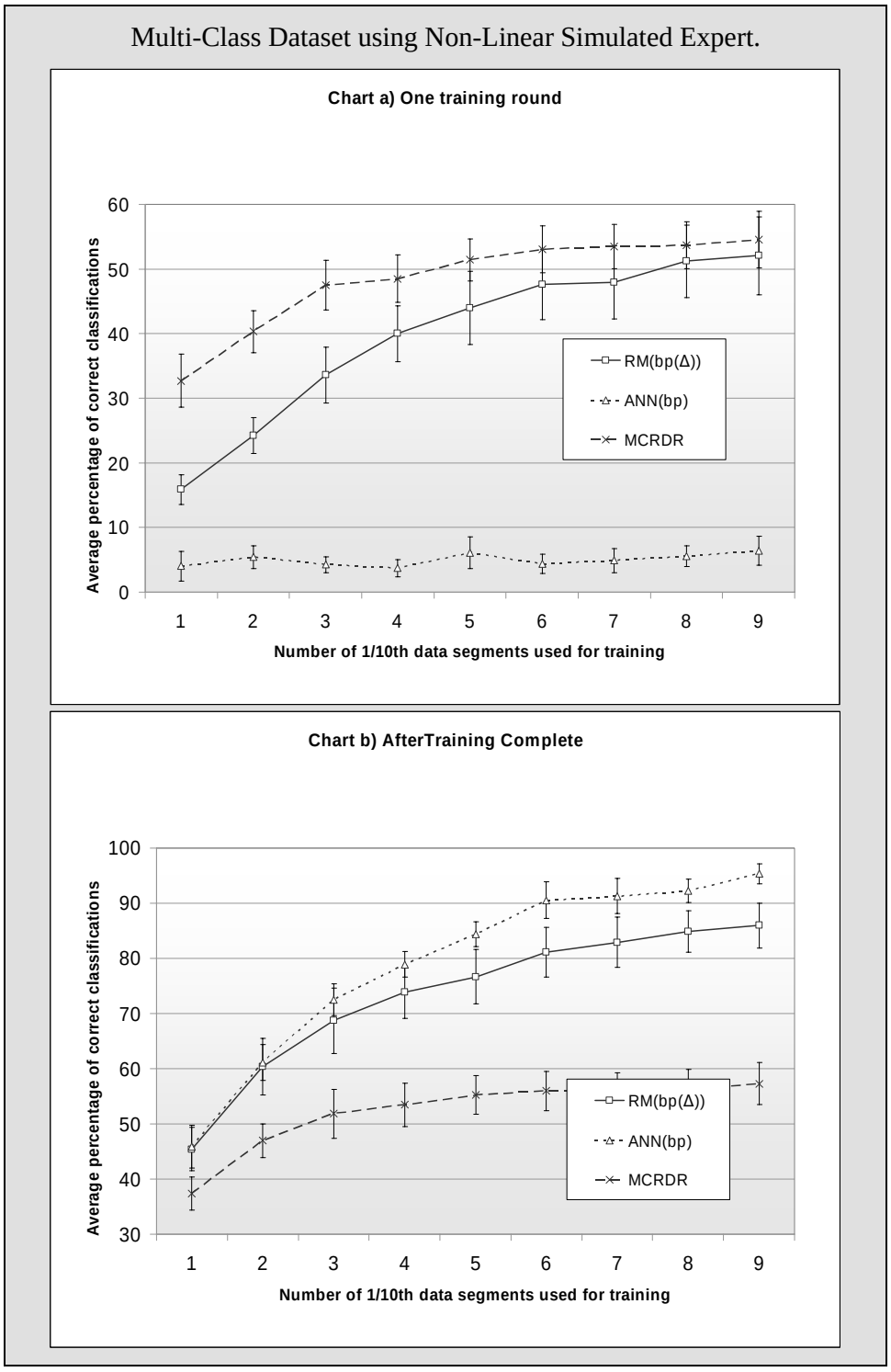


Figure 6-21 Two charts comparing how each method $RM_{bp(\Delta)}$, $ANN_{(bp)}$ and MCRDR, perform on the Multi-Class dataset using the Linear Multi-Class simulated expert. Chart a) shows how the methods compare after only one viewing of the training set. Chart b) shows how the methods compare after training was completed. The X-axis shows how many tenths of the dataset were used for training. All results used the last tenth for testing. The Y-axis shows the percentage of cases that were classified correctly (must get all six classes correct). Error bars are included showing the 95% confidence range.

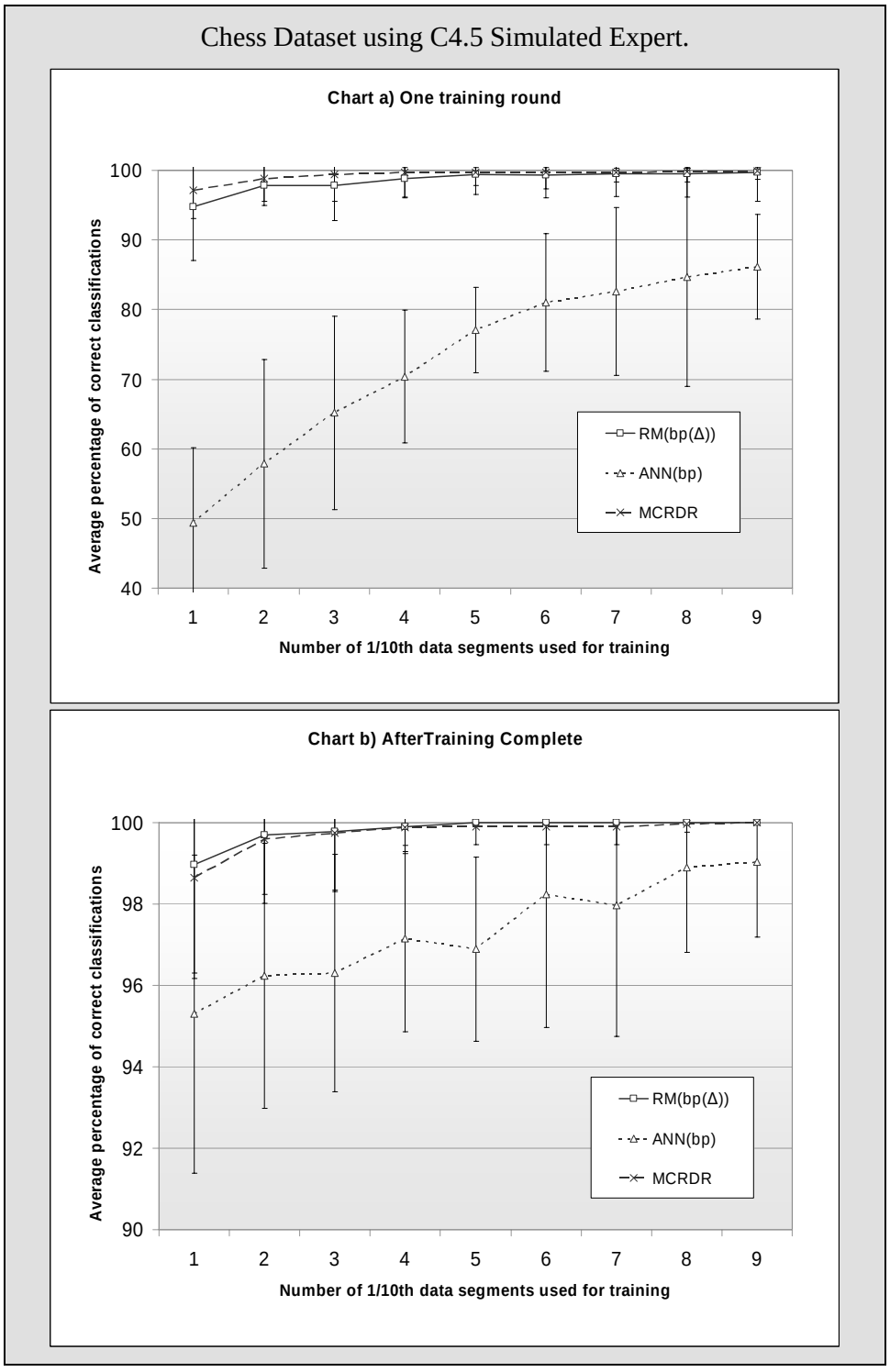


Figure 6-22 Two charts comparing how each method $RM_{bp(\Delta)}$, $ANN_{(bp)}$ and MCRDR, perform on the Chess dataset using the C4.5 simulated expert. Chart a) shows how the methods compare after only one viewing of the training set. Chart b) shows how the methods compare after training was completed. The X-axis shows how many tenths of the dataset were used for training. All results used the last tenth for testing. The Y-axis shows the percentage of cases that were classified correctly. Error bars are included showing the 95% confidence range.

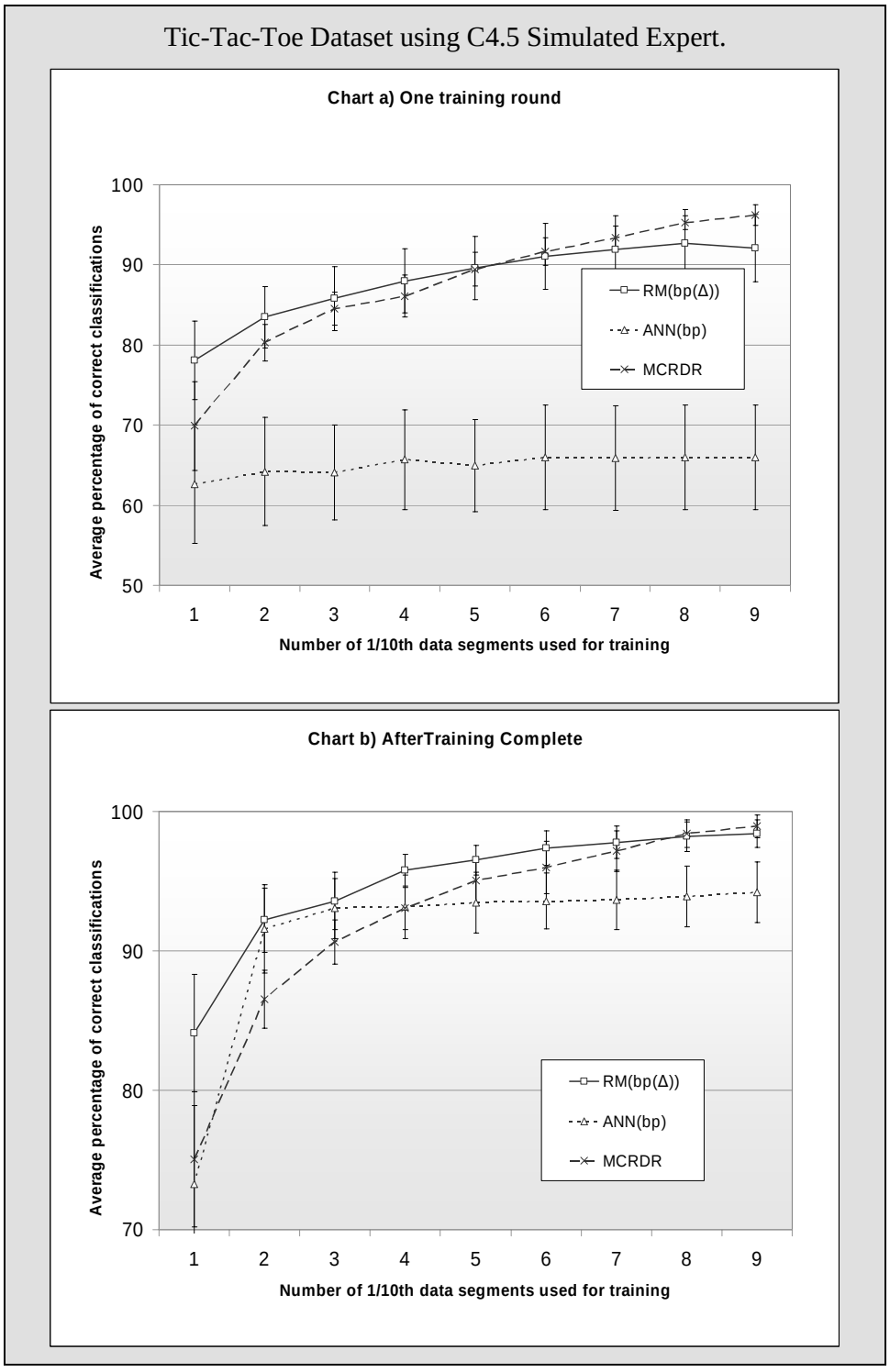


Figure 6-23 Two charts comparing how each method $RM_{bp(\Delta)}$, $ANN_{(bp)}$ and MCRDR, perform on the Tic-Tac-Toe dataset using the C4.5 simulated expert. Chart a) shows how the methods compare after only one viewing of the training set. Chart b) shows how the methods compare after training has completed. The X-axis shows how many tenths of the dataset were used for training. All results used the last tenth for testing. The Y-axis shows the percentage of cases that were classified correctly.

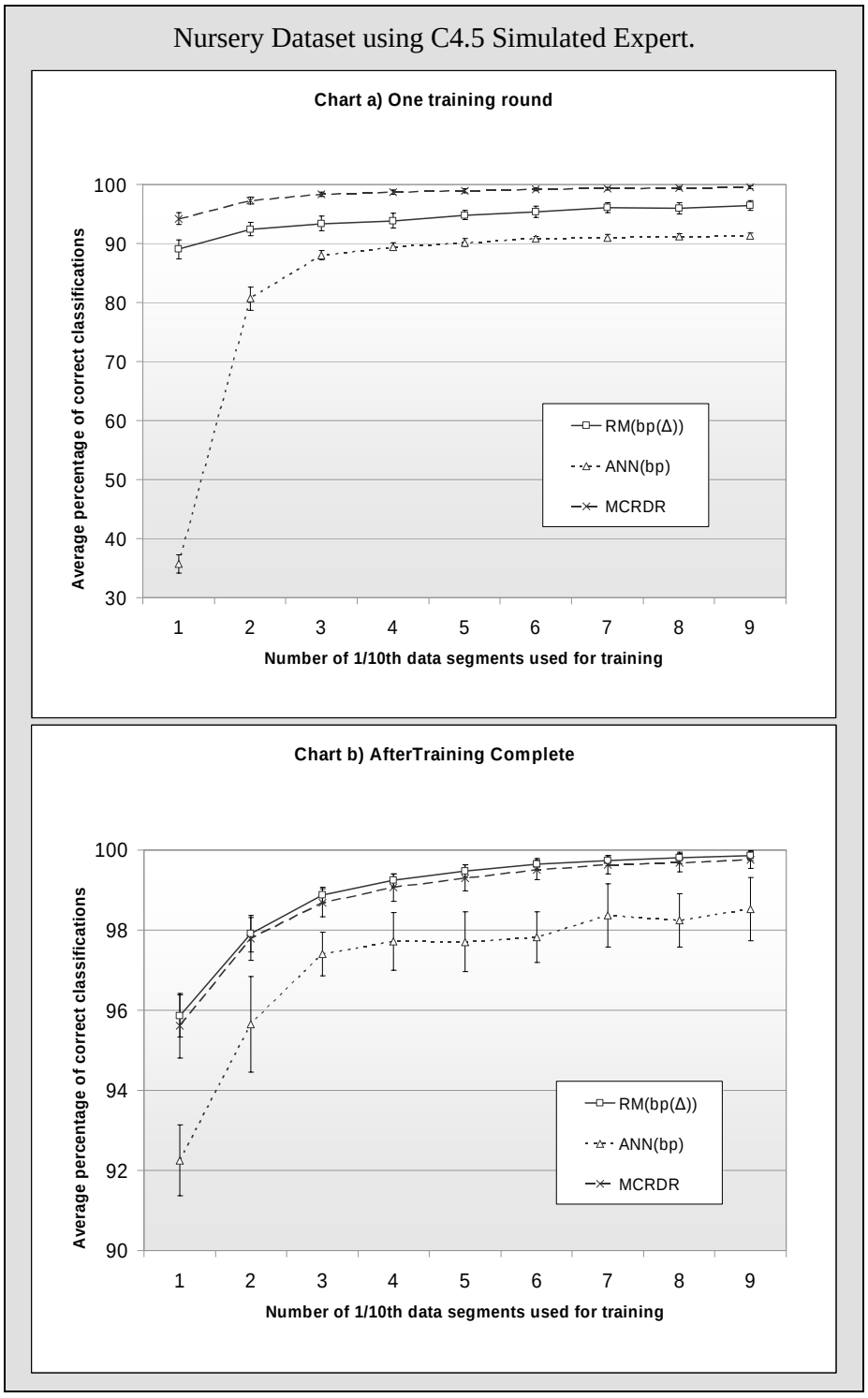


Figure 6-24 Two charts comparing how each method $RM_{bp(\Delta)}$, $ANN_{(bp)}$ and MCRDR, perform on the Nursery dataset using the C4.5 simulated expert. Chart a) shows how the methods compare after only one viewing of the training set. Chart b) shows how the methods compare after training has completed. The X-axis shows how many tenths of the dataset were used for training. All results used the last tenth for testing. The Y-axis shows the percentage of cases that were classified correctly.

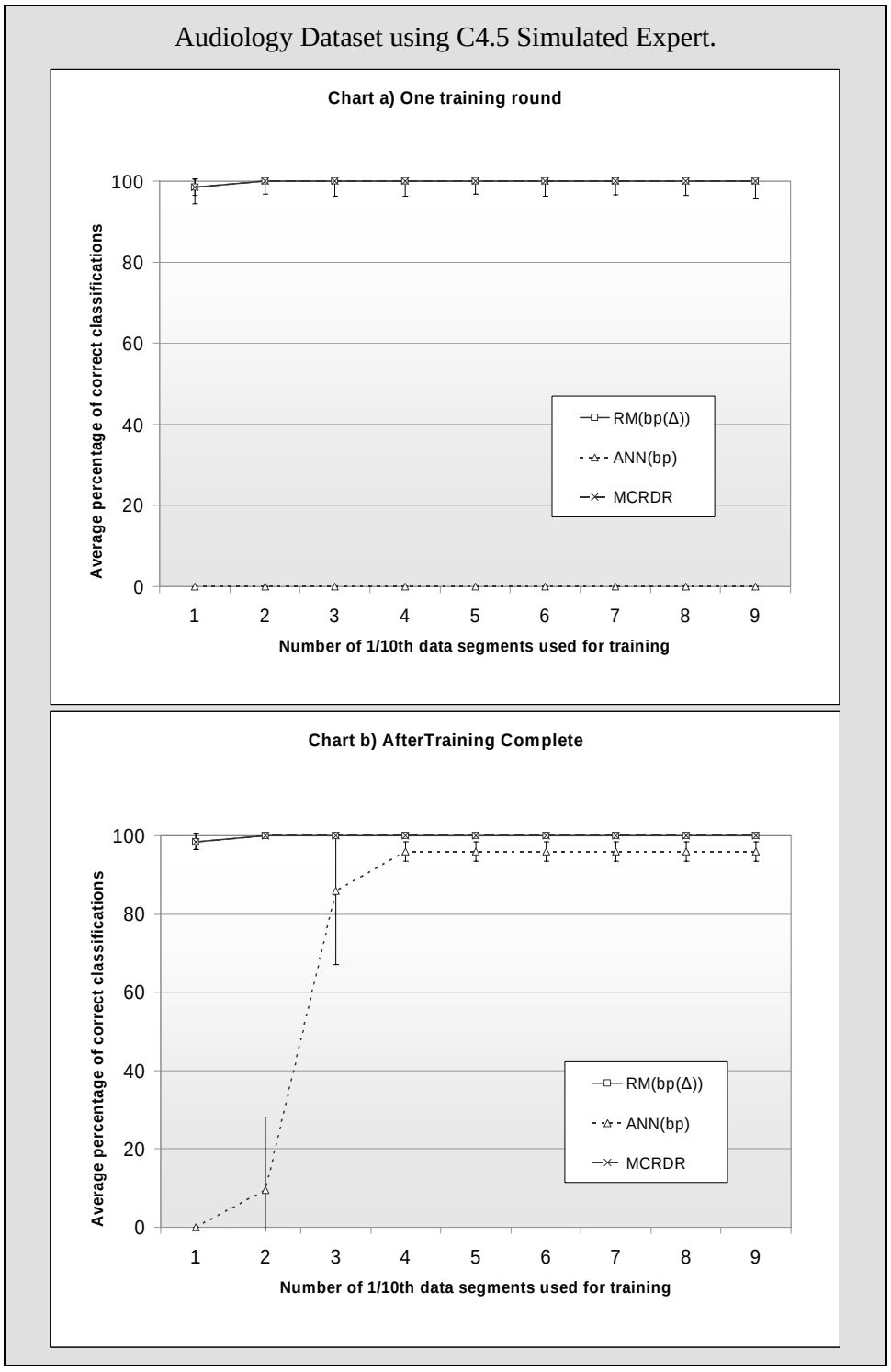


Figure 6-25 Two charts comparing how each method $RM_{bp(\Delta)}$, $ANN_{(bp)}$ and MCRDR, perform on the Audiology dataset using the C4.5 simulated expert. Chart a) shows how the methods compare after only one viewing of the training set. Chart b) shows how the methods compare after training has completed. The X-axis shows how many tenths of the dataset were used for training. All results used the last tenth for testing. The Y-axis shows the percentage of cases that were classified correctly.

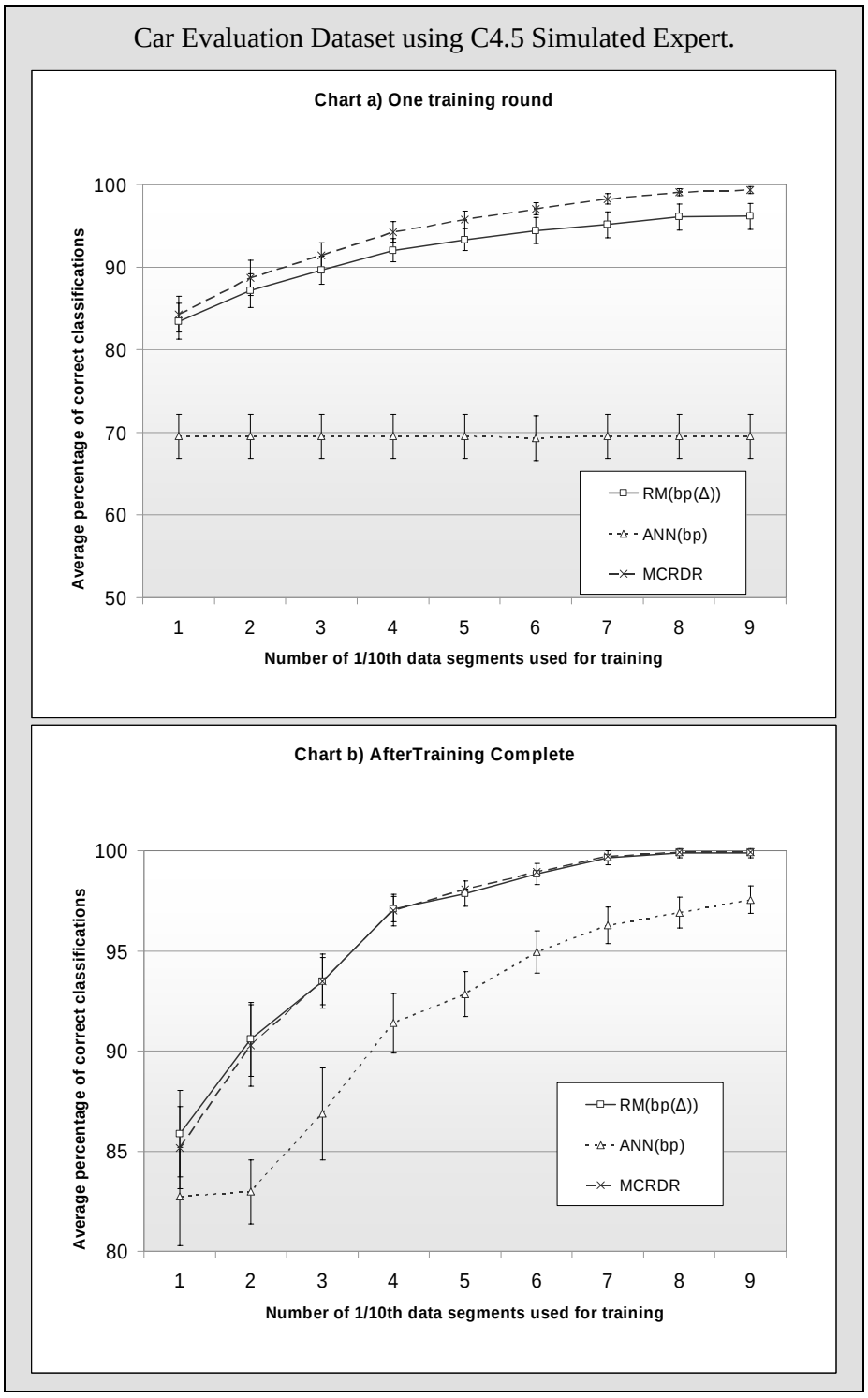


Figure 6-26 Two charts comparing how each method $RM_{bp(\Delta)}$, $ANN_{(bp)}$ and MCRDR, perform on the Car Evaluation dataset using the C4.5 simulated expert. Chart a) shows how the methods compare after only one viewing of the training set. Chart b) shows how the methods compare after training has completed. The X-axis shows how many tenths of the dataset were used for training. All results used the last tenth for testing. The Y-axis shows the percentage of cases that were classified correctly.

These results show that the RM hybrid system has done exceptionally well both initially as well as after training is complete when generalising. However, it was unable to perform as well as hoped. In both of the multi-class experiments RM very significantly outperformed the neural network. However, while it performed well, it was not able to match the performance of MCRDR. Also, after training was completed RM significantly generalised better than MCRDR but could not match the neural network. Although, it should be noted, that this was rarely a statistical significant difference in performance.

RM's performance on the chess, TTT, Audiology and Car Evaluation datasets (and to a lesser extent the Nursery dataset) was considerably better than in the multi-class environments. For instance, it was able to match MCRDR in the first iteration on a number of the datasets. Additionally, it was able to learn a much improved generalisation function over both the neural network and to a lesser extent MCRDR after training.

While RM was not always able to perform quite as well as hoped, some points should be noted. For instance, when it did not reach its full potential it did come close. Secondly, in both of the multi-class datasets, it was also found that after the second training iteration RM had outperformed MCRDR across most of the nine tests. It should also be noted that the results for the neural network had required significantly more training to get its marginally better results.

6.3.2.2 Online Classification

As previously identified, the aim of RM was to gain fast learning online and the ability to generalise. The results in this section investigate how RM compares with its two underlying methodologies in the online environment. Figures 6-27 through to 6-32 show how RM, MCRDR and the ANN perform on the six datasets. The multi-class dataset using the linear simulated expert was not used. Each point on the charts is an average of the previous 10 data segments (except 2 data segments for the Audiology dataset as it is much smaller) which are then further averaged over the ten randomised runs. Each segment contains a random selection of cases, each $1/50^{th}$ the size of the whole dataset. Error bars have been omitted to allow for greater readability.

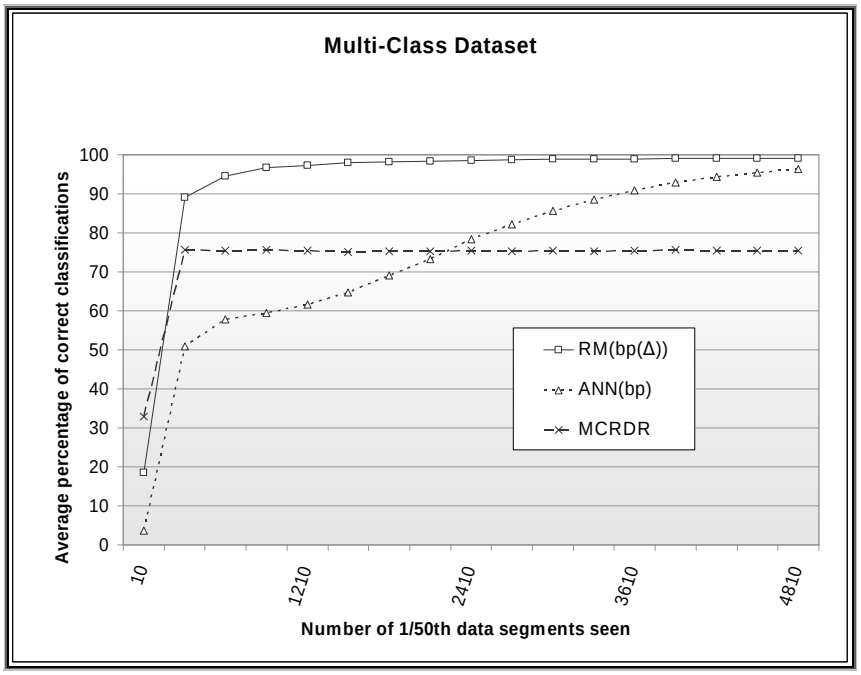


Figure 6-27: Chart comparing $RM_{bp(\Delta)}$ against MCRDR and an ANN using backpropagation on the multi-class dataset using the non-linear simulated expert. The x-axis shows the amount of 1/50th data segments that have been seen. The y-axis shows the average error over the last 10 data segments.

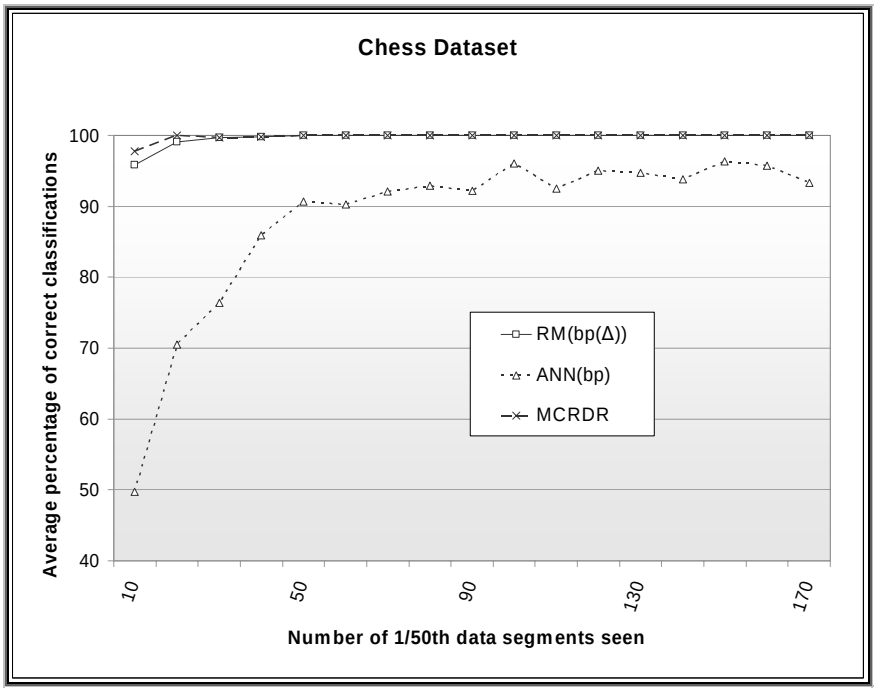


Figure 6-28: Chart comparing $RM_{bp(\Delta)}$ against MCRDR and an ANN using backpropagation on the chess dataset using the C4.5 simulated expert. The x-axis shows the amount of 1/50th data segments that have been seen. The y-axis shows the average error over the last 10 data segments.

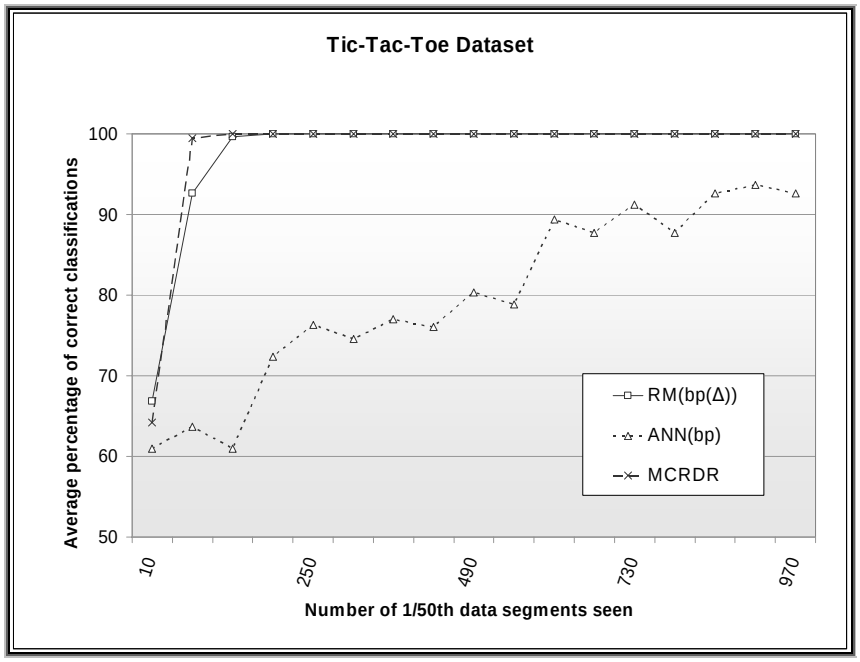


Figure 6-29: Chart comparing $RM_{bp(\Delta)}$ against MCRDR and an ANN using backpropagation on the tic-tac-toe dataset using the C4.5 simulated expert. The x-axis shows the amount of 1/50th data segments that have been seen. The y-axis shows the average error over the last 10 data segments.

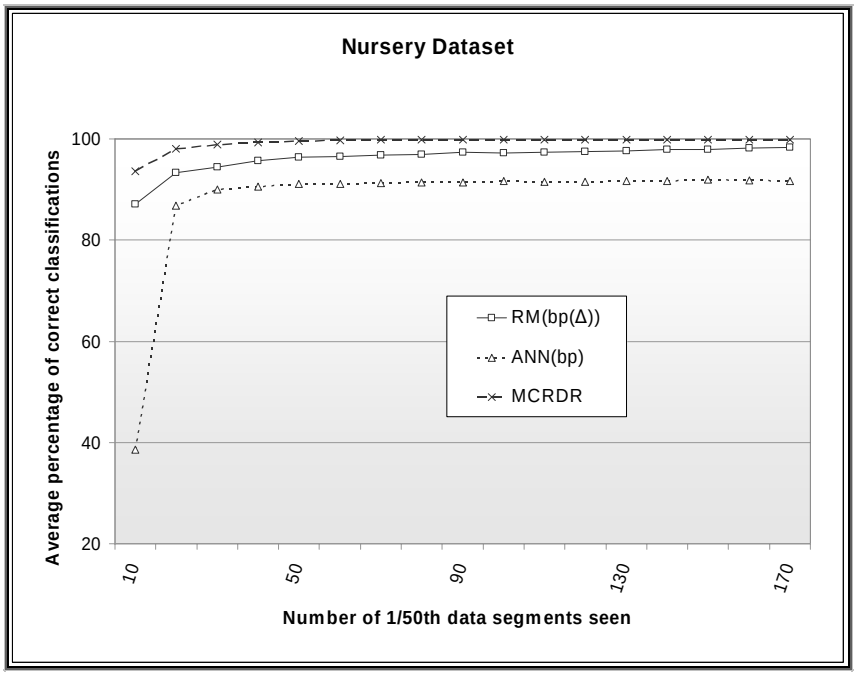


Figure 6-30: Chart comparing $RM_{bp(\Delta)}$ against MCRDR and an ANN using backpropagation on the nursery dataset using the C4.5 simulated expert. The x-axis shows the amount of 1/50th data segments that have been seen. The y-axis shows the average error over the last 10 data segments.

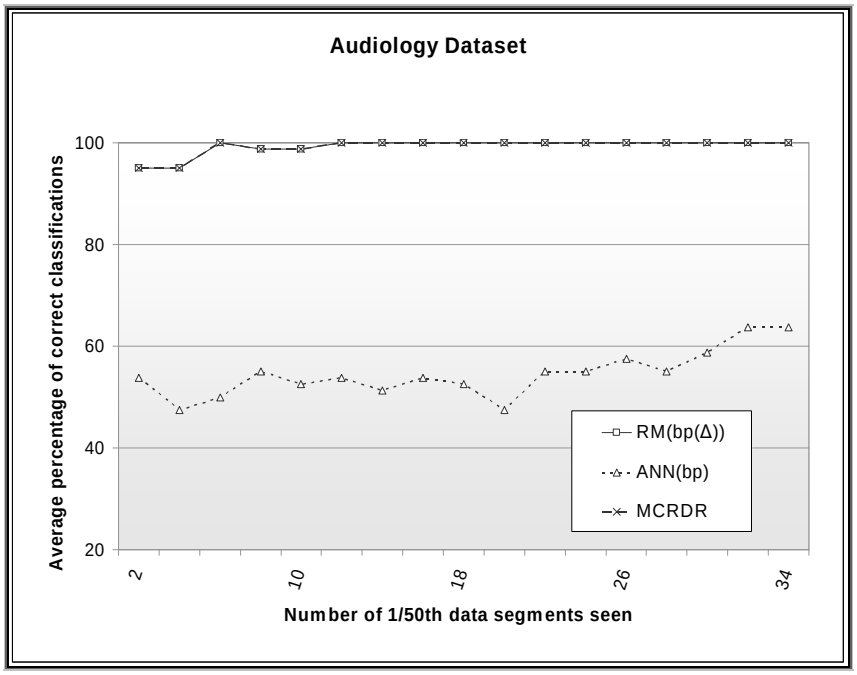


Figure 6-31: Chart comparing $RM_{bp(\Delta)}$ against MCRDR and an ANN using backpropagation on the audiology dataset using the C4.5 simulated expert. The x-axis shows the amount of 1/50th data segments that have been seen. The y-axis shows the average error over the last 2 data segments.

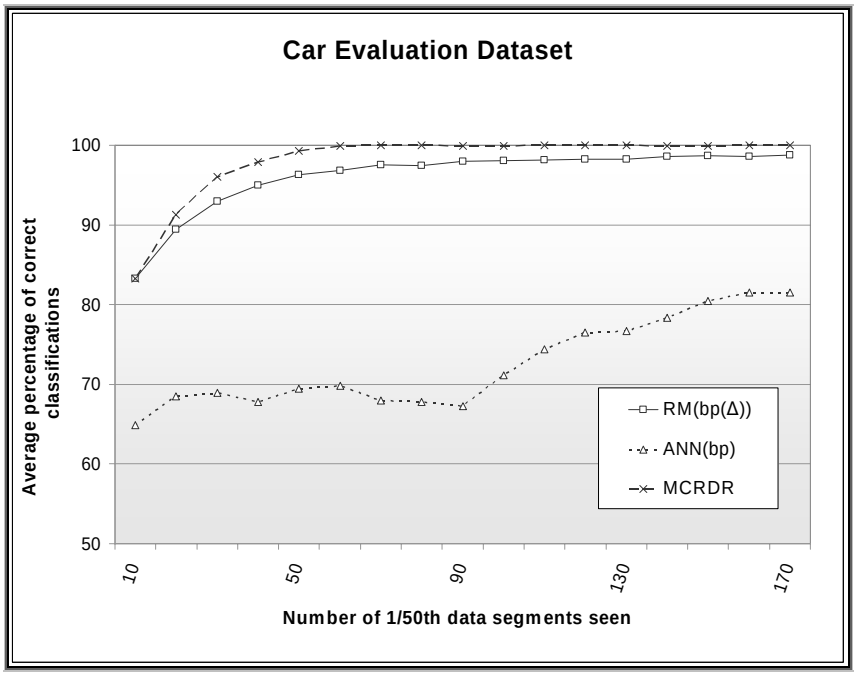


Figure 6-32: Chart comparing $RM_{bp(\Delta)}$ against MCRDR and an ANN using backpropagation on the car evaluation dataset using the C4.5 simulated expert. The x-axis shows the amount of 1/50th data segments that have been seen. The y-axis shows the average error over the last 10 data segments.

These comparisons are powerful indicators of the advantages of RM over a standard backpropagation neural network when being applied in an online environment. On the chess, TTT and audiology datasets it can be seen that RM has learned as fast or nearly as fast as MCRDR. On the nursery and car evaluation datasets it was only between 3% and 6% below MCRDRs performance and overtime was narrowing this gap. This meets our original goal of gaining the speed of MCRDR's instantaneous learning as soon as a rule is added. In the multi-class results this same result can be observed, except it can also be seen how RM continued to learn after MCRDR had accrued all its possible knowledge.

MCRDR's failure to continue to learn after its initial gains was a point of concern in the multi-class test. However, after investigation, it was found to be caused by two main factors. Firstly, for a case to be correctly classified it must get all six classes correct. Therefore, MCRDR's performance was not as poor as it first appears. Secondly, there is one unusual problem in the MCRDR rule creation and validation phase. It is possible that when an expert attempts to create a rule there may be no suitable attributes available. This generally only occurs on the later difference lists generated when there are multiple cornerstone cases. This problem was partly solved by Kang's (Kang 1996) incremental difference list generation method (3.3.2.2), but can still occur in rare situations.

The complexity of the multi-class dataset, especially the use of attribute pairs highlights this problem. This caused the simulated expert to be unable to create required rules on some occasions. One solution to this was to randomly select an unimportant attribute; however, this only compounds the problem and passes it on to a later stage of KB development. Therefore, for the purposes of this test, failed rules were not added to the knowledge base, partially limiting the method's performance.

However, these lost rules can now be treated as a form of hidden context. Therefore, RM's ability to significantly outperform the MCRDR's performance shows its ability to capture that hidden information even when it is unavailable to the knowledge base. The performance of RM appears to essentially learn exactly like any standard learning curve but rather than start from scratch it began from where MCRDR had finished learning.

6.3.3 Prediction

As discussed in section 6.2.2, MCRDR and other KBSs are generally only applied in classification domains. When they are applied to prediction problems they usually use the same basic classification process but with conclusions as values. This is highly problematic and does not allow for partial values. The huge advantage of RM in this domain is that it allows expert knowledge to be incorporated into a more general approach to prediction.

As detailed earlier, this means that the expert is still required to *'classify'* each case, as well as judge its accuracy in predicting the required value. Due to this restriction, our expert would be needed to both determine a form of classification, as well as a value. This meant that the C4.5 based experts could not be used and, therefore, neither could the chess and TTT datasets. Instead only the multi-class prediction expert was used (6.1.1.4).

6.3.3.1 Prediction Generalisation

This section's tests were performed in the same way as in section 6.2.2.1, where the neural network receives input in the same way as described in section 6.3.1. Figure 6-27, on the following page, shows how RM compares to an ANN using backpropagation in a generalisation test for value prediction. Each point on the chart is from the average across the last 10 data segments over 10 runs.

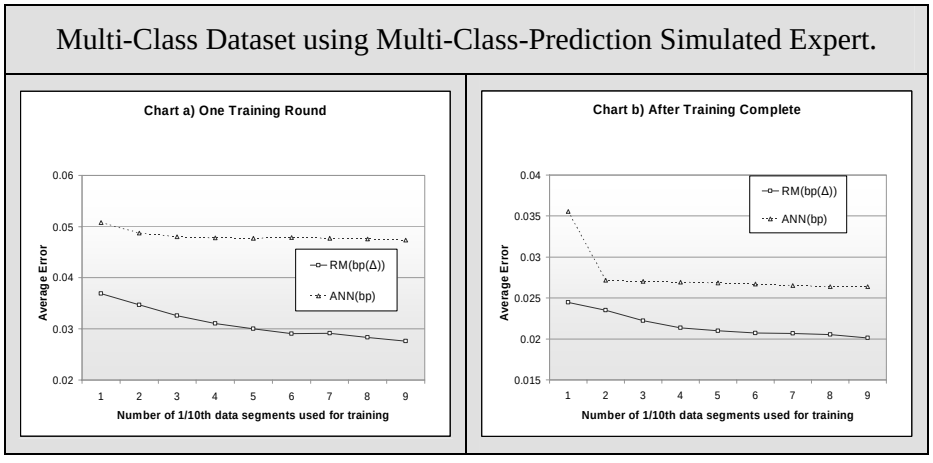


Figure 6-33 Two charts comparing how each method $RM_{bp(\Delta)}$, and $ANN_{(bp)}$, perform on the Multi-Class-Prediction dataset using the Multi-Class-Prediction simulated expert. Chart a) shows how the methods compare after only one viewing of the training set. Chart b) shows how the methods compare after training was completed. The X-axis shows how many tenths of the dataset were used for training. All results used the last tenth for testing. The Y-axis shows the average error.

These results introduce another interesting potential advantage of RM over the standard backpropagation method. Here it can be seen that the neural network was unable to continue to learn after it had reached a particular point. This problem is thought to be caused by the network having fallen into a local minimum, a problem common to neural networks, especially in prediction domains. RM is less likely to encounter this learning problem as the knowledge base provides an extra boost, similar to a momentum factor, which propels it over any local minima and closer to the true solution. Therefore, not only does RM introduce KBSs into potential applications in the prediction domain, as well as, allow for greater generalisation similar to an ANN, but it also helps solve the local minima problem.

6.3.3.2 Prediction Online

The process of RM being able to predict an accurate value in an online environment could potentially allow the use of RM in a number of environments that have previously been problematic. For instance, KBSs in *information filtering* (IF) have difficulties due to their problems in prediction, while neural networks are far too slow. RM allows for the inclusion of expert knowledge with the associated speed but also provides a means of value prediction. Figure 6-28 shows a comparison between RM and an ANN in an online environment.

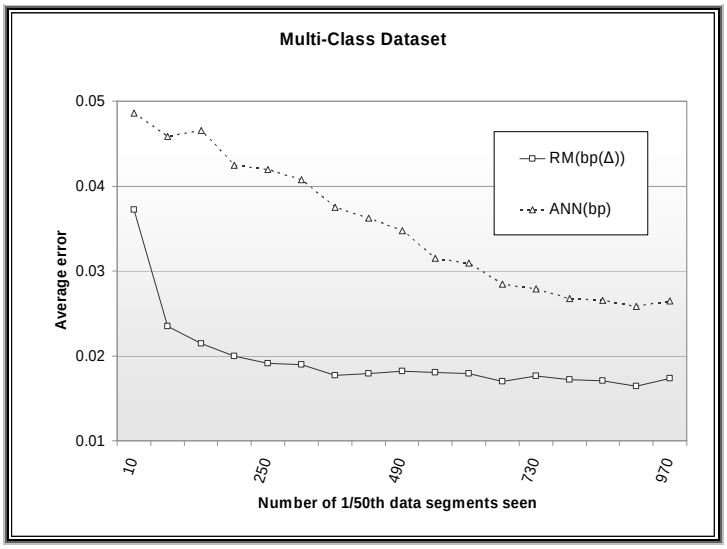


Figure 6-34 This chart compares how $RM_{bp(\Delta)}$, and $NN_{(bp)}$, perform on the Multi-Class-Prediction dataset using the Multi-Class-Prediction simulated expert. The x-axis shows the amount of 1/50th data segments that have been seen. The y-axis shows the average error over the last 10 data segments.

Once again it can be seen that RM has performed outstandingly well from the outset and was able to maintain this performance. This fast initial learning can be vital in many applications as it is what users usually expect. Additionally, although the chart only shows the first thousand iterations, the fully trained network was only able to achieve a best of 0.23, while RM was able to continue on learning to a minimum error of 0.14. Once again this could indicate the network fell into a local minimum during training preventing it from further improvement.

6.3.4 Further Discussion

Overall, the results in this section have shown how powerful RM is at acquiring knowledge and learning. These results highlight that it was able to generalise nearly as well as a neural network, thereby, allowing a reduction of the knowledge based systems brittleness, while still being capable of accruing knowledge at a rate comparable to MCRDR. This indicates a positive answer to the first hypothesis for both the classification and prediction tasks. The combination of these abilities makes RM unique in the field of classification and prediction.

RM's ability to perform so well can be put down to two features of the system. First, is that the flattening out of the dimensionality of the problem domain by the MCRDR component allows the system to learn a problem that is mostly linear even if the original problem domain was non-linear. This allows the network component to learn significantly faster. Second, the network gets an additional boost through the use of the single-step-Δ-rule, allowing the network to start more closely to the correct solution when new knowledge is added to the KB.

In addition to meeting the challenges set by the first hypothesis is the suggested evidence that RM moves beyond this in two key areas, offering significant advantages to other classifiers and predictors. First is the strong indication of RM's ability to continue learning about hidden contextual relationships and using this information to further improve classification. This is highlighted by its ability to learn beyond the knowledge stored in the KB. Second are the indications that the use of MCRDR, combined with the single-step-Δ-initialisation-rule, provides a means for avoiding local minima. Therefore,

RM not only fulfils the first hypothesis, but also represents a system that is significantly greater than the sum of its parts.

6.4 Summary

This extensive chapter served two primary purposes. The first was to fully test each of the seven RM methods developed in the previous chapter. This allowed the merits of certain design decisions to be analysed and for the best performing method to be selected; $RM_{bp(\Delta)}$. The selected method could then be compared to the underlying methods that were used to construct RM: MCRDR and a backpropagation neural network. This comparison allowed the chapter to test the selected method in two primary areas of general application for AI methodologies: classification and prediction. These tests where intended to be general in nature and not represent any one particular problem domain. The chapters in part 3 of this thesis will test RM in a specific problem domain, prudence analysis, and will show how RM's classification and prediction abilities allow it to perform these more specific tasks.

Initially, this chapter investigated which of the seven methods developed performed the best across a number of problem domains. The methods were tested on their ability to generalise and to learn online in both classification (6.2.1) and prediction (6.2.2) tasks. These tests' results showed that features such as the *single-step-Δ-initialisation-rule* were very effective in speeding up learning. This was due to its providing a significant head start as well as helping the network to avoid local minima. It was also found that most of the non-linearity of a dataset was absorbed by the *MCRDR* component allowing the network to learn faster due to having a primarily linear problem left to solve.

The later part of this chapter investigated how the best method, $RM_{bp(\Delta)}$, performed against the two components that were used in its hybridised construction (6.3). This was directly aimed at verifying the first hypothesis. It was found that $RM_{bp(\Delta)}$ learned nearly as fast as MCRDR during the initial stages (6.3.2). This fast learning allows the method to be applied in online learning tasks effectively. However, $RM_{bp(\Delta)}$ continued to learn beyond the knowledge *MCRDR* provided initially allowing it to generalise as effectively as the *ANN* after training was completed. It was also found that the method was able to

significantly outperform the *ANN* in a predictions domain due to its ability to avoid local minima (6.3.3). These results strongly suggest that the method developed in this thesis is better than the sum of its parts, fulfilling the first hypothesis.

Part 3:
Prudence Analysis

7

Discovering the Knowledge Frontier

'It all comes,' said Pooh crossly, 'of not having front doors big enough.'
'It all comes,' said Rabbit sternly, 'of eating too much…' *(Milne 1926, p25).*

The aim of this thesis was to develop a methodology that went someway to incorporating hidden and dynamic contexts, and thereby, meeting the *intermediate situation cognition* view of knowledge (chapter 2). Once developed, this was shown in the previous chapter to be highly successful in solving general classification and prediction tasks. These initial results show that the hybrid system has captured key advantages from each of the underlying techniques, as well as started capturing other hidden information not captured by MCRDR. These results by themselves suggest that RM can successfully be applied in any classification and prediction based domain, where human knowledge is available.

It was argued (chapter 1) that this additional contextual information captured by RM would not only improve classification and prediction, but also provide valuable information to other problem domains, such as Information Filtering, Data Mining, Natural Language Processing, and even Reinforcement Learning. However, rather than only doing a weak analysis of each of these, it was decided to perform a more significant investigation of the generation of prudence warnings. The provision of a working system for providing prudence based warnings, supplied by a dynamically context aware KBS, could significantly aid knowledge acquisition.

This chapter will first provide a brief overview of prudence analysis and the related field of research, verification and validation, along with an investigation of previous RDR based prudence systems. This will be followed by the development, testing and analysis of two prudence systems developed using RM. These results will be compared with a previous prudence based system. Finally, this chapter will investigate the effects of using warnings on the final KB.

7.1 Review

Prudence analysis is not one of the most widely known areas of study in KBS research. This section will provide a brief overview of the domain, providing an introduction to what prudence analysis is attempting to achieve in the area of brittleness and how it is related to verification and validation. Also discussed is what other work has been done in prudence analysis. At this time, work in RDR-based research is leading this field of study, thus, the systems discussed all stem from attempting to provide warnings with RDR-based KBSs.

7.1.1 Brittleness

Regardless of how intelligent we think we are, there will occasionally be that slip up - that embarrassing moment when we do or say something that reveals an extreme lack of knowledge about one particular detail that *everyone* else knows. This, fairly rare human ailment, has long been recognised as being a *frequently* occurring flaw in knowledge based systems (KBS). While such an error may make the person that made the mistake uncomfortable, it can usually be covered up or laughed at and then simply put aside. However, when a knowledge base makes this error it highlights a major error in a system's knowledge base and causes the system's user to lose faith in its ability to give an accurate and meaningful conclusion. The famous apocryphal example is the expert system that diagnosed a man as being pregnant (Compton et al. 1996).

This *brittleness*, often associated with KBSs, is founded in the system's inability to realise when its knowledge base is inadequate to provide an accurate and meaningful conclusion. The cause of such inadequacies is generally recognised as being due to the concentration of specialised knowledge in the target domain for the particular system (Lenat and Feigenbaum 1991). Within this particular domain they may perform exceptionally well, however, the moment some form of knowledge is needed from just outside of this domain their competence drops off quickly to complete incompetence. As the previous paragraph alluded, people are also subject to this difficulty. However, they have the ability to fall back on layer upon layer of general knowledge providing a much less precipitous slope (Op cit).

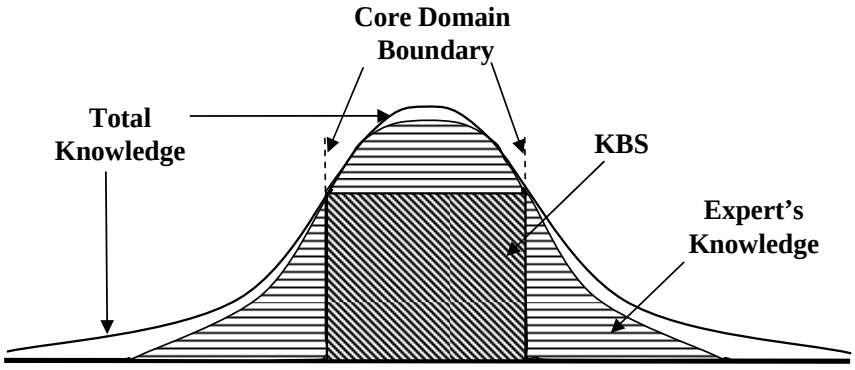


Figure 7-1: Conceptual diagram of knowledge distribution within a particular domain.

This can be viewed in the conceptual diagram, Figure 7-1, where the full knowledge needed for a particular domain is represented by the normal distribution. The choice of the normal distribution is to highlight that, while the vast majority of the knowledge required is in the core of the domain, the potential knowledge needed is much greater. It also shows that the further you move away from the core domain knowledge the less relevant and less likely that knowledge will actually be required. It can be seen in this diagram that the KBS generally has a very poor coverage of the general knowledge beyond the core domain as well as incomplete knowledge of the domain itself. This is primarily due to the inherit difficulties of current KA methodologies being unable to extract all of the expert's knowledge.

7.1.2 Verification and Validation

The issue of brittleness has been investigated from many possible sides, and as mentioned earlier was the catalyst for the development of the majority of KBS methodologies. One area of research moved away from finding better methods of acquiring knowledge, and instead, developed a means of checking whether a KB was complete. This process is commonly referred to as Validation and Verification (V&V) (Preece 1995).

Basically, the majority of V&V systems attempt to check a system by generating all of the possible cases for the data pattern in the domain that the KBS will operate. Each of these is then processed by the inference engine attached to the knowledge base and the conclusions are checked by a human expert. Generally, however, these techniques tend not to be particularly useful or cover a large range of data patterns. The larger the data pattern the more

frequently the expert will be asked to speculate about hypothetical situations of which they may have no experience and that may not occur in reality (Colomb and Sienkiewicz 1996; Compton and Jansen 1990; Compton et al. 1996).

In RDR, V&V has been an incremental process since the outset. As each rule is added it is verified immediately within the context it was added. However, Kang et al (1994; 1996) identify that this incremental V&V does not always ensure that old knowledge is in fact unchanged when new knowledge is added. This is because only the case that causes a rule to be created is stored and used as a cornerstone case. Other cases can be different and still be correctly classified yet when a new rule is added it is not validated with these cases that were never stored as a cornerstone case. This could be resolved by storing all cases. However, the reason only the first is used as a cornerstone case is to reduce storage, and more importantly lessen the validation effort required on the expert during KA.

Kang et al (1994; 1996), concerned about the theoretical potential of the V&V problem, performed tests finding that only 2% of old cases were incorrectly classified after new knowledge was added which is significantly less than the 10%-20% of errors on unseen cases. Many of the methods created since RDR have investigated V&V in relation to the proposed system. Additionally, one of the influences in the creation of the MCRDR methodology was the problem of V&V.

7.1.3 Prudence Analysis

Fundamentally, V&V is attempting to identify whether all the possible cases are covered by the KBS. However, as mentioned above, many cases will not occur and thus should not require validation. The incremental nature of RDR, however, provides the opportunity for a significant paradigm shift, through the possibility of identifying cases that are not valid during the inferencing process. Essentially, the system would need to be aware to some degree of what knowledge it currently possessed. Once a case is found that appears to require knowledge from outside the KB, it would provide a warning to the expert. These warnings, therefore, would be provided progressively allowing the expert to check the classification found.

This is potentially a very powerful tool that significantly reduces the KA effort for an expert because the expert will no longer be required to check all cases classified for correctness. Additionally, depending on the accuracy of the warnings this would go a long way to alleviating users' fears of harmful errors hidden in a KB due to brittleness. Also, in maintenance based systems, such as RDR, the ability to provide warnings at the time significantly aids the interactive nature of the methodologies.

7.1.4 RDR Prudence Systems

There have been a number of techniques used, which have attempted to generate a measure of prudence in RDR based methods. The first work carried out relating to prudence checking was known as WISE by Kang (1996) and Edwards (1996)[22], which, after inferencing in standard RDR was complete, would search the RDR tree for repeated instances of the conclusion found. The paths to these conclusions were then compared to test the usefulness of storing the history of corrections. It was found that apart from adding to the explanation ability of the KBS it may be possible to also use such information for prudence checking (Richards 1998a).

WISE was later extended using reflective learning through what Edwards et al (1995a; 1995b)[23] and Edwards (Edwards 1996), termed prudence and credentials. The system developed was called Feature Recognition Prudence (FRP) which tested if further inferencing beyond the similarly matched conclusions was possible and resulted in different conclusions. Such situations indicated that there was a possibility of an error in the original conclusion found.

A second approach taken at the same time for prudence detection was Feature Exception Prudence (FEP), where, after a conclusion was generated, the system looked at the database and flagged features of the current case that have not been previously validated by the expert as permissible. If any flags are generated then the conclusion is identified as potentially invalid. These systems were used on the LabWizard ES showing the potential of finding all errors. However, it did suffer from a large quantity of false positives (Edwards 1996;

[22] WISE was not published until Kang's and Edward's, PhD and Masters Theses respectively were submitted, even though they were created significantly earlier.
[23] FRP and FEP, which followed WISE, however, were published separately, prior to Kang's and Edward's, PhD and Masters Theses were submitted, and so have earlier publication dates.

Edwards et al. 1995a; Edwards et al. 1995b). A later study by Nadioo (1998) into prudence checking methods questions Edwards's results claiming they are highly reliant on the expert, order of cases and the nature of the dataset used.

While there are interesting features in Edwards's work, the fundamental problem is that there must already be knowledge within the KB for it to be able to predict missing knowledge, which does not truly solve the full prudence problem. Compton et al (1996) took a new approach of comparing cases with previously seen cases within context, and provided warnings if they differed in some unusual way. Basically, the method compared individual value-attribute pairs and warned if they exceeded what had been previously seen. This simple method achieved a reasonably high level of accuracy on some datasets with significantly less false positives. However, it concluded that the results were still not sufficiently accurate and still produced too many warnings to be applied in a real world application. Since Compton et al's work, little research has been published on further improving these results. Therefore, these published results are used in this thesis for comparison with the performance of RM when applied to prudence analysis in this chapter.

7.2 RM_p: Prudence Methodology

There are always a number of ways of applying any particular algorithm to a problem. Using RM for prudence analysis, referred to as RM_p, is no different. Therefore, no guarantee can be provided that any method used is the best approach. In this thesis there have been two methods developed to handle this particular domain. These are based on the work carried out in the previous chapter. Therefore, one method is designed around the RM prediction method, called $RM_{p(p)}$, and the second approach is based on the classification technique, referred to as $RM_{p(c)}$. This section of the chapter will provide a detailed description of these two methods developed and detail the reasons behind their designs. These two approaches are then compared against each other, as well as against Compton et al's (1996) results, in the experiments performed later in this chapter.

7.2.1 $RM_{p(p)}$: Prediction Method

The first method developed uses the idea of predicting a single value, which represents a *confidence-like* factor. This prediction is not a true confidence factor in the statistical sense, but rather, a rating or measure of whether the input pattern representing a case is likely to be an unreasonable or unseen pattern. The system begins with the assumption that all input patterns are unreasonable and always generate a warning. Over a short period of time it will learn which patterns are likely to be correct and stop warning on those patterns. This is essentially a simple online prediction task; learning to predict which patterns are correctly classified. In order to achieve good results in this learning task it is, therefore, paramount that the system learns quickly in order to reduce the amount of warnings.

As discussed in chapter 5, the neural network component of RM is attempting to learn patterns of rules, attributes and/or classes. When it receives a new input node, the pattern used to generate that input will be trained immediately to a high level of confidence. This can be done because the expert has essentially just confirmed the pattern when creating the new rules. If this pattern is then seen again it will return a high confidence. However, other patterns, not seen, will still only return a low level of confidence, causing a warning to be generated. Subsequently, if a new case uses the new input but the other inputs are different, then this new pattern defaults to a low confidence – causing a warning to be generated.

There were a number of questions needing answers when forming a workable design for developing this value-prediction based prudence system. What information was reasonably available for the system to use for training the network, when was this available and how should it be used? Furthermore: what value should new nodes be assigned during initialisation in order to achieve the required training advantages? Another question included: at what point should a low confidence actually be low enough to generate a warning? The decisions made, answering these questions will be discussed in more detail in the following subsections.

7.2.1.1 Training

One of the primary difficulties in applying RM to prudence analysis was the lack of information available to train the network component. Essentially, the only information that can be used is whether the expert has created a new rule or not. This was further limited however, because we can only use the cases that have generated a warning for training. Cases that do not result in a warning cannot be assumed to be correct and, therefore, cannot be used for training the network. This leaves relatively little information available to be used for training.

The training methodology developed is reasonably simple. Training only occurs when a warning has been generated. Then, if the user discounts the warning and accepts the offered classification, the level of confidence for the case's input pattern should be increased. Alternatively, if the user accepts the warning and creates a new rule, then the confidence level for the original pattern prior to the new rule being added should be trained down, while the new pattern with the new rule will be given a high confidence.

The second point of consideration is to decide what values produced by the network should be considered high, and therefore, which are low. Part of this problem is what kind of network range should be used. Generally, when using an ANN, little consideration is made as to whether the network should simply produce values in the range $0 \rightarrow 1$, or whether the sigmoid function should shift values down producing outputs in the range $-0.5 \rightarrow 0.5$. It was found, however, that the more common range of $0 \rightarrow 1$ failed almost completely. This was found to be because the vast majority of rewards given to the network are to increase the confidence level. Therefore, the network is constantly being trained towards 1. Eventually, the system always produces a very high confidence even for cases never seen.

The first solution was to use the $-0.5 \rightarrow 0.5$ output range. This allowed the system to treat the value 0 as low confidence and 0.5 or -0.5 as high confidence. In this approach, each pattern would either be trained towards the positive or negative ends of the scale. The end it was trained towards could be randomly chosen, thereby, insuring a roughly 50/50 split. This method meant patterns that had not been seen tended to be made up of both positive and negative values which then averaged out to a value close to 0, giving a low confidence.

This second method was significantly more effective but still suffered from the previous problem, just less frequently. Occasionally, all the feeds to the output were all positive or all negative causing a high average again. The final solution used solved this problem. In this approach the high confidence was taken to be either positive or negative 0.25. Therefore, low confidence was 0, 0.5 and -0.5. This allowed compounding averages to also produce a low confidence. Theoretically, the compounded averages could also occasionally equal ±0.25 as well, however this was found to be rare.

Therefore, the actual values used for rewarding the network are ±0.25 if the expert does not create a new node after a warning and a reward of 0.0 if they do create a new node. The choice for selecting a positive or negative value of 0.25 is done by looking at what the current value is. If the current value is positive then it is trained closer to +0.25. Likewise, if it is negative it is trained towards -0.25.

7.2.1.2 Initialisation Value

This reward structure also applies to the initialisations used by the network when a new node is added. Therefore, when the expert creates a new rule, after a warning was given, the system adds the appropriate new input nodes. These nodes are given initial values using the single-step-Δ-update-rule, where they step towards either ±0.25. The choice of positive or negative alternates with each new input node, thereby, ensuring the network remains balanced. Prior to this new node being added, however, the incorrect pattern is trained towards a low confidence ensuring that this pattern will create a warning next time it is seen.

7.2.1.3 Threshold

At some point the variable confidence factor must switch from low confidence, requiring a warning to high confidence, not requiring a warning. This threshold point needs to be located to best meet the needs of the experts using the system. For instance, a low threshold will result in very few warnings, but also poor accuracy, thereby missing many incorrectly classified cases and visa versa. In this study two types of thresholds were trialled.

The first was simply a static threshold. For instance, a threshold might have been 0.05. This threshold is then applied to either side of our high confidence point of ±0.25. Therefore, a warning is issued when the absolute rating is outside the value range 0.2→0.3. It was found, however, that in the early stage of prediction that only confidence factors that were very accurate should not be warned about, while later on only cases that were distant from the high confidence mark warranted a warning.

Therefore, a second approach of varying the threshold was also trialled. In this approach the dynamic threshold was increased when a warning was unwarranted and reduced when the expert created a new rule. Clearly, as fairly few rules are created this thresholding value will tend to increase in value over time, and therefore, less warnings will be produced later in the life cycle of the KB. Due to this constantly increasing threshold, some form of upper bound is required. This simple approach was found to be more effective than the static approach, although it does require the training rate for the threshold and a maximum value to be defined.

7.2.2 $RM_{p(c)}$: Classification Method

The second method developed employs the idea of using the neural network for classification comparisons. The basic idea is to allow MCRDR to develop classifications in the general way. Meanwhile, the network passively watches rules being added to the MCRDR tree. As it watches it also attempts to identify the correct classifications. However, as the network gets additional *a posteriori* information it can make improved classifications. The improved classification of the network, shown in chapter 6, is used as a dynamic check of the MCRDR classification.

As previously discussed in the results of chapter 6 concerning online classification, MCRDR and the neural network often produced a difference of opinion when classifying. This was especially apparent during the early stages of the KBs development. This can be seen more clearly in Figure 7-2, where the dark area shows when there was a difference of opinion on the multi class dataset. Additionally, when both MCRDR and the ANN misclassify a case (the light grey area) they are both likely to disagree.

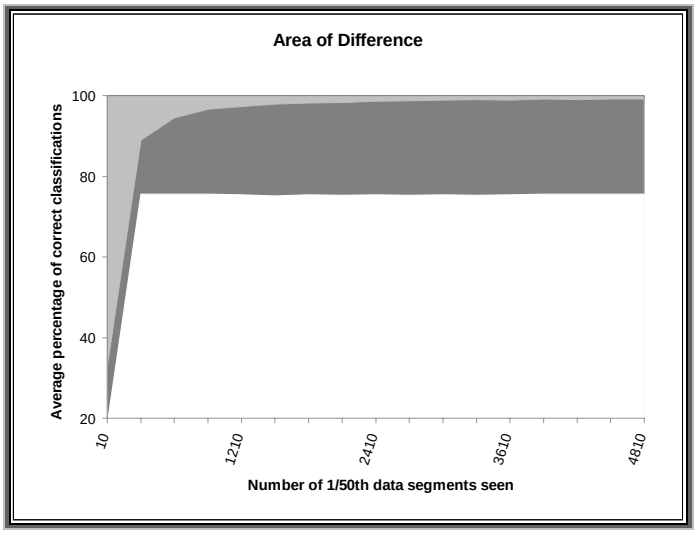


Figure 7-2: Re-formatting of results shown in Figure 6-18, where the difference between MCRDR and the attached neural network is shown. The white area shows agreement between MCRDR and the ANN. The dark grey area shows disagreement but where one is correct. The light grey area shows areas where both MCRDR and the ANN are wrong.

The $RM_{p(c)}$ method is designed to produce a warning whenever such a disagreement between the two classifiers occurs, allowing the user to correct the difference. One interesting aspect of this method is that it allows the provision of additional information for the user because now the system can inform the user which classification specifically is the likely cause of concern, and potentially what is the correct classification. These added abilities even further simplify the expert's task. This prudence method was developed after $RM_{p(p)}$ and turned out to be much easier to design and implement, with fewer difficulties and subjective design decisions being required. However, some simple issues still required resolution, which are described in the following subsections.

7.2.2.1 Training

Training is a simple process of identifying the correct classification that the expert has agreed to when accepting a case. Obviously, however, this can only be done when a warning has actually been generated. On the occasions when no warning is generated the system is unable to train because the system cannot be certain whether the expert would have wanted to alter the classification. The actual reward assigned is the same as in the previous chapter when learning classifications. That is, a positive value was given as a reward when it should have been classified as a particular case and a negative value otherwise.

7.2.2.2 Initialisation Value

Likewise, the initialisation value can be set in precisely the same manner as in chapter 6: stepping a good distance to the assigned value using the single-step-Δ-update rule. As shown in the previous chapter this is an effective manner of quickly learning a classification.

7.2.2.3 Thresholding

Unlike the value prediction method, the classification method has a conceptually natural thresholding point, 0.0, for each class. Using this threshold would literally result in the type of system described, where it warns when there is a difference between the two sub-components. While this does work with some success, it was found that often the two components matched but the network was not particularly decisive, meaning the value was just above or below the threshold. This was especially the case during the earlier stages of training. Therefore, the notion of a threshold was introduced to this method as well.

Thresholding in the classification method was done on a per class basis. Basically, if the MCRDR and ANN classes were the same, then a warning was not generated. However, if the network's absolute rating for a particular class was below a certain threshold then it was interpreted as the network being unsure of its rating, and therefore, a warning would be generated. This second method of warning only occurred when the network had the same result as the MCRDR inference engine. This simple tool was found to be highly effective at improving prediction.

In a similar fashion to the prediction approach, the thresholding value was found to only really be needed during the early phase of knowledge acquisition. Therefore, the thresholding value was also made dynamically adjustable. When a warning was warranted, the threshold was increased and when the warning was not needed, the threshold was reduced. Once again, it could not be adjusted when there was no warning generated. One change that could have been trialled was to have had separate thresholds for each possible classification. While it is unlikely to have an effect in the studies done in this thesis, a per-classification threshold would be needed in situations where classes are created during training.

7.3 Experimental Method

Two types of tests with four different datasets were performed to evaluate the success of the above prudence analysis approaches. The first test simply tested the ability of the methods to produce warnings accurately with the minimum amount of unnecessary warnings. The second looked at what effect on rule creation, and the eventual KB, occurs when the expert trusts the warning system and only checks cases that produced a warning. This section will provide a detailed overview of the experiments performed and the nature of the results gathered. Additionally, it will look at the simulated experts and datasets used in gathering the results. The following chapter will investigate how each of the methods performs at similar tasks using real world human knowledge.

7.3.1 Overview

The first experiment tested both prudence methods 10 times with each dataset randomly reordered. The results shown in the following section are an average of these 10 experiments. For each dataset tested there were small variations made to various learning parameters and the thresholds used to identify when warnings were given (Appendix D). The first test carried out did not use these generated warnings directly. Instead, it simply gathered statistics on how accurate its predictions actually were. Thus, the simulated user actually still checked every case, ignoring any warnings, and created new rules whenever it found a case incorrectly classified.

The aim at this stage was to test RM's ability to identify misclassifications. In the test discussed in this thesis, one of the following four details were recorded for each case as it was processed:

- If the conclusion was correct and RM gave a warning, then a *False Positive* (FP) was recorded.
- If the conclusion was correct and RM did not give a warning, then a *True Negative* (TN) was recorded.
- If the conclusion was incorrect and RM gave a warning, then a *True Positive* (TP) was recorded.
- If the conclusion was incorrect and RM did not give a warning then a *False Negative* (FN) was recorded.

This simple testing methodology is identical to that used by Compton et al's (1996) work on prudence analysis, thereby, allowing a direct comparison with the earlier system. From these results the accuracy of the system can then be determined. A successful prudence system would need to produce a high level of accuracy, while not producing an excessive amount of warnings. The best performing method was then used for further analysis in the second set of experiments.

The second experiment in this chapter tested how different the KB would be when the expert only checked cases that were first warned about by the prudence system. These experiments were designed to determine whether the misclassified cases that were not warned about caused a major reduction in the accuracy of the completed KB. For instance, it was suspected that if the system misses a warning and the expert does not correct that particular case, a compounding series of misclassifications could be caused. This potentially exponential problem could result in a significantly incomplete KB. If this turned out to be the situation then even a high level of accuracy may not be good enough. Such a result would not only negate this method's approach, but show that any form of prudence analysis is fundamentally flawed.

To determine whether this conceived problem is actually a flaw in the theory of prudence analysis, two experiments were performed. The first is essentially the same as the previous test, except the call to the expert to test the correctness of a classification and to correct it, if it was wrong, is only made when a warning has been generated. In this test we are interested in how big a difference there is in the size of the knowledge base compared to the system where the expert checks every case.

The second test is essentially a repeat of the generalisation test for classification in the previous chapter. In this test the system where the expert only checks warned cases is compared against the approach where the expert checks every case regardless of the warnings generated. In each series of tests the KBs are built on $9/10^{ths}$ down to $1/10^{th}$ of the dataset and tested on the tenth segment. It is unlikely that the system relying only on the warning system will perform as well, but it is hoped it will not perform significantly worse. If the warned expert approach can come close to the complete KB, then prudence analysis and in particular this approach to prudence can be regarded as viable.

7.3.2 Simulated Experts

This chapter used the same simulated experts that were described in section 6.1.1. Therefore, once again the C4.5 expert was used for the majority of datasets, chess, Tic-Tac-Toe and the new dataset introduced in section 7.3.3, GARVAN. Also, the Linear Multi-Class simulated expert was used for the Multi Class dataset. Both of these simulated experts are setup and operate in the same way as previously described (6.1.1).

7.3.3 Datasets

Once again the same three datasets are used as described in the previous chapter (6.1.4). The two standard datasets, chess and Tic-Tac-Toe have been selected as they are the same as used by Compton et al (1996), thereby allowing a direct comparison. The multi-class dataset used was expanded to include a variable number of attributes per case. Therefore, instead of each case having 6 attributes (6.1.4.2) they now contain between 3 and 9 resulting in 3938 possible cases.

All of these datasets are reasonably small in size, therefore, a forth dataset was introduced allowing the prudence system to be tested on a much larger problem referred to as GARVAN. This dataset was gathered by the GARVAN-ES1 expert system over a number of years and provides medical diagnoses for thyroid problems. However, the actual version of the dataset used in this thesis differs from the full version significantly. The version of the GARVAN dataset used in this study is the same one used by Compton et al (1996), thereby, allowing a direct comparison to be made.

The dataset used differs from the full dataset in two ways. Firstly, the full dataset contains 43,472 cases, while the version used here contains 21,822. The full dataset was taken over the full life of the GARVAN-ES1 system. However, during the period of its operation some additional medical tests were added resulting in changes to the attributes. These changes caused a major shift midway through the full dataset. This is a problem when randomising the order of the dataset for each trial. The smaller version only contains cases from one extended period of steady data. The second alteration was the discretization of a number of continuous attributes. This discretization was performed prior to Compton et al's (1996) work and was not performed as part of this thesis.

7.4 Prudence Results[24]

The ability of RM_p to predict which cases are outside its knowledge base can be measured through a combination of accuracy and the number of warnings. For a prudence system to be viable it requires a high degree of accuracy, while keeping the amount of warnings to a minimum. Table 7-1, shows results for each dataset tested with both prudence systems and averaged over ten randomised runs and rounded to two significant figures. The full results for the classification based method are shown in section 7.4.4[25]. The furthest left column identifies which dataset the results were from, while the second column shows the method used for prudence analysis. The last column is highlighted as it is a derived column, calculated from the first two columns of raw results.

Datasets	Algorithms	False Neg %	True Pos %	False Pos %	True Neg %	Accuracy %
GARVAN	$RM_{p(p)}$	**0.36**	**1.5**	**18**	**80**	**80**
	$RM_{p(c)}$	**0.13**	**1.7**	**15**	**83**	**93**
Multi	$RM_{p(p)}$	**0.94**	**2.4**	**31**	**66**	**72**
	$RM_{p(c)}$	**0.11**	**3.3**	**10**	**86**	**97**
Chess	$RM_{p(p)}$	**0.21**	**0.75**	**12**	**87**	**78**
	$RM_{p(c)}$	**0.19**	**0.79**	**8.4**	**91**	**81**
TTT	$RM_{p(p)}$	**4.5**	**8.4**	**31**	**56**	**65**
	$RM_{p(c)}$	**2.9**	**8.8**	**12**	**76**	**75**

Table 7-1: Comparison of the averages between the two prudence analysis systems developed. These results have been rounded to 2 significant figures. The four columns of raw results are shown plus the calculated accuracy.

[24] The results in this section differ to those published in Dazeley and Kang (2004b, 2004c). Previously, published material had used the RM_{rbf+} method which has since been found to be generally inferior. As discussed in Chapter 6 this thesis is only using $RM_{bp(\Delta)}$ in the prudence part of the project. Additionally, the environment used in those earlier results had mistakenly not matched Compton et al's (1996) work as claimed at the time. The environment was still legitimate but significantly simplified, involving repeated viewings of cases. Compton et al's (1996) work would theoretically perform the same regardless of how many times it viewed the case. However, RM improves performance with additional viewings. Also, training had been permitted on cases not warned about by the system, which it would not be able to receive when actually used for prudence analysis. These two factors allowed significant extra training, which had provided RM_p with an advantage and inflated the accuracy and number of true negatives. These issues have been corrected in this thesis and the results presented here are based on a single viewing of each case, without training on cases not warned about. Training is still performed when a new node is added even if a warning was not given. This, however, cannot be avoided and it also matches Compton et al's (1996) work as their system also acquired knowledge when a warning was not given. Therefore, these results provide a more accurate comparison with Compton et al's work.

[25] The results for $RM_{p(p)}$ are the only ones gathered. The results for $RM_{p(c)}$, however, are only a selection, chosen for this section's discussion. The results selected here are using parameter settings that generated results that most closely matched Compton et al's (1996) level of warnings. For the multi-class dataset the selected result was the furthest from the dotted random result line Figure7-5, indicating it was the best all round performer.

7.4.1 Accuracy

The level of accuracy can be calculated by dividing the amount of cases that were warned about and for which the expert created rules, *true positives*, by the total amount of cases that needed to have new rules added, *true positives* plus *false negatives*. The level of accuracy for each method, including Compton et al's (1996) results, are compared for each dataset in Figure 7-3. It should be noted that Compton et al's system was never tested on the multi-class dataset.

It can be seen in these results that the $RM_{p(c)}$ prudence technique has outperformed Compton et al's (1996) results across all datasets except the chess dataset where they performed equally well. One interesting result was its success on the harder datasets. This indicates that the more complex applications have a greater degree of relationship information available during training. This additional information aids the prediction of misclassifications. In situations where there is less information the system struggles more to determine when a rule may be needed. For instance, on the TTT dataset especially, but also on the chess dataset, all methods struggled to identify warnings correctly. These results are interesting as they indicate that the more real world datasets may still be able to achieve good accuracy even though the little fake domains cause problems.

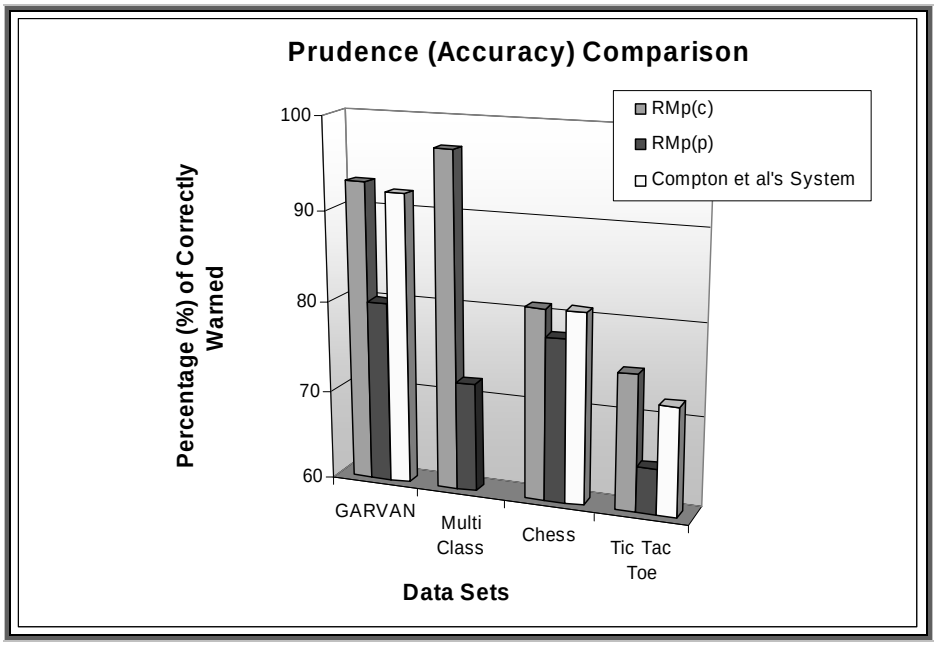


Figure 7-3: Chart comparing the accuracy of the two systems developed in this thesis: $RM_{p(p)}$ and $RM_{p(c)}$ against Compton et al's (1996) results for each of the four datasets.

The three datasets where a comparison is made to Compton et al's (1996) system are not the ideal domain for RM. RM is designed for multiple classification domains where it received much more useful information. The results in Figure 7-3 show that RM performed significantly better in this domain over all of the other results.

The $RM_{p(p)}$ approach performed reasonably on the smaller chess datasets, but was relatively unsuccessful on the others. It is believed that the single output does not provide enough information for identifying misclassifications. In the complex multi-class dataset, where many inputs in the network fire, $RM_{p(p)}$ bounces wildly never being able to train towards a solution.

7.4.2 Number of Warnings

Having a good level of accuracy by itself is, however, easy to achieve. Simply warning every case will get perfect accuracy. The key is to achieve this high level of accuracy while being able to reduce the number of warnings generated. Therefore, we also wish to maximise the amount of *true negatives* found by the system. While this information is in the above table it is shown again in a column chart in figure 7-4 along with Compton et al's (1996) results.

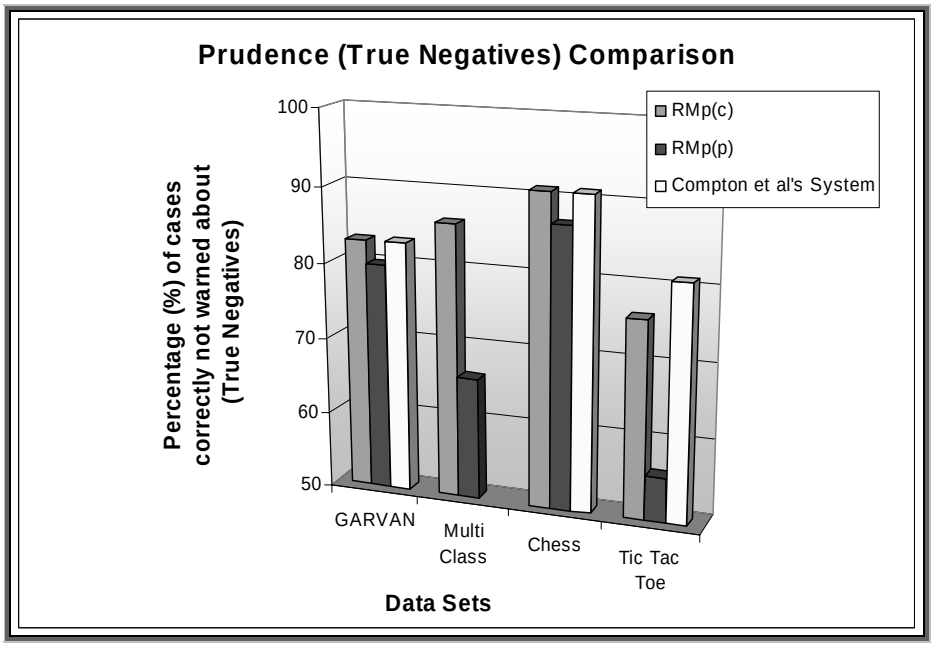


Figure 7-4: Chart comparing the percentage of true negatives of the two systems developed in this thesis: $RM_{p(p)}$ and $RM_{p(c)}$, against Compton et al's (1996) results for each of the four datasets tested. The higher the column the less warnings that were generated.

The number of cases not warned about by $RM_{p(c)}$ is identical to Compton et al's (1996) system, across all datasets except TTT. This is because the results selected in Table 7-1 were selected because they had the same amount of *true negatives* as the earlier prudence work. By selecting results with the same amount of warnings it is easier to see the small improvement in accuracy (7.4.1) of $RM_{p(c)}$. As will be discussed in the next section, this selection was possible because RM actually allows the expert to determine which is of greater importance, accuracy or the number of warnings.

$RM_{p(c)}$'s results on the TTT dataset were, however, unable to match the earlier work in relation to the number of *true negatives*. This was because $RM_{p(c)}$ was unable to reduce the amount of warnings to the level required. $RM_{p(c)}$ compensated this by producing a much higher accuracy with the TTT dataset than Compton et al's (1996) system. $RM_{p(c)}$'s and Compton et al's (1996) low performance on the TTT dataset shows that highly ridged datasets where each case is unique are difficult domains for any prudence system. It is, however, very unlikely that any real application for any prudence system would be like the TTT dataset.

RM kept warnings quite low for the multi class dataset. While not as high as the results for the chess dataset it achieved this with a much higher degree of accuracy. The results for the multi-class dataset indicate that in the ideal environment for RM it can achieve significantly higher accuracy with a reasonable number of warnings. All the above results indicate that RM performs slightly better than earlier work in the domains not suited to it, but significantly better in its ideal domain. Chapter 8 will investigate this further by testing RM in a real world multi-classification domain instead of the fabricated set used here.

Lastly, it can be seen that the number of cases that did not generate a warning were quite low for the $RM_{p(p)}$ method, compared to $RM_{p(c)}$ and Compton et al's (1996) system. Even though $RM_{p(p)}$'s performance on the chess dataset was reasonable, it was found that the classification method was much more effective at learning not to warn and does show a marked improvement over the prediction based results. Generally, the prediction based approach failed and is no longer considered in the results gathered in this chapter.

7.4.3 Total Error

Overall the above results indicate that the $RM_{p(c)}$ method was able to gather additional information and that this information can be used to provide reasonably good prudence analysis. One interesting observation of these results is that the total number of wrongly classified cases that were not warned about is very small. For instance, on the GARVAN dataset there is only 0.1% of errors over the entire knowledge base life cycle that the expert was not warned about. This is remarkable, especially when you consider that the KB initially started with no knowledge. This equates to a knowledge base with an accuracy of 99.9%. No long term knowledge base can claim they are this accurate.

7.4.4 Versatility of RM

These results provide a small improvement over previous prudence analysis results, indicating the value of $RM_{p(c)}$ as a potential predictor of the boundary of knowledge in a KB. While this separation between $RM_{p(c)}$ and Compton et al's (1996) results is only small, there was one very significant advantage found in the RM approach during the testing procedure. After the basic parameters have been selected, the final threshold adjusting rate can have a significant and worthwhile affect on the system's results.

This parameter can be set to adjust the threshold quickly or slowly. If it adjusts quickly the system is less accurate but produces fewer warnings. If it adjusts slowly then this significantly improves the accuracy at the cost of producing more warnings. Effectively, this allows an expert direct and simple control over the warning and accuracy of the prudence system. It would be envisioned that in an application this adjustment could be made via a simple slide bar. For example, such a system would allow an expert creating a knowledge base for a nuclear reactor to set the system to provide many warnings but will gain a KBS with virtually 100% accuracy. Alternatively, an expert creating a KB to filter their emails into appropriate folders may reduce the accuracy, thereby, only having to check a few more accurate warnings.

The potential for this versatility was noticed during early testing of the prudence methods and so a number of tests were performed to verify its occurrence. These tests involved setting up ten different values for the threshold

adjustment factor and running the system, each with ten different randomisations. This was done for each of the four datasets. Figure 7-5, shows eight charts, two for each of the datasets. The first column shows a full chart for each dataset with a number of pieces of information. In this column there is a dotted straight line joining the 100% accuracy with the 100% number of no-warnings. This line represents the theoretical case for a random generation of warnings. Secondly, there is one unfilled and nine filled triangles, joined by a smoothed line, representing the different performances of the $RM_{p(c)}$ approach for the different threshold adjusting rates. The unfilled triangle identifies the result that was given in Table 7-1. These charts also include a single square showing the result for $RM_{p(p)}$. Finally, the empty circle represents the location that Compton et al's (1996) system achieved for each dataset[26].

The second column of charts provides a close up of the RM results section of the chart in the first column. The charts in this column do not show the line indicating the theoretical results for random prediction of warnings. Included on these charts instead are two sets of error bars for the square and each of the triangles. The vertical error bars indicate the 95% confidence range for the average accuracy, while the horizontal bars show the same for the percentage of cases not warned.

It can be seen in these results that RM was able to achieve virtually 100% accuracy for all datasets. For example, with the GARVAN Dataset, the system can achieve nearly 100% accuracy (the actual average result was 99.93%) if warnings are provided on just over 50% of cases. This is clearly a lot of warnings but is a vast improvement on the current use of RDR methodologies where the expert must check every case. It can also be seen that the 100% accuracy could only be guaranteed if the system provided a warning on every case, as would be expected. What is important in these results is that you can achieve a result very close to perfect with much fewer warnings. The other aspect of these results is that the number of warnings can be reduced to around 84%, in the GARVAN dataset, of cases if the expert can tolerate an accuracy of just fewer than 90%, which is still very high. Similar results allowing variation can also be seen for the other datasets.

[26] Not included on the multi-class dataset as that is unique to this thesis.

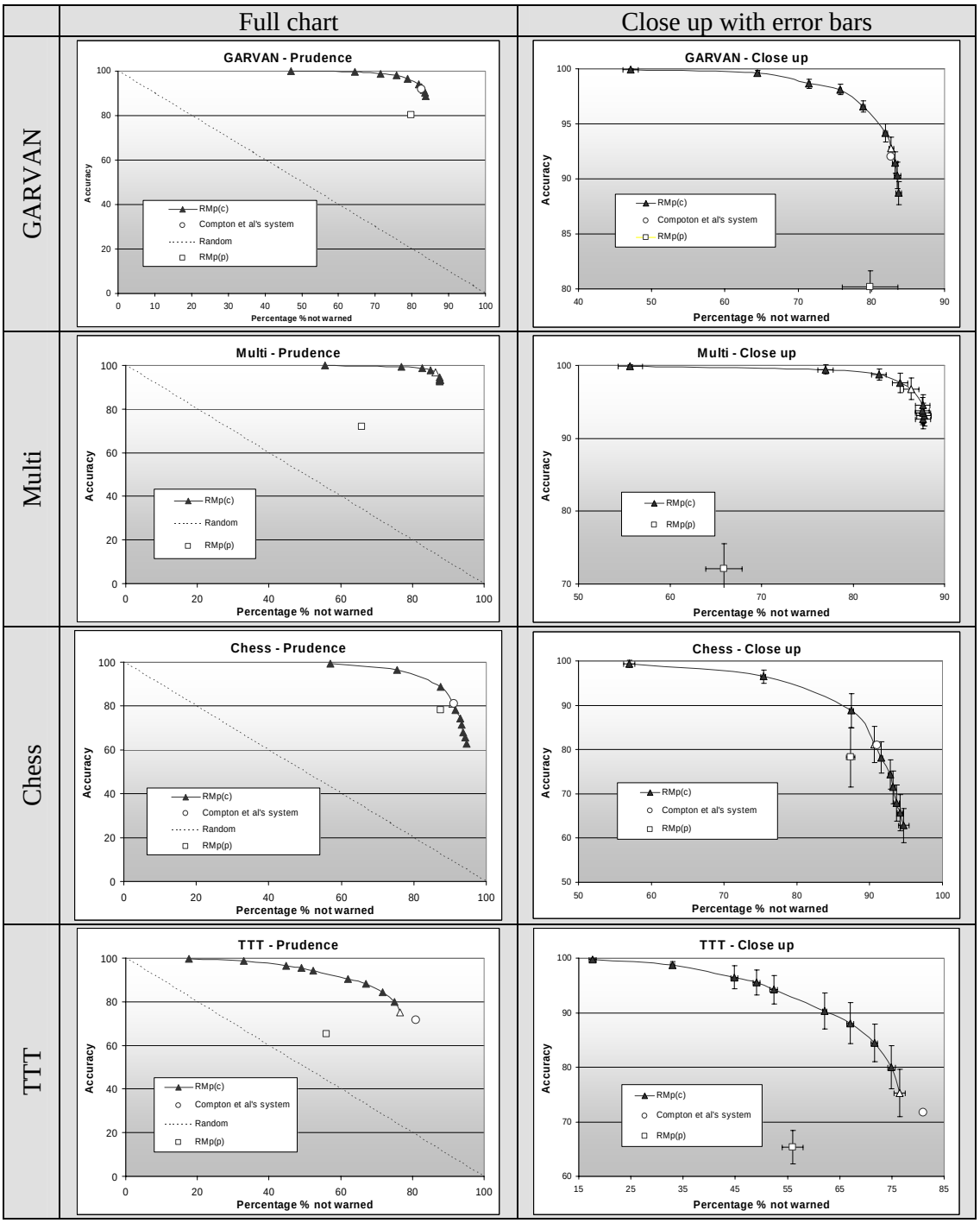


Figure 7-5: Comparison of results from a range of tests using different threshold adjustment values. The y-axis represents the level of accuracy achieved by each experiment. The x-axis shows the percentage of cases where a warning was correctly not given. Each triangle and square represents an average from ten runs. Each also has error bars for both the x and y axis indicating the 95% confidence level. The hollow triangle identifies the result also shown in Table 7-1. The square identifies the result for $RM_{p(p)}$. The hollow circle locates Compton et al's (1996) results. The dotted line represents how a random prediction would be expected to perform.

These results also reveal that there is a natural limit in $RM_{p(c)}$ when trying to reduce the number of warnings. As the threshold adjustment variable approaches infinity the system effectively loses the effect of the threshold. This represents the situation for the lowest level of accuracy and warnings. However, this does not result in zero warnings. On the contrary, the system still produces a number of warnings and is still reasonably accurate. This is because warnings are still being produced when there is a difference between MCRDR and the ANN's classifications. No amount of adjustment to the threshold can stop these warnings. This is why in the TTT dataset the number of warnings could not be matched to Compton et al's. This does not, however, negate the advantages of the versatility just discussed, because this appears to only occur at relatively low levels of accuracy, which is unlikely to be selected by an expert.

These results also show Compton et al's (1996) results sit on or very near the performance line of $RM_{p(c)}$. This shows that, with the appropriate parameter settings for RM, the systems perform similarly well. However, the versatility of $RM_{p(c)}$ provides a very significant advantage over Compton et al's (1996) system, which is incapable of adjusting its performance. The way it performs for a particular dataset is entirely governed by that dataset.

7.4.5 Learning Speed

When this system has been presented at conferences, the question that has commonly arisen is how can the ANN in this method learn so quickly to get such spectacular results? This question originates from a long held belief that ANNs only learn very slowly. The answer for the success in the RM system essentially boils down to two factors. The first is the speed of learning in the fundamental methodology. As described in the previous chapter's results, MCRDR reduces the dimensionality of the problem space making the task of the network easier especially when combined with the *single-step-Δ-update-rule*. The second is a sleight of hand regarding the application of the method to the task. Normally, an ANN is trained for what you are trying to learn, for instance, what cases are incorrect and require a warning. The key to the approach used in this thesis is that the methods are learning what they do *not* want to know. In other words, they are learning when not to warn. This can best be described by the chart in Figure 7-6.

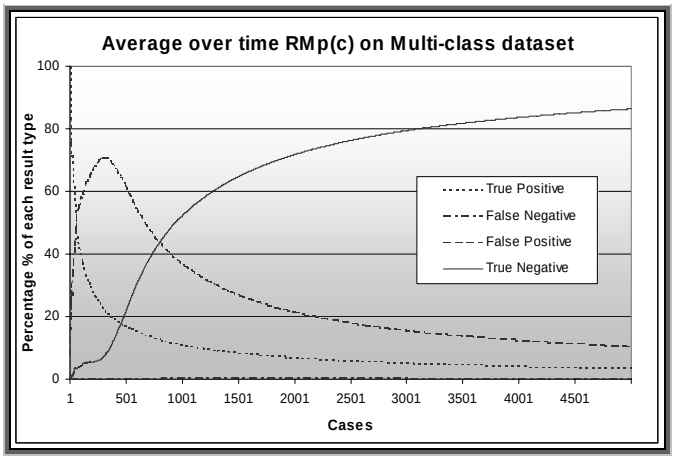


Figure 7-6: Comparison of the average percentage of cases that produced each of the four types of results over ten randomly generated trials. This chart shows the $RM_{p(c)}$ system on the multi-class dataset.

This chart shows the percentage of each of the four possible results for each case using the multi-class dataset with $RM_{p(c)}$. These are an average taken over the ten randomised runs for the same parameter settings used in the results shown in Table 7-1. It can be seen that the *true negative* line follows a fairly typical ANN learning curve and is essentially what the network is trying to learn. Therefore, there is no real trickery in the ANN itself just how it has been applied to the task.

7.4.6 Error Distribution

Two early fears in testing prudence analysis with RM were that there may be either: a peak of *false negatives* during the early cases presented to the system or, an increase in the number of missed warnings towards the end of the expert system development. An early peak could cause many of the major generalising rules to be omitted causing many errors in the final KB. Without the thresholding rule this peak does in fact occur, although it is only small. One reason for introducing the thresholding was to prevent this early peak of *false negatives*. The increase in missed warnings would indicate that the system was steadily deteriorating, which is the opposite effect to what is required and rendering the whole process as unviable. The progressive results displayed in Figure 7-6, however, show that neither of these potential problems occurred. The few *false negatives* that did occur were spread out evenly over the first half of the expert system's development. These results strongly indicate that $RM_{p(c)}$ is certainly viable and worth further testing.

7.5 Using Prudence Warnings

The results in the last section show an advance from Compton's earlier work. They illustrate that prudence analysis is viable and that it may be possible to build a system that can self-assess its knowledge base in relation to each new case presented. In the field of knowledge acquisition this could be a significant improvement. However, the above results do not show how well the system would perform if the user placed their trust in the prudence analysis and only checked cases that had generated a warning. Furthermore, due to no previous prudence systems producing a consistently high enough accuracy, no one has published any studies on the effect of an expert trusting such a system.

It is possible that even a relatively small amount of inaccuracy in a prudence based system could result in large problems in the knowledge base. For instance, when an expert does not correct a rule, because it was not warned about, any of the following could result:

1) The system may simply notice the next time a similar case arrives and warn the expert at this later stage.
2) The missed rule may never be created and the prudence system may not warn about future cases that may have also caused the creation of the rule.
3) The missed rule may have a compounding effect, where it causes rules that would have produced warnings later to now be missed because the knowledge base is damaged.
4) The compounding effect could be exponential resulting in a KB that is woefully inadequate.

Without testing, it is unclear which of these results would occur. The greatest obstacle to the idea of prudence analysis as a viable technique would be the compounding effect described in the last two points. Therefore, the aim of this final result section on prudence analysis will attempt to determine the effect on the final KB when developed by a trusting expert. The first result simply investigates the number of rules created, while the second section compares the trusted KB against the full KB when classifying unseen cases.

7.5.1 Rule Creation Comparison

This first test was performed to investigate how many rules are created in a system where the expert trusts the prudence analysis. In these experiments each dataset was re-run with the same parameters as the earlier prudence studies shown in Figure 7-5. However, on this occasion rules were only corrected if there was a warning generated. Figure 7-7 shows the results gathered in four stacked area charts, one for each dataset. Each area joins the ten different parameter settings, which line up with those in Figure 7-5. For instance, the result with the highest accuracy in Figure 7-5 is placed in the left most position.

The bottom area in each of the charts of Figure 7-7 show the percentage of rules created by the trusting expert against the total number a full expert would have created. In a system with full accuracy these would be the same giving an area of 100%. However, in systems with less accuracy this percentage would be expected to reduce. The second dark-grey stack shows the percentage of rules we know were not warned about from the previous study. If these missed corrections are not fixed but all other rules are, as per the second point above, then the stacked percentage should be approximately 100%. If the missed rules are warned about later, as per the first point, then the percentage of the bottom area will increase resulting in the combined total for the two areas going over 100%. If missed rules cause a compounding effect, where rules correctly warned about in the previous study are now missed because the knowledge base is incomplete, then the combined areas will drop below 100%.

Upon inspection, it can be seen that the results are mixed. Both GARVAN and the Multi dataset maintain a total area above 100% for the majority of parameter settings. This is an extremely promising result and indicates that the only rules missed are those not warned about and that some of these are even found later during the development cycle. However the last two GARVAN results do start to degrade marginally, suggesting that this promising result is only possible when the accuracy is sufficiently high. Comparing these results with those in Figure 7-5, the reduction in performance correlates to parameter setting where the accuracy drops below 90%. The multi-class dataset does not show the same drop on average but the error bars do indicate that this performance was beginning to be affected by the order cases were presented.

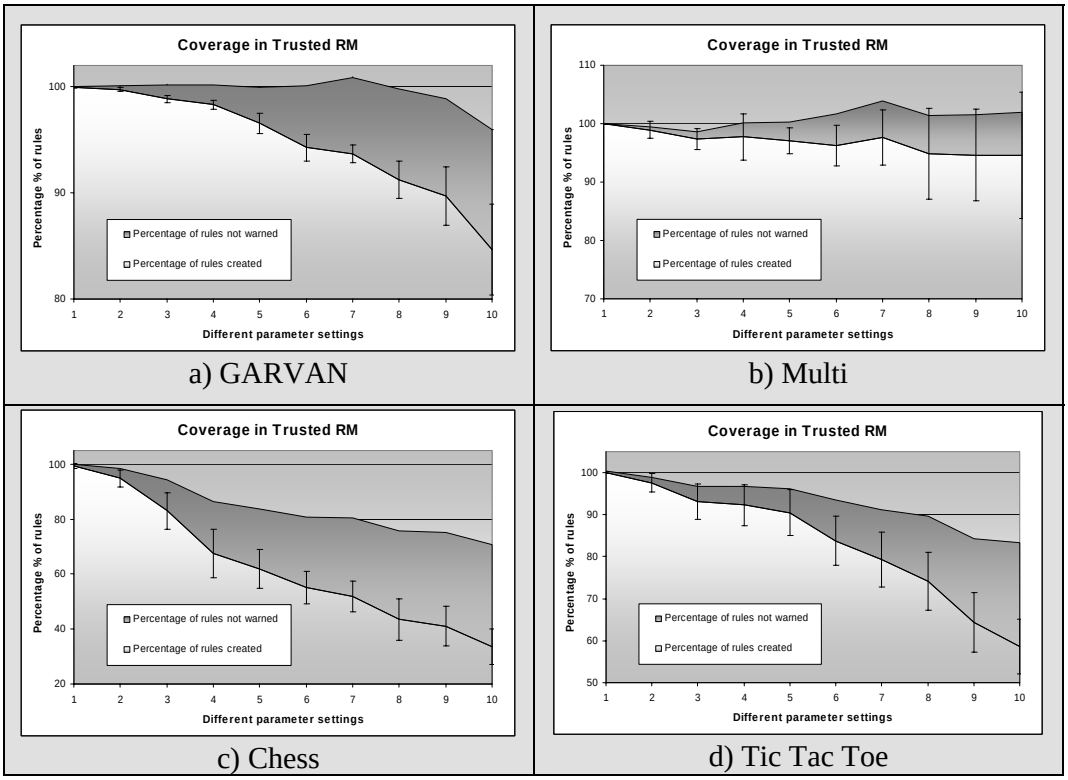


Figure 7-7: Four stacked area charts showing whether there is any compounding effect on a KB when rules are missed. The bottom area shows the percentage of rules created by the trusting expert compared to the total amount the full expert would have created. The top area shows the percentage of missed warnings from the previous prudence experiment. Error bars are shown for each point on the bottom area. They are not shown on the top area as they are too small to be visible.

Both the chess and TTT datasets' results, however, are not as impressive. It can be seen that they drop off relatively quickly. These show that unless near perfect accuracy is achieved then there is a severe compounding effect. Not only did these experiments miss the rules not warned about but the damaged KB caused additional results not to be warned. While this degrading performance does not appear to be exponential, it does appear immediate. Unlike the GARVAN and multi dataset which tolerated a degree of missed rules, the chess and TTT datasets dropped off the moment the degree of accuracy started to reduce. The issue with these datasets is that they are both constructed from rigid environments where the smallest change in the position of a piece can significantly alter the case. Therefore, missed cases are more likely to have an affect on the resulting knowledge base. On reflection these are not ideal datasets for this environment but were used only to compare with the previous work in prudence analysis.

7.5.2 Classification Accuracy

The promising results in the last section show that the only rules not created are those not warned about and that even some of these are still found later on. This is important for the viability of prudence analysis, however, it does not show what effect even these few missed rules have on the classification ability of the resulting knowledge base. To judge the over-all affect on the KB, the generalisation test from chapter 6 has been performed on each dataset with both the full and trusting experts. Figure 7-8 shows four charts showing the 9 classification tests. Each of the nine indicates how many $1/10^{th}$ segments were used for training prior to being tested on the last $1/10^{th}$ segment. The parameters chosen for each experiment are the same ones used for the results shown in Table 7-1. These all had a reasonably low level of accuracy and not many warnings.

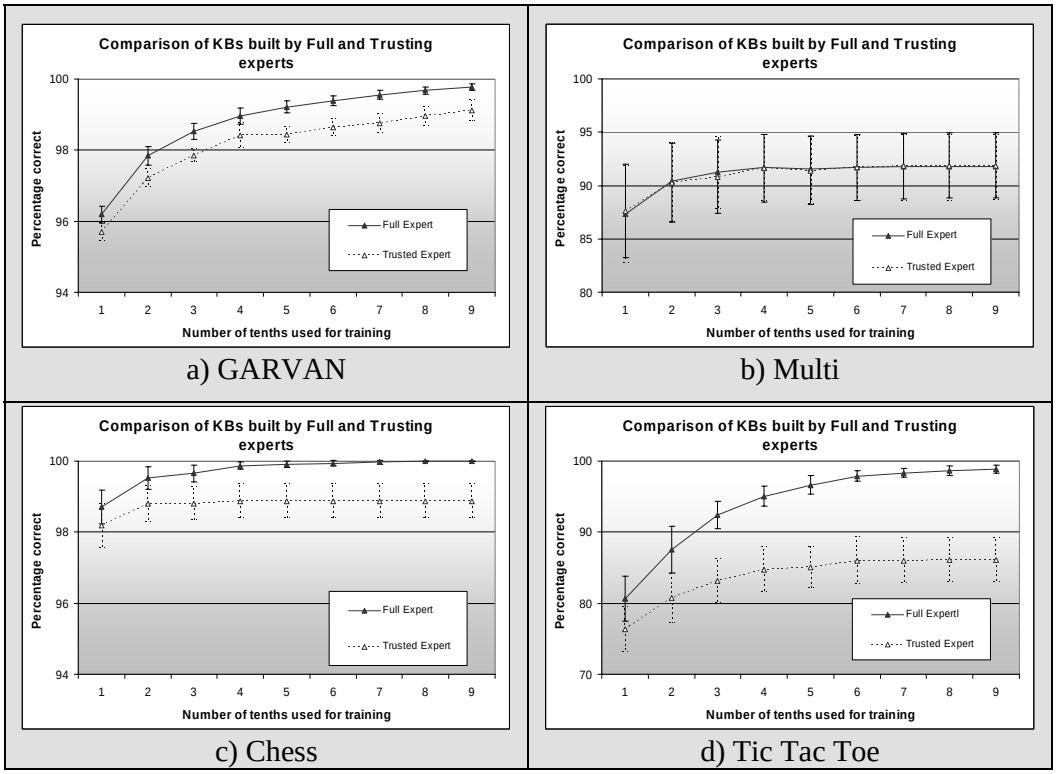


Figure 7-8: Compares the full and trusting experts' knowledge bases after training. Each chart shows the KB's performance for the full system (where the expert checks every case) and the trusted system (where corrections are only made if a warning is generated). Error bars are also shown at a 95% confidence range. The full expert is shown with filled triangles and solid lines while the trusting expert has hollow triangles and dotted lines.

Interestingly, the trusted KBs perform nearly as well across each of the datasets. On the multi-class dataset the KB is almost identical while the GARVAN dataset only degraded by less than 1%. This is remarkable as the KB had lost approximately 7% of its rules because the expert did not receive a warning. Likewise, a similar performance can be seen on the chess and TTT datasets. For instance, the chess dataset only lost a little over 1% in its classification ability even though it had lost over 30% of its rule base and TTT had lost less than 15% of its classification even though it lost more than 40% of its rules.

At first glance this seems remarkable that the KBs could still maintain such a high level accuracy. This result though is most likely related to the general nature of KBSs. For instance, between 1984 and 1987 the GARVAN-ES1 KBS increased its number of rules by approximately 80% but only improved its level of accuracy from 96 to 99.7% accuracy (Compton et al. 1988; 1989)(3.1.1). Therefore, like in the GARVAN-ES1 system, the rules RM fails to warn on appear to be the more specialised rules that only cater for the occasional rule. This cannot be proven in this study but the high degree of accuracy after losing so many rules indicates that the rules remaining are more general in nature.

7.5.3 Non-Perfect Humans

The above results compared trusting the prudence system against an expert that checks every case. This is fine in a simulated environment. However, in a real world knowledge acquisition (KA) task a human expert is rarely in a situation to fully check every case. It could be argued that, in fact the human expert is likely to miss the occasional misclassification in the full system. This is likely to be especially prevalent during the later stages of KA as they become more complacent towards the system's accuracy. It is entirely possible that the human expert could fail to notice more misclassifications than RM_p. For instance, it would not seem unlikely that a human expert could easily miss 5-10% of errors in the KB. The results for RM_p, however, can achieve a much improved level of accuracy.

It could be further argued that a human expert is more likely to pay closer attention to a case when the prudence system produces a warning. Therefore, they are much less likely to miss a misclassification after a warning. Thus, it

may be that in a real world system, the small reduction in performance of the trusted KB would be entirely dissipated or even reversed. Unfortunately, the performance of an expert in the two different environments is extremely difficult to measure, and is not within the scope of this thesis.

7.6 Summary

Prudence analysis represents a method for predicting when a case requires knowledge beyond the system's current KB. It is one way of attempting to resolve the issue of brittleness in current knowledge based systems. In theory, prudence analysis would be a very powerful tool when performing knowledge acquisition and maintenance. Previous work in this area (7.1.3), however, has yielded results that are not of sufficient accuracy, or that produce too many warnings, to make such a system viable. Theoretically, RM studies the internal structure of the KB as it is being developed. Therefore, it was believed that this information could be applied to this domain, by allowing it to learn the contents of the KB, and thus, know when a case requires knowledge from outside.

In this chapter, two methods of prudence analysis were developed utilising RM. The first, $RM_{p(p)}$ (7.2.1), predicted a single value indicating whether the expert should check the case. Generally this method did not perform as well as either the second approach developed or previous research in the area. The second method developed, $RM_{p(c)}$ (7.2.2), used the classification ability of both hybridised components in the RM methodology and compared the results. If they disagreed a warning was produced.

The results presented in this chapter show that $RM_{p(c)}$ is able to predict errors more accurately than previous work (7.4.1) without increasing the number of warnings (7.4.2). The most interesting results were those detailing $RM_{p(c)}$'s versatility (7.4.4). It could be seen that, through a simple process, the expert could control precisely what level of accuracy was required for the task at hand. This versatility makes $RM_{p(c)}$ more useful when applied in a range of applications.

Due to the $RM_{p(c)}$'s excellent performance, it was further tested to see how the KB would be effected if the expert trusted the system and only checked cases that first produced a warning (7.5). Previously, no study had been

undertaken to test the application of a prudence system and it was feared that the missed errors could compound through the KB development, resulting in a significantly flawed KB. Results (7.5.1), however, indicate that compounding of errors only occurs with very low levels of accuracy or in particularly rigid datasets where the smallest change in the case results in a different conclusion. In fact, some evidence was found to suggest that missed errors appear to be noticed and corrected in subsequent cases.

The most impressive result found in this chapter was that, even though a number of rules may not be created because no warning was produced, a high level of classification accuracy could still be expected (7.5.2). For example, the KB built with the GARVAN dataset lost approximately 7% of its total rule base. However, this only caused a loss of less than 1% of the KB's classification ability.

It is further argued (7.5.3) that in many application domains the resulting KB may even be better than one fully checked by an expert. This is due to the expert having less complacency when checking cases warned about by a prudence system than when every case must be verified. This advantage would be most prevalent when warnings are kept to a minimum. Therefore, the $RM_{p(c)}$ system presented in this chapter represents a truly viable solution to prudence analysis which is deserving of further study in a real world domain using human expertise.

8

MonClassifier$_{RM}$: Venturing into the Real World

"You never can tell with bees" said Pooh (A. A. Milne, p11).

The previous chapter introduced the concept of prudence analysis and showed how RM is a highly effective tool at generating accurate warnings, while minimising the total number of spurious warnings. It was illustrated that in the example datasets, RM's use of hidden contextual relationships allowed the system to outperform Compton et al's (1996) previous results. Combined with its versatility, this system supplies a viable and realistic approach to providing a working prudence system for incremental knowledge base development. This was further supported by the results suggesting that the final knowledge base produced comparable classification accuracy to a fully checked knowledge base.

These strong results, however, are all from either fabricated datasets or using artificial expert knowledge, generally in the form of a decision tree. While these simulated experts provide strong indications of how well a method performs in a given situation – they are still only indications. However, these simulated experts do not show if the system will work when using real human knowledge. This is due to human knowledge often differing from the simulated knowledge of systems like C4.5. Therefore, this chapter is aimed at moving the prudence methods developed in the last chapter into a more realistic environment. In this environment the approach used can be tested in a domain using human knowledge. If the prudence system can predict misclassifications with a reasonable degree of accuracy then this will show that the system is viable for use in real world knowledge based products.

This chapter uses an application, referred to as MonClassifier, developed by Sung Sik Park, as its testing environment. MonClassifier is part of a larger suite of applications, called PWIMS (Personalized Web Information Management System), that have been published, in different forms and sometimes using different names, across a number of international conferences (Kim et al. 2004a;

Kim et al. 2004b; Park et al. 2003; 2004a; 2004b). This application has been used to collect and classify a number of knowledge domains. For each of these domains a knowledge base has been incrementally built using human experts.

After detailing the overall PWIMS system, this chapter will describe the MonClassifier application including why it was selected to be used in this study. It will also describe how this tool and the datasets were used when testing RM within this environment. This will include a discussion of various alterations made to the knowledge base and the dataset formats to allow for integration of RM and MonClassifier. This is required as the two systems were not directly compatible in their original states. Finally, this chapter will then describe the experiments performed as well as present and discuss the results gathered.

It is important to note that the MonClassifier application itself is not part of this thesis and that the author of this thesis had no direct input in its development. MonClassifier is presented here purely to give the reader an understanding of the application used. This thesis is only concerned with the philosophy of hidden and dynamic contexts, the development of a method to capture such concepts and showing the developed technique's ability to perform in prudence analysis. MonClassifier is only being used in this thesis, as it provides access to a large real world compatible dataset along with a near complete real world knowledge base developed by a human expert. Therefore, it provides an environment that is as close to a real world test as is possible within the time frame of this project.

8.1 The MonClassifier Application

MonClassifier is a component of a larger integrated knowledge management system for newly uploaded pages on the World Wide Web (WWW). Currently, the larger system which is still in development has no official name, but has been published in a similar form where it was referred to as PWIMS (Personalized Web Information Management System). The difference between the published system and the current system is that it is no longer a stand alone personalised tool, rather, it has been redeveloped using a server-client architecture (Park 2005). For the purposes of this thesis, the larger system will be referred to by the older name PWIMS as it is fundamentally the same system.

8.1.1 PWIMS

The PWIMS system, and the current version of the system, consists of three components:

- a *Web monitoring agent*
- a *storage management* (or *knowledge management*) component, and
- a *knowledge sharing agent.*

PWIMS provides support for dynamic and personal web portals in a simple suite of applications. The Web monitoring agent monitors a number of user-specified websites for newly uploaded pages. When a new page is uploaded the system retrieves the page and stores it in the database. Therefore, this system is ideal for monitoring sites such as news or research portals. When pages are gathered the storage management component is used by a domain expert to classify each article appropriately. The system can then redistribute the classified articles to a web page or push relevant pages to clients. Figure 8-1 shows a diagram showing the PWIMS architecture. The Web monitoring and knowledge sharing agents, however, are not relevant to this thesis and are not discussed further.

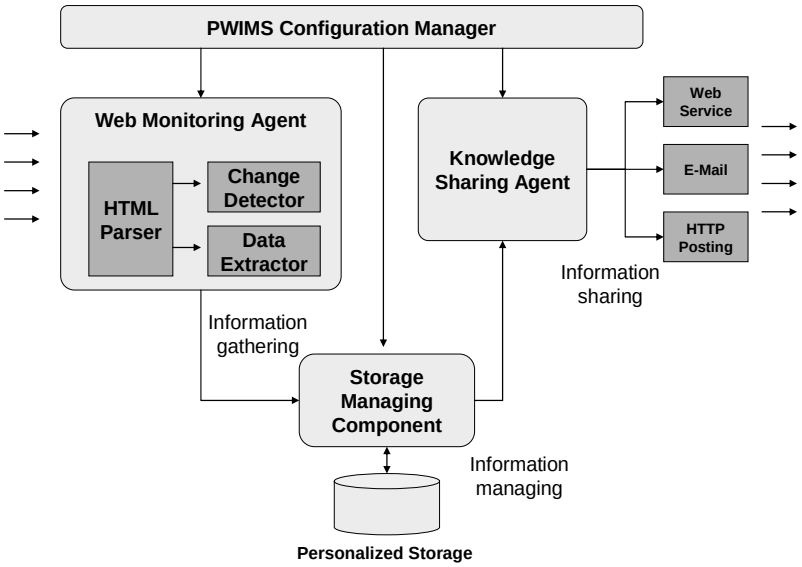


Figure 8-1: PWIMS Architecture (Park et al. 2003)

8.1.2 MonClassifier Overview

The only component of PWIMS that is directly relevant to this thesis is the central storage management component. In PWIMS this was just a component of the larger system, however, in the currently unpublished version this component is the stand alone client application referred to as MonClassifier (Park 2005). This sophisticated application takes each gathered web page (case) and presents it to the expert who classifies each in turn. Generally, this simply involves accepting the offered classification, but occasionally may require a reclassification.

The classification engine used is MCRDR, hence the interest in the system to this project. Therefore, the human expert can classify each web page into any number of possible classes. These classes are actually folders organised in a tree like hierarchy in a similar fashion to Windows Explorer. The expert actually only really needs to understand the folder organisation and does not require any understanding of the underlying knowledge base. A screen shot of the main view from the MonClassifier application is shown in Figure 8-2.

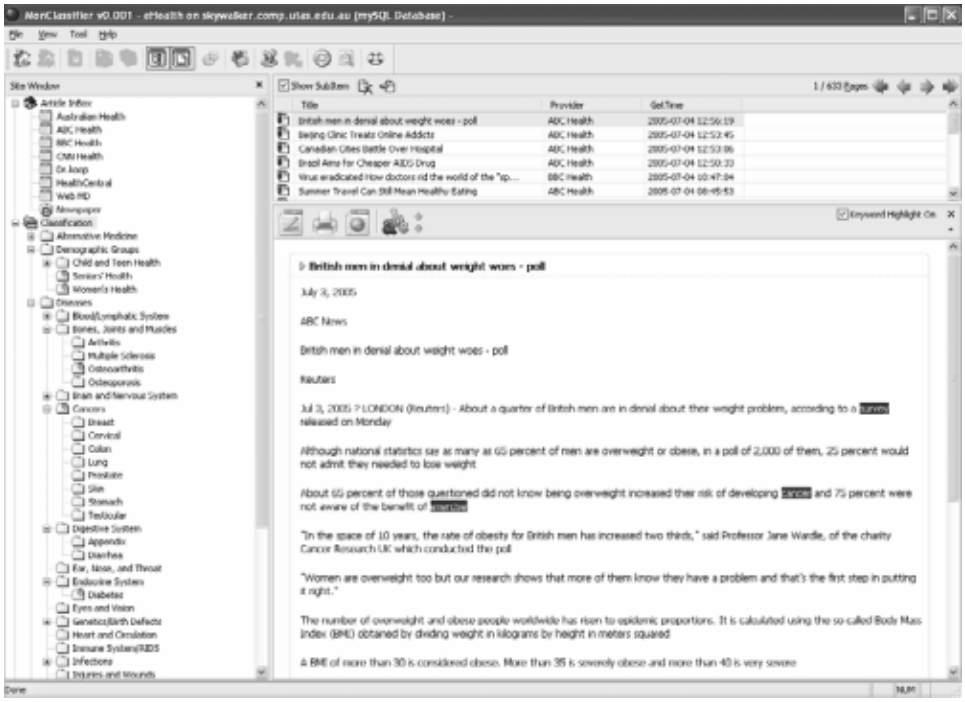


Figure 8-2: Main view of the MonClassifier application. It shows the monitored sites and classification tree on the left with block arrows indicating the current article's classifications. The current article has been parsed with only the relavant details shown in the main window. The keywords, used for the current classifications, are highlighted. The top splitter window shows the current list of classified articles.

In situations where the expert feels a case has been misclassified they can correct it via the KA interface provided. A new class can be created by the expert by simply creating a new folder. The expert can then create rules by simply selecting words from a difference list. It can be seen from the previous screen shot that each case consists of many possible keywords. Therefore, the MCRDR engine allows the expert to select the words that they believe adequately separate the cases within the current context.

The MonClassifier application stores each article gathered in a MySQL database. Additionally, the database stores all relevant information required about the knowledge stored and the relevant cornerstone cases. It is in fact this database that RM directly interfaces with in order to generate the results in this chapter.

8.1.3 Weakness of MonClassifier

The MonClassifier system is an extremely sophisticated application and one of the most elaborate implementations of MCRDR in a real world system. It's an application designed to be used by non-technically minded users. However, like in any RDR based application it fails significantly in one regard. The work load required by the expert to build a knowledge base is far too high. The dataset of gathered articles used in this chapter has taken a number of months (part-time) for the expert to build the knowledge base. The process has involved the expert reading each article and deciding whether the conclusion was correct. Clearly, this is excessively time consuming, especially during later stages of KB development, as so few cases actually need correcting.

While the MonClassifier has an important market in its current form it could be significantly improved by the inclusion of a viable prudence system. A prudence system would allow an expert to reduce the amount of cases that require reviewing while still being able to maintain a reasonable level of confidence in the accuracy of the knowledge base. It is the viability of RM in this role that this chapter will investigate.

8.1.4 MonClassifier Knowledge Representation

RM was not actually directly interfaced with the MonClassifier system. Instead, it remains a separate application and testing environment that directly interacts with the MonClassifier's database and knowledge base. While the MonClassifier used MCRDR for its knowledge base it was not implemented exactly as stipulated in Kang's (1996) PhD thesis. The altered form of the MonClassifier's MCRDR structure did cause some integration issues with the RM system, which are detailed in this section.

MCRDR rules were designed to incorporate any number of conditions linked with a Boolean *'AND'* operator. However, MonClassifier implemented a simplified design where each rule contained only one condition. In situations where more than one condition was selected by the expert, a series of single child branches were added to the tree where only the final rule contained the actual classification. The different representations can be seen in Figure 8-3. This alternative representation was not a major problem, although care was required when creating rules, which is detailed in section 8.2.2.

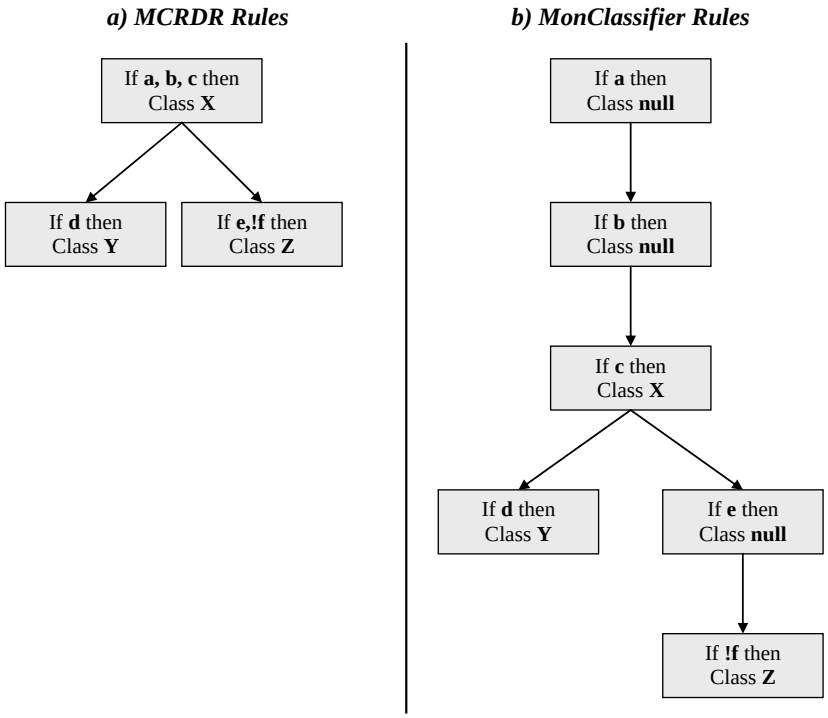


Figure 8-3: Comparison of the rule structures used in traditional MCRDR and that used in MonClassifier. a) Shows how the MCRDR KR compactly incorporates the conditions. b) Illustrates the more spread out KR used in the MonClassifier application. Rules without an associated class simply have a null classification.

8.2 Experimental Method

The testing methodology used to generate the results in this chapter is essentially the same as the one used in the previous chapter. However, using RM in this environment did present a number of difficulties. While most of these were only implementation difficulties, some deserve mentioning in this section. The expert knowledge available had to be used indirectly, rather than producing warnings for the user directly. Other difficulties revolved around the memory requirements of a large dataset as well as the use of unclassified cases. This section will first provide an overview of the tests performed, the human-expert knowledge, and the dataset used.

8.2.1 Tests Overview

The tests performed for the results in this chapter are the same as the previous chapter. Essentially, two types of tests were performed using both of the prudence analysis implementations, $RM_{p(p)}$ and $RM_{p(c)}$. The first test simply tested each method's ability to accurately predict when to warn the user of a potential misclassification. This involved recording true positives, false positives, false negatives and true negatives for each case presented. The second test investigated what effect using the $RM_{p(c)}$ prudence system would have on the accumulated creation of rules when the expert trusts the system. The last chapter also included a generalisation test on both the full and trusted KB. However, this experiment could not be duplicated for the MonClassifier's dataset due to the difficulties in processing large collections of full text cases in a limited time frame.

8.2.2 Expertise

The ideal test would have been to have RM fully integrated with the MonClassifier system and see how the human expert actually performs in this prudence based environment. The difficulty here is that the time frame of this project did not permit for this model of testing, nor was it possible to have an independent expert provide months of their time to produce a single test. Instead, the project used the knowledge base already built by a human expert in a similar way to the C4.5 simulated experts in the previous chapters. However, instead of

first generating a decision tree through machine learning, the human expert's MCRDR tree was used. This involved creating a new semi-simulated expert. The reference to this new expert being only semi-simulated identifies that, while the expert behaviour is software driven, the actual KB used is not simulated.

Regardless of the similarities to the previous chapter's simulated expert, this semi-simulated expert is fundamentally different to its brother in many respects. Firstly, the knowledge base is full of all the usual inconsistencies, errors, intuitions and foibles that would be expected from a human built system. These 'errors' from a human expert do not occur in the machine generated knowledge base. Secondly, the human built tree was built from a multiple classification domain, whereas, the C4.5 based simulated expert could only produce single classifications.

The semi-simulated expert first loads the knowledge tree from the MonClassifier's database, which is kept in its original form, as described in section 8.2. This tree then views each case presented and classifies it, just as in the C4.5 expert. However, occasionally a classification can result in a '*null*' class. In these situations the last '*non null*' class is used as the classification. This is the same method of inferencing used in the MonClassifier application.

If the semi-simulated expert's conclusions differ from those of RM then new knowledge is added. When selecting attributes for the conditions of any new rule the same process was taken as in the previous C4.5 based simulated expert. Starting from the root, attributes are selected by moving down the knowledge tree until a certain number are identified as not being in the RM rule path of the node currently being corrected. The main difference here is that there is no arbitrary number of attributes required. The expert will select all the attributes needed to reach the relevant *non null* node.

The resulting semi-simulated expert creates rules in a fashion that is very simular to the real world human expert and does so with the human knowledge previously recorded. However, this approach is not using the actual human expert for interaction. Thus, it is not a true representation of a real world test. Nevertheless, it is as close as possible, in any realistic sense, to a real world environment. Additionally, it offers advantages that direct interaction with a human would not offer, for instance the ability to perform the tests multiple times with different sequences of cases.

8.2.3 Dataset

The monitoring agent in PWIMS has been gathering articles for an extended period of time within a number of various domains. The domain with the largest collection of articles is the eHealth domain which contains articles gathered from a number of health related news sites, such as Australian Health, BBC Health and CNN Health. At the time of testing this dataset contained 17571 full text articles. More importantly though, was that the expert had classified 12645 of these cases into the KB. In classifying these articles the expert had created some 120 classes. This presents a very substantial real world dataset.

There were two main difficulties, however, with the dataset as it was supplied by the MonClassifier application. The first was the 4926 cases not yet classified. The vast majority of these could simply be correctly classified immediately using the existing knowledge in the KB. The only reason they are in the pool of unclassified articles was because the expert has not checked them yet. However, some may need new rules to be added while others may have been seen by the expert but rejected as not being relevant to the domain. The problem for the experiments in this thesis was what to do with these cases. If they were simply omitted, the tests would not be checking whether RM can recognise default cases that do not need a classification.

The decision made was to treat these cases exactly the same as the classified ones. This meant making the assumption that the knowledge base was complete and that the classifications that would be given to the, as yet unclassified, cases were correct. The benefit of this decision meant that all types of cases even those rejected as irrelevant or requiring a default classification could be included.

The second issue was purely an implementation issue. Each case was very large and access times to the MonClassifier database were far too slow to be done on an individual basis. Therefore, large chunks of the dataset were loaded during each query. This reduced the database access time but slowed testing down due to memory issues. In order to reduce these memory issues each case had irrelevant words removed. Therefore, each case was reduced down to only those keywords that appear somewhere in the KB created by the expert. This had no effect on the behaviour of RM as the words removed would not have been used anyway, but it did speed up the expert's attribute selection.

8.3 Results

To measure RM's performance in a real world domain, the same approach was taken as in the previous chapter. This involved viewing the differences in both accuracy and the number of warnings. Table 8-1 shows the raw results of each system's performance on the eHealth dataset. Included in this table is the calculated level of accuracy in the final column.

Algorithms	False Neg %	True Pos %	False Pos %	True Neg %	Accuracy %
$RM_{p(p)}$	0.12	1.6	22	76	93
$RM_{p(c)}$	0.058	1.7	17	81	97

Table 8-1: Comparison of the raw averages and the calculated accuracy between the two prudence analysis systems developed on eHealth. These results have been rounded to two significant places.

These results are very interesting when viewed in correlation with those from the previous chapter. Both methods, but especially the classification based approach, have been able to maintain a very high level of accuracy, far beyond previous research. $RM_{p(c)}$ has performed exceptionally well outperforming its earlier performance in the last chapter. Likewise, $RM_{p(p)}$ has also achieved a higher than expected accuracy. This high level of accuracy is essential to maintain a KB that is as similar as possible to what would be created if every case is checked.

The main issue revealed in these results is the percentage of warnings generated. When using the prediction based approach there was a warning produced on a quarter of cases. This is because of the increased size and complexity of the system. As previously discussed the algorithm is actually learning when not to warn. Therefore, $RM_{p(p)}$ has failed to learn well enough for a prudence system. When an expert used a system producing excessive warnings it is likely they would cease paying full attention to the warnings produced.

$RM_{p(c)}$, on the other hand, has produced significantly better results. It was able to maintain a higher level of accuracy while also being able to limit the number of warnings produced. However, like the first approach, it has also increased its percentage of warnings from the earlier test sets. Interestingly, it suffered much less from the more complicated domain. This is most likely due to it being able to produce more information from the network's outputs, thereby, preventing a single node from having to provide all the analysis required.

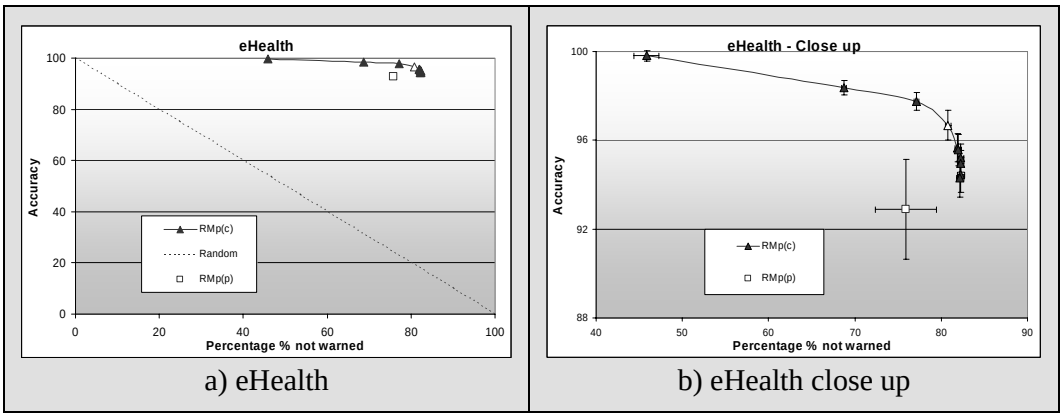
8.3.1 Versatility in eHealth

As discussed in section 7.4.4, one major advantage of $RM_{p(c)}$[27] when applied to prudence analysis is its versatility. It was found that the threshold adjustment factor could be modified, which had very interesting affects on the final results. By altering the threshold at different speeds and maximising the threshold at different levels, a significant degree of control could be achieved. For instance, it was found that the accuracy could often be improved to around 100% at a cost to the number of warnings generated. Likewise, the number of warnings could be significantly reduced at a cost in accuracy.

The results for $RM_{p(c)}$, shown in Table 8-1, were selected as it was judged the best all round performer. However, just as in chapter 7, it was found that a range of performances were possible. An identical experiment to the one performed in 7.4.4 was performed for the eHealth domain. This allowed a determination to be made as to whether the versatility noticed previously is the same for the human knowledge based domain. Figure 8-4 displays two charts showing a selection of experiments with different threshold parameter settings. Each triangle represents the average for a particular parameter setting over 10 runs. Each triangle also has error bars in both the x and y planes, indicating the 95% confidence range. As before the theoretical random system is shown as a dotted line.

The interesting aspect of these results, as discussed in 7.4.4, is that by simply adjusting a single parameter, the system can control the warning frequency and accuracy of the system. Therefore, in a critical system, such as a nuclear reactor, the parameters can be set to give approximately 100% accuracy, while still reducing the amount of warnings from a fully checked system by nearly 50%. Likewise, in minimum risk applications, it may be better to set the parameters to produce as few warnings as possible. In this situation the expert could still expect to get a high degree of accuracy.

[27] This versatility is also possible using $RM_{p(p)}$ as well, however this was not tested as its general performance was significantly worse than $RM_{p(c)}$.



a) eHealth b) eHealth close up

Figure 8-4: Comparison of results from a range of tests using different parameter settings. The y-axis represents the level of accuracy achieved by each experiment. The x-axis shows the percentage of warnings. Each triangle represents a single run.

8.3.2 Parameter Tuning

While this feature makes RM a very powerful tool as it allows the system to be set up specifically for a particular user's needs, it also represents its most significant failing. The sheer number of parameters and possible variations does make it difficult to ensure that the best parameters are shown. These parameters and possible variations also require time and effort to set up the system for each domain applied. This has been a common issue in ANN research. However, this issue was exasperated in RM, because not only were there the ANN parameters but also parameters related to the integration of the ANN with the MCRDR engine.

This problem of parameter tuning has been consistently a major issue in gathering results throughout this project. For instance, the many results generated throughout this thesis were all hampered by the difficulty of parameter tuning. When being applied in a real world application, significant testing of the system would be required to determine a reasonable setting. However, it was also found that with experience many of the parameters were becoming easier to set as they tended to following various patterns. Therefore, while parameter tuning is a major development issue, it alone does not prevent RM's application.

8.4 Using Prudence Warnings

The previous chapter showed that in situations where cases did not generate a prudence warning when the expert would have created a rule produced no compounding effect, unless the accuracy of the system was very low. This is important, because a compounding error would mean a prudence system would require virtually 100% accuracy, thereby, rendering prudence analysis as effectively useless. For completeness, this test has also been applied to the eHealth domain, in order to confirm that errors do not perform any form of compounding issues in a real world domain.

This test was performed by only showing cases to the expert in situations where the RM agent had produced a warning. Figure 8-5, details the average percentage of rules created compared to what should have been created and the amount of missed warnings. It can be seen that the average total of these regularly passes 100%, indicating that there was no compounding effect caused when rules were missed. These results strongly indicate that $RM_{p(c)}$ is viable in such a domain. Not only does it maintain a high accuracy, with a manageable number of warnings, but it also suffers no error compounding. Additionally, the error bars in Figure 8-5 suggest that $RM_{p(c)}$ is very consistent regardless of the order cases are delivered to the system for analysis.

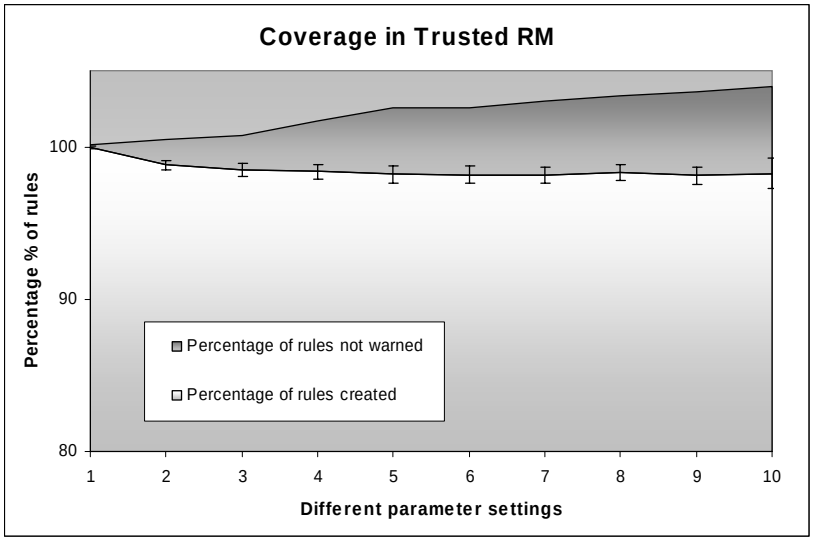


Figure 8-5: Stacked area chart showing the average number of rules created when trusting the prudence system for the eHealth domain. The number of false negatives are taken from the full implementation. The total rules and missed warnings is also provided

8.5 Summary

This chapter built upon the findings in the previous chapter by applying the developed prudence methods into an environment using human knowledge. The environment used was developed from the cases collected and knowledge acquired in the MonClassifier application, developed by Sung Sik Park (8.1). MonClassifier is an MCRDR based classifier for newly uploaded web pages. The MonClassifier program is one part of a larger suite of applications, referred to as PWIMS (Personalized Web Information Management System) (8.1.1). This chapter briefly described this application and detailed the few issues that arose when attempting to integrate it with the RM methodology developed in this thesis.

The purpose of this chapter was to apply RM in as close to a real world environment as possible. The advantage of using the MonClassifier system was that it allowed the use of a large dataset of world health news stories collected over more than a year, combined with the knowledge already extracted from a human expert. This was important as the knowledge contains all the usual inconsistencies, errors, intuitions, and foibles that would be expected from a human expert. Furthermore, this knowledge uses MCRDR and so is in a compatible format.

The results discussed during this chapter certainly support the general trend observed in the simulated environments. Both algorithms showed a strong ability at being able to accurately predict when a case was beyond the current system's knowledge to accurately classify (8.3). This is of great importance as it supports chapter 7's findings that RM applied to prudence is a powerful tool. It was also found that the classification based approach showed a significantly stronger performance in both accuracy and avoiding false warnings.

The results from the previous chapter, as well as, those presented here, highlight the ability of RM in one major domain of study. Here RM has out-performed earlier approaches to prudence analysis. This strongly indicates that the hidden and dynamic context discovered in the RM system provide valuable information that can very effectively be applied in a highly useful way, thereby, supporting the second hypothesis of this thesis that RM produces significant and useful information.

9

Conclusion

> *'It's a Missage[sic],' he said to himself, 'that's what it is. And that letter is a "P," and so is that, and so is that, and "P" means "Pooh", so it's a very important Missage[sic] to me…'*
>
> *(Milne 1926, p123).*

Gaines's (2000) conceptual framework for human psychology, sociology, action and knowledge, identified the *process of practice* as a central core region that, as yet, remains virtually untouched by researchers in the AI community. Connectionists have been attempting to emulate higher brain functions while knowledge based approaches like Cyc have attempted to extend the application of reason downward through codifying general knowledge. However, connectionist based approaches suffer from their slow computational nature and inability to consistently reason and justify their conclusion. While knowledge based approaches are unable to avoid their inherit brittleness, partly from a lack of general knowledge, but also through an inability to adapt and generalise stored facts.

The dichotomy between connectionist and knowledge based methodologies is not simply limited to their performance in various domains, but also stretches back to some of the underlying philosophies. The knowledge based approaches have traditionally been based on the physical symbol hypothesis which is built around the idea that knowledge is a substance that exists. Furthermore, this knowledge can be imparted from one person to another, which must involve the transference of this substance. Logically, if this material can be passed around, then it can be codified and passed to a computer. Yet, numerous failed systems have forced some researchers to revise these concepts of knowledge and move towards a situation cognition *(SC)* based view. The SC view revolves around the premise that knowledge is generated at the time of its use. This implies that the existence of knowledge is based on the context of a given situation.

However, the few methodologies specifically attempting to incorporate the context of a situation, either in the knowledge itself or in the structure knowledge is represented, only treat context as a static entity. Strong situation cognition advocates claim that this is inadequate and that context has a much more significant influence over knowledge. They go on to claim that any form of symbolic representation should be completely discarded as systems should be purely reactive. This is the fundamental philosophical nature of most connectionist based methodologies. Nevertheless, a purely reactive system is unable to plan, reflect or apply any other form of reason with any great significance. Additionally, strong SC advocates fail to recognise that symbolic approaches have been, and continue to be, successful in a many applications.

The position of this thesis was to find a middle road between these dichotomous streams of philosophical and methodological research. Philosophically, this meant finding a position that potentially satisfies both streams of situation cognition. This was achieved by defining an intermediate situation cognition view of knowledge, by taking a position that is essentially similar to that of the strong SC advocates, with one primary exception – it does not disregarding symbolic reasoning. Basically, this new philosophical position agrees with the strong SC view that context is more complex than weak SC advocates commonly assume. However, if the symbolic approach can incorporate hidden and dynamic forms of context, then they can achieve a similar level of reactivity as is present in non-symbolic approaches.

Methodologically, this thesis intended to develop an approach that took this intermediate SC position, by including the incorporation of hidden and dynamic contextual information within a system that was symbolic at its core. Although not tested directly, such a method moves much closer toward being able to model the *process of practice*. The remainder of this chapter will briefly summarise the methodology and the results achieved in classification and prediction tasks. Additionally, it will review the methodology's application to the domain of prudence analysis. Secondly, this chapter will discuss some aspects of this work not directly addressed in the core of the thesis, as well as identify some of the problems with the methodology. Finally, this document is only a first step in the possible uses of RM. Therefore, the last section details a number of avenues that could be further investigated.

9.1 Methodology and Results Summary

This project involved the development of a hybrid methodology, referred to as Rated MCRDR (RM), combining MCRDR (Multiple Classification Ripple-Down Rules) with a function fitting technique, namely an artificial neural network (ANN). This hybridisation was performed in such a way that the function fitting algorithm learns patterns of conclusions found during the inferencing process. The position of rules and conclusions in the MCRDR structure represents the context of the knowledge, while the network adjusts its function over time as a means of capturing hidden relationships. It is these relationships that represent the methodology's hidden contexts, while the ability to alter these contexts during the system's life allows a form of dynamic contextual adjustment.

This amalgamation appears simplistic but was by no means trivial. The fundamental difficulty was finding a means for taking the inferenced results from MCRDR and coding an input sequence for the network. Basically, the problem comes from the fact that the MCRDR structure is expanding constantly. Therefore, the network's input space must also grow to match. However, previous work in the function-fitting literature has not attempted to develop a network capable of increasing its input space. The problem arises from the internal structure of neural networks where, as the input space is altered, the interconnections between neurons and the associated weights are also changed.

During this project a number of different approaches to solving the growth of the input space were investigated. Included in chapter 5 is a description of the main seven methods developed. Attempts to use other network forms or different types of function fitting algorithms were unsuccessful due to their lack of extendibility. These seven methods were not all developed to be the best - some were used in testing mainly to show that aspects of the more complex approaches did offer some learning assistance.

The initial results chapter (6) first involved performing a number of experiments on the seven techniques. These tests were aimed at finding which method learned the fastest and maintained the best generalisation in both classification and prediction tasks. These tests all used simulated expertise, allowing for a greater range of testing that could never be performed using real

human expertise. The C4.5 simulated expert was developed in the same manner as those used by previous researchers (Compton 2000; Kang 1996). The other simulated experts were specifically designed to test particular features of the system, such as multiple classification and non-linear knowledge domains.

The experiments investigating the various methods' abilities at classification involved both testing their ability to generalise and to learn in an online environment. The generalisation test involved seeing varying sized portions of a dataset for training. A separate section, not seen during training, was then used for testing. Each system's ability to correctly classify these previously unseen cases provided a metric for measuring which method was most capable at generalisation. The online experiments involved showing the entire dataset to the system in small amounts repeatedly. A system capable of learning online should be able to correctly classify a large percentage of cases after only a few viewings. Similar experiments were also performed for the prediction domain. Success, however, was measured by the average amount of error in the system's conclusion rather than the amount of correct classifications. Likewise, the online test for predictions maintained a running average error over time to judge the system's accuracy.

Section 6.2 of the thesis was extensive and resulted in the conclusion that the $RM_{bp(\Delta)}$, approach was the most versatile. It also found that the linear version, $RM_{l(\Delta)}$, also performed just as well in many situations. However, it rarely outperformed the non-linear approach. One interesting result was the failure of the RBF based techniques which did not live up to the expectation of being the most effective implementation. The RBF result was put down to a problem with over fitting. This over fitting was most likely due to the MCRDR component having reduced the dimensionality of the problem space. This has reduced the difficulty for the network and caused the RBF approach to create too many hidden nodes. Likewise the speed of learning in the backpropagation based approaches was improved by the reduction in dimensionality and then further improved with the use of the single-step-Δ-initialisation-rule.

The second collection of experiments in chapter 6, tested $RM_{bp(\Delta)}$ against each of the methods used in its construction, MCRDR and a backpropagation neural network. RM was not tested against other classifiers or predictors as this project was only interested in whether the hybridised approach outperforms each

of its parts. The tests performed in this section are identical to the generalisation and online tests performed in the first section. When doing these tests the best effort was made to set the neural network up in the most optimal fashion possible. Therefore, a number of tests were run in an attempt to find the best parameter settings for the tasks required.

The results in section 6.3 showed that RM was able to capture the power of both techniques, achieving both excellent generalisation and learning fast enough for an online environment. These results also indicated that when the knowledge structure is unable to capture some aspects of a domain that the network was able to compensate, producing significantly better results. The results in chapter 6 show that across all datasets RM is a very powerful methodology and generally outperformed both of its underlying algorithms. This effectively answers the first hypothesis positively.

While part 2 was looking at developing and illustrating the performance of RM in general, part 3 investigated its performance in a single specific domain, prudence analysis. Prudence analysis is a specialised form of verification and validation and is concerned with attempting to predict when a situation is outside its KB. In situations where the system believes it requires knowledge from outside its current area of coverage then a warning is issued, requesting that the expert investigates the situation.

Two approaches were developed to attempt to predict misclassifications using RM. One method was based on the prediction technique (7.2.1), referred to as $RM_{p(p)}$, and the other used the classification approach (7.2.2), called $RM_{p(c)}$. Both of these methods were tested on four datasets and compared to Compton et al's (1996) results. These tests once again relied on simulated expertise, allowing for a more extensive evaluation. In the experiments (7.3) the prudence systems were shown each case in turn. They first classified and then produced a warning if the system felt it was not sure about the conclusion it had determined. Then, ignoring if a warning was given or not, the expert corrected any misclassifications. A log recorded when the system proved these warnings and whether the expert corrected any errors. The analysis of these results could then determine the accuracy of the warnings provided.

Both methods were effective on the majority of datasets, although the prediction based technique failed in the more complex multiple classification domains (7.4). Most impressive, though, was the exceptional performance of the classification based approach. This method outperformed Compton et al's (1996) approach in both the level of accuracy (7.4.1) and the prevention of too many false warnings (7.4.2). Additionally, it achieved these results across all datasets. The method represents the first truly viable approach to prudence analysis yet found.

During the testing phase a surprising result also occurred (7.4.4). It became apparent that by simply adjusting a single parameter within the classification based approach it was possible to have direct control over the prudence system's accuracy. You could increase the accuracy at the cost of producing more warnings or reduce the amount of warnings by reducing the accuracy. This provides RM with a unique and very useful ability. It allows the expert direct control over their knowledge base development. Previous research, such as Compton et al's (1996) work, had relied entirely on a case's attributes to judge if a warning was required. Therefore, warnings were produced by the dataset and the order cases were presented. Therefore, the expert had no ability to control the warning frequency or accuracy.

Due to the system's success in this domain, it was further tested to ensure that a KB constructed using prudence analysis would still result in being viable (7.5). This involved re-running the previous experiments with one fundamental difference. This time the expert would only correct misclassifications if the warning system had first produced a warning. This test emulates an expert that fully trusts the prudence application. In the few studies on prudence analysis done previously no attempt has been made to measure what the effect of missing errors has on a knowledge base. The fear was that a missed classification could compound, causing many other misclassifications to then be missed. These tests showed, however, that this was generally not the case. There appeared to be no compounding of errors unless the accuracy of the system was very low. In fact there was an indication that missed classifications were often detected later.

The final experiment performed in section 7.5.2 tested to see what effect on a KB's classification ability was incurred by a trusting expert. This involved performing the generalisation experiments from chapter 6 again but this time

only creating a new rule if a case was warned about by the expert. These experiments produced some spectacular results. It was found that even when a large amount of misclassifications are missed, due to no warning being produced, only a relatively small effect was produced on the classification accuracy of the resulting KB. For instance, a KB built for the GARVAN dataset only lost 1% of accuracy even though it had 7% less rules.

These result show that even the small amount of missed warnings in the $RM_{p(c)}$ system have very little effect on the KB illustrating the effectiveness of this approach. RM provides a distinct advantage over other knowledge acquisition tools because the expert can confidently leave all cases that do not generate warnings and know they will still achieve a solid KB. These results illustrate how the hidden contextual information found can be interpreted to provide significant and useful information, effectively answering the second hypothesis.

The advantage of using simulated expertise in these earlier experiments was that it allowed for many tests to be performed, which would otherwise be impossible. However, simulated results using artificially generated knowledge only provides an indication of a system's viability. Ideally, a new KBS such as this should be applied in a real world domain using human knowledge. This however, is far beyond the scope of this thesis. It would not have been possible to find an expert willing to devote their time and energy to develop two expert systems: one using prudence analysis and one not. Additionally, even if an expert was available, the test would be invalid, because the expert would gain experience during the development of the first system, which they could use in the development of the second.

Instead, in chapter 8, this thesis attempted to test the prudence analysis system in as close to a real world environment as possible. This involved using a semi-simulated expert. This expert used a simulated method of interacting with the RM system, similar to that used in the C4.5 simulated expert, but using human rather than artificially generated knowledge as a basis for the expert. This means the knowledge was consistent and repeatable for all experiments but also still contained all the potential foibles that are characteristics of human knowledge. This represented a significantly more realistic series of experiments to judge the feasibility of RM.

This final chapter of results used the MonClassifier program developed by Sung Sik Park. This application is used for gathering, classifying and distributing newly uploaded pages from a selection of websites. This was the perfect real world environment to test RM's prudence abilities. MonClassifier used the same knowledge acquisition and inferencing techniques as RM and the eHealth domain had already gathered a substantial dataset. Most importantly, though, was that it already had a large store of human expertise for that dataset.

The results from the experiments carried out on RM's analysis of the eHealth domain showed that its performance was very similar to the simulated tests. The accuracy for $RM_{p(c)}$ in the eHealth domain was very high, 97%, with a manageable level of warnings, 81%. The versatility of $RM_{p(c)}$ was also examined (8.3.1) finding that this ability was maintained when using human knowledge, allowing an expert to vary the accuracy and number of warnings. Additionally, tests on how much the KB was affected by missed warnings reinforced the idea that the system was robust.

This is of great importance, as it shows that not only were the simulations reasonably accurate in many regards, but also that RM performs just as well in a full text, multi-classifiable domain with human knowledge. These results highlight that RM should be a viable approach to developing real applications designed for extracting human knowledge and releasing the expert from being required to check and verify all cases presented to the system.

9.2 Discussion

In developing the method for this section a small number of issues and questions arose concerning the methodology developed. This discussion section is aimed at briefly addressing some of these concerns to provide the reader with an understanding of some of these side issues. The first subsection discusses how RM would be expected to perform if attached to a KB that is already partially constructed. The second challenges a simplifying assumption made throughout all testing that we know at the outset how many classifications to expect. The final subsection discusses some problems encountered throughout testing. While these had been briefly identified earlier they are stated more clearly here as they are important issues if RM was to be applied in a real world system.

9.2.1 Building upon Existing Knowledge

All the studies undertaken in this thesis have been based on the idea that RM is applied from the first stages of knowledge acquisition. Fundamentally, it is the information gathered when rules are created that allows the system to learn quickly and effectively. This, however, does not mean the network cannot be applied retrospectively to an MCRDR tree. Providing the earlier cases used to build the tree have been retained, such as the stored cornerstone cases, then these can simply be used to train the network. Therefore, when being applied to an existing tree, the network can be initialised with random weights and input nodes for each current feature from the tree, such as an input node for each rule. Then, the network can simply be trained through repeated viewings of each case in the exact same manner as any neural network.

There is one clear disadvantage of adding the network to a partly constructed tree. It is unable to use the shortcut initialisation rule. Therefore, this initial training process cannot occur online. However, the tree will still provide a flattening of the dimensionality of the problem space, thereby, allowing the system to quickly learn the required hidden contexts and associated values.

The resulting network after training will be expected to be able to provide the same classification and prediction results for the partially complete knowledge base as it would have been able to if it had been present during the KB creation. Furthermore, the KB can still then be built upon, after the network had been added, using the *Δ-initialisation-rule* in exactly the same manner as shown throughout this thesis.

While this process has not been directly tested during this project, there is no reason to expect the network would be unable to eventually learn the required functions. Neural networks perform exceptionally well – especially in relatively linear domains. The training of the network in the above situation is just a standard application of a neural network in a dimensionally flat domain. Additionally, once the network has been attached and trained there is essentially no difference between the current structure and what would have been formed, had the network been present from the outset.

9.2.2 Incrementing Network Classifications Online

Throughout this thesis all test results were generated with the network knowing how many classifications were possible in the dataset used. This is irrelevant in the prediction based methods as they only ever provided a single output node. However, in the classification based techniques the amount of output nodes was set at the outset. The problem with this approach is that in a real world domain there is no way to know how many unique classifications are going to be generated by the expert when adding knowledge to the MCRDR tree.

The alternative approach is to add new outputs when the expert creates a new and previously unseen class. The experimentation in this thesis did not test this approach. However, there is no reason to expect that the system would perform any differently. When a new output is added it would simply give all the weights from the already present hidden and input nodes the default value as detailed in chapter 5. This will certainly mean the performance will be the same. Secondly, the weight would be set on the new input to the new output using the Δ-initialisation rule, once again exactly as done previously. Thus, the adding of output nodes dynamically during run time should be no different as the new node does not affect and is not affected by any of the other nodes. A second possible approach is to create more output nodes than required and simply allocate them as needed. This, of course, assumes we can accurately determine a maximum. This method also should perform with the same level of performance.

9.2.3 Problems

There was one primary qualitative problem noticed when doing the testing in this thesis. This was mentioned briefly in section 8.3.2 on parameter tuning. This problem is the large number of parameters that require a setting prior to the system's use. One of the complaints about neural networks is the arbitrary nature for setting parameters. RM incorporates those problems from neural networks while also needing to select strategies used by MCRDR and, more importantly, the different types of interaction between the two methods. This problem was even further exacerbated during the prudence analysis section due to the introduction of additional thresholding parameters.

As a result every test of each method and dataset required adjustments to many of these parameters. Even now, the results published here may not be the best results possible. Many tests were run for each experiment to enable the selection of the most appropriate parameters. However, there may still be combinations not fully tested. The issues relating to parameter tuning caused all testing throughout this project to be extremely difficult and time consuming.

Fundamentally, this difficulty runs the risk of rendering RM unusable. It was found, however, that in some of the later testing, the parameter tuning process became significantly easier. For instance, when performing the prudence analysis on the eHealth dataset using the classification based approach, it was found that the parameters selected on the first attempt worked exceptionally well (and are the results used in chapter 8). This indicates that after a level of experience has been achieved that the parameters can be chosen reasonably accurately. Therefore, while the number of parameters is a major issue it should not prevent its application, providing developers gain sufficient experience.

Furthermore, in the results shown in this thesis it is unclear whether the best parameter settings were governed by the dataset or by the way the expert created rules. If each expert's behaviour is an important aspect in parameter selection, then it would be difficult to build generic software. While this was not directly tested, it was noticed that during testing with the C4.5 simulated expert that good parameters were effective for each different selection strategy trialled by the expert. This indicates the expert may not have a strong effect, but this requires further study using human expertise and behaviour patterns and is beyond the scope of this thesis.

9.3 Future Work

This thesis has travelled from the investigation into finding a new view of situation cognition and how that affects our understanding of knowledge, through to the development of a unique methodology. This algorithm has been shown to be exceptionally powerful at classification and prediction, thereby, reducing the brittleness of a KBS, while also showing exactly how it can be applied in the prudence domain with human expertise. However, the forum of a PhD dissertation is too brief to adequately follow all the possibilities that a new

technique such as this opens up. The aim of this brief section is to detail just a few possibilities available to RM. The first subsection briefly reviews some adaptations to the method that would allow its application in many other areas, while the second sub section looks at a few plausible applications.

9.3.1 Algorithm Additions

One very useful extension to the development of a prudence analysis method is that when a warning is generated it may be possible to analyse the ANN structure and identify which rule in MCRDR had contributed most to the generated warning. Once identified, a relatively simple statistical analysis on the attributes identified in a difference list, could identify the most significant and warranting selection for a rule. This would allow the system to not only identify misclassifications but also identify and create rules. Effectively, this introduces a unique form of incremental online induction, RM_{Induct}. This approach would be even further improved if some feedback information, a reward, could be supplied to aid in the rule identification.

Some very preliminary work in testing whether the correct rule could be identified had been carried out during the early stages of using RM in prudence analysis. The results had been promising, indicating that this could be worth further investigation. The primary problem in this approach is that incorrect warnings will potentially result in the creation of irrelevant rules. This should not prevent the ability of the KB to operate but would certainly result in a much larger structure than would be developed in a traditional induction method.

One issue not directly addressed in this dissertation was how noise may affect the learning ability of the ANN. Noise is not normally a concern in a KBS, but it is frequently considered when developing ANNs. Due to RM requiring some feedback this can sometimes be affected by noise within the environment being used. For instance, in a prediction domain, a reward may be received as a measure of the importance of a case to the user. However, the importance of this document can shift over time and the measuring of that importance can be affected by many external factors. Therefore, RM should have a means for addressing this issue.

The use of an ANN, however, has in itself mostly solved this issue. ANNs are extremely good at learning in a noisy environment - an ability RM could be

expected to have acquired. One approach used in ANNs to reduce the effect of noise even further, is to use a reducing gain. During early work with RM this approach was trialled, except it was performed on a per arc basis. Therefore the more times a connection was used the faster its gain was reduced. Those early studies had shown that the reducing gain technique significantly helped RM in noisy environments. This work, and the associated work in information filtering, published in the Pacific Rim International Conference in Artificial Intelligence (PRICAI)(Dazeley and Kang 2004a), were not included in this thesis as they were neither complete nor relevant to the hypotheses presented.

The final extension to RM identified here is the notion of applying RM in temporally separated situations, referred to as *RM(λ)*. In particular, by adding an eligibility trace to each arc, RM could potentially solve reinforcement learning type problems. This adjustment would have no effect on the existing methodology when applied in the previous domains. Reinforcement learning methods, such as Sarsa, Temporal Difference Learning (TD), Q-Learning, DG-learning and Concurrent-Q-Learning (CQL) are often used in conjunction with neural networks to allow for generalisation of the state and/or action spaces. Through applying these algorithms in an identical manner to the network used in RM, allows for a significantly more expressive state-action pairs' representation. Furthermore, this could incorporate RM_{Induct} ideas, allowing for a fully automatic reinforcement learning KBS. This could represent the ultimate solution to the ideas of the *process of practice*.

9.3.2 Possible Applications

One major potential application of RM using prudence analysis is in the domain of data mining. Effectively, the prudence analysis system, not only can be used to identify cases where the expert has failed to provide a classification, but also to identify special cases that the expert may not have been aware of in the first place. Fundamentally, this is the goal of KDD (Knowledge Discovery in large Databases) type systems. The case found could be an unusual shopping pattern by a particular group of customers or a previously unknown combination of drugs that produce a particular side effect. Furthermore, RM in its general sense could also offer further advantages to data mining through its application as a missing value predictor (Dazeley and Kang 2004c).

Another obvious application of RM is in the application domain of information filtering (IF). RMs application in IF is interesting as it allows the development of systems capable of being easily used by general users and fits the standard models for such a system. RM can offer an versatile approach to this domain. RM is most similar to the common approach of using keyword weightings. The difference with RM is that the keywords are selected by the expert and are contextually dependant. With the inclusion of methods for reducing noise, discussed in the previous subsection, RM becomes a potentially powerful tool. Furthermore, RM could be used to classify documents, as in the eHealth domain, or for rating the importance of each. For instance, placing emails in order of importance, or personalising the order results are presented from a search engine (Dazeley and Kang 2004a).

There are, in fact, very few application domains where RM could not be applied. Other applications are possible, such as natural language processing, Web-spidering, and even robot navigation are all potential areas of application. Generally, if a reward structure can be developed for the required goal and expert knowledge is available, then the system should be able to learn a fast and generalised solution.

9.4 A Final Word

The work in this project represents a significant paradigm shift in both philosophical thinking and methodological development for symbolic based applications. The dissertation supporting this thesis is only a first move in exploring the philosophical step taken. The proposed methodology represents only one plausible approach to the inclusion of hidden and dynamic contextual information in a symbolic methodology. The methodology itself has shown that it is a powerful classifier and predictor, and that the information it learns can successfully be applied in highly useful ways. Additionally, the application of the methodology developed has only been fully explored in one domain but many other uses are clearly identifiable.

References

- Aamodt, A. and E. Plaza (1994). "Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches." AI Communications **7**(1): 39-59.
- Agre, P. E. (1990). "Book Review of Plans and Situated: Actions: The Problem of Human-Machine Communication." Artificial Intelligence **43**(3): 369-384.
- Al-Jadir, L. and G. Beydoun (2002). Using the F2 OODBMS to Support Incremental Knowledge Acquisition. International Database Engineering & Applications Symposium, Edmonton, Canada, IEEE Computer Society.
- Anderson, J. (1990). Cognitive Psychology and its Implications, 3rd edn. New York, W. H. Freeman and Company.
- Arbib, M. (1993). "Book Review - Allen Newell, Unified Theories of Cognition." Artificial Intelligence **59**: 265 - 283.
- Bachant, J. and J. McDermott (1984). "R1 revisited: four years in the trenches." AI Magazine **5**(3): 21-32.
- Bartlett, F. C. (1932). Remembering: a study in experimental and social psychology. 1977, London, Cambridge University Press.
- Beale, R. and T. Jackson (1990). Neural Computing: An Introduction. Bristol, Great Britain, IOP Publishing Ltd.
- Beydoun, G. (2000). Incremental Knowledge Acquisition for Search Control Heuristics. School of Computer Science and Engineering. Sydney, University of New South Wales.
- Beydoun, G. (2002a). Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web. 13th International Conference, EKAW 2002, Siguenza, Spain.
- Beydoun, G. (2002b). An OO Model for Incrmental Hierarchical KA. Proceedings of the Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web : 13th International Conference, EKAW 2002, Siguenza, Spain, Springer-Verlag.
- Beydoun, G. and L. Al-Jadir (2002). Implementing NRDR Using OO Database Management System (OODBMS). PRICAI 2002: Trends in Artificial Intelligence : 7th Pacific Rim International Conference on Artificial Intelligence, Tokyo, Japan, Springer-Verlag Heidelberg.
- Beydoun, G. and A. Hoffmann (1997a). Acquisition of Search Knowledge. The 10th European Knowledge Acquisition Workshop (EKAW97), Spain, Springer.
- Beydoun, G. and A. Hoffmann (1997b). NRDR for the Acquisition of Search Knowledge. Proceedings of Tenth Australian Joint Conference On Artificial Intelligence, Perth, Australia.
- Beydoun, G. and A. Hoffmann (1998a). Building Problem Solvers Based on Search Control Knowledge. Proceedings of Eleventh Workshop on Knowledge Acquisition, Modeling and Management, Banff, Canada.
- Beydoun, G. and A. Hoffmann (1998b). Building search heuristics at the knowledge level. Pacific Rim Knowledge Acquisition Workshop (PKAW98), Singapore, National University of Singapore.
- Beydoun, G. and A. Hoffmann (1998c). Simultaneous Modelling and Knowledge Acquisition Using NRDR. Proceedings of 5th Pacific Rim International Conference on Artificial Intelligence, Singapore.
- Beydoun, G. and A. Hoffmann (1999a). A Formal Framework of Ripple Down Rules. The Fourth Australian Workshop on Knowledge Acquisition (AKAW1999), Sydney, University of New South Wales.

- Beydoun, G. and A. Hoffmann (1999b). Hierarchical Incremental Knowledge Acquisition. 12th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW99), Canada, SRDG publications.
- Beydoun, G. and A. Hoffmann (1999c). Holism and Incremental Knowledge Acquisition. Proceedings of The 11th European Workshop on Knowledge Acquisition, Management and Modelling, Dagstuhl Castle, Germany, Springer.
- Beydoun, G. and A. Hoffmann (2000a). "Incremental acquisition of search knowledge." International Journal of Human-Computer Studies **52**(3): 493-530.
- Beydoun, G. and A. Hoffmann (2000b). Knowledge Engineering and Knowledge Management. Methods, Models, and Tools. 12th International Conference, EKAW 2000, Juan-les-Pins, France, Springer-Verlag Heidelberg.
- Beydoun, G. and A. Hoffmann (2000c). Monitoring Knowledge Acquisition, Instead of Evaluating Knowledge Bases. 12th European Conference on Knowledge Acquisition and Knowledge Management (EKAW2000), France, Springer.
- Beydoun, G. and A. Hoffmann (2001). "Theoretical basis for hierarchical incremental knowledge acquisition." International Journal of Human-Computer Studies **54**(3): 407-452.
- Beydoun, G., R. B. H. Kwok, et al. (2000). Towards order independent incremental KA. Proceedings of the 6th Pacific Knowledge Acquisition Workshop, The University of New South Wales, Sydney, Australia.
- Birkhoff, G. (1938). "Lattices and their applications." Bulletin of the American Mathematics Society **44**: 793-800.
- Blake, C. L. and C. J. Merz (1998). UCI Repository of machine learning databases, University of California, Irvine, Dept. of Information and Computer Sciences.
- Boose, J. H., J. M. Bradshaw, et al. (1989). From ETS to Aquinas: Six Years of Knowledge Acquisition Tools. Proceedings of the 4th AAAI-Sponsored Knowledge Acquisition for Knowledge-Based Systems Workshop, Banff, Canada.
- Borges, J. L. (1956). Fictions. Paris, Gallimard.
- Bose, N. K. and P. Liang (1996). Neural Network Fundamentals with Graphs, Algorthms, and Applications. New York, McGraw-Hill, Inc.
- Bransford, J. D., N. S. McCarrell, et al. (1977). Toward unexplaining memory. Perceiving, Acting, and Knowing: Toward an Ecological Psychology. S. R. E. and B. J. D. Hillsdale, New Jersey, Lawrence Erlbaum Associates**:** 431-466.
- Brezillon, P. (1999). "Context in Artificial Intelligence: II. Key elements of contexts." Computer and Artificial Intelligence **18**(5): 425-446.
- Brooks, R. A. (1991). "Intelligence without Representation." Artificial Intelligence **47**: 139-159.
- Broomhead, D. S. and D. Lowe (1988). "Multivariable function interpolation and adaptive networks." Complex Systems **2**: 321 - 355.
- Buchanan, B. G. and R. G. Smith (1989). Fundamentals of Expert Systems. The Handbook of Artificial Intelligence. A. Barr, P. R. Cohen and E. A. Feigenbaum. Reading, Massachusetts, Addison-Wesley. **IV:** 151-192.
- Busch, P. A. and D. Richards (2004). Acquisition of Articulatable Tacit Knowledge. Proceedings of the Pacific Knowledge Acquisition Workshop 2004 (PKAW04), Auckland, New Zealand.
- Cao, T. M., T. Le Gia, et al. (1999). An Expert system using RDR that supports fuzzy rules. http://www.cse.unsw.edu.au/~tmc/Fuzzy/FRDR.html.
- Catlett, J. (1992). Ripple-Down-Rules as a Mediating Representation in Interactive Induction. AAAI Workshop on Knowledge Acquisition and Machine Learning.

- Cendrowska, J. (1987). "An algorithm for inducing modular rules." International Journal of Man-Machine Studies **27**(4): 349-370.
- Chaudhri, V. K., A. Farquhar, et al. (1998). OKBC: A programmatic foundation for knowledge base interoperability. AAAI'98, Madison, WI.
- Chellen, V. (1995). Induct/RDR and Knowledge Base Compaction. BE Computer Engineering Honours Thesis, University of New South Wales.
- Clancey, W. J. (1986). "From GUIDON to NEOMYCIN and HERACLES in Twenty Short Lessons: ORN Final Report 1979-1985." AI Magazine **August**: 40-60.
- Clancey, W. J. (1991). "Book Review - Israel Rosenfield, The Invention of Memory: A New View of the Brain." Artificial Intelligence **50**(2): 241-284.
- Clancey, W. J., P. Sachs, et al. (1998). "BRAHMS: Simulating Practice for Work System Design." International Journal of Human-Computer Studies **49**: 831 - 865.
- Colomb, R. and J. Sienkiewicz (1996). Analysis of Redundancy in Expert Systems Case Data. AI'95 Eighth Australian Joint Conference on Artificial Intelligence.
- Compton, P. (1992). Insight and Knowledge. AAAI Spring Symposium: Cognitive aspect of knowledge acquisition, Stanford University.
- Compton, P. (2000). Simulating Expertise. Proceedings of the 6th Pacific Knowledge Acquisition Workshop, Sydney, Australia.
- Compton, P., G. Edwards, et al. (1991). Ripple Down Rules: Possibilities and Limitations. 6th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW91), Canada, SRDG publications.
- Compton, P., K. Horn, et al. (1988). Maintaining an Expert System. 4th Australian Conference on Applications of Expert Systems, Sydney.
- Compton, P., K. Horn, et al. (1989). Maintaining an Expert System. Applications of Expert Systems. J. R. Quinlan. Sydney, Addison-Wesley. **2:** 366-384.
- Compton, P. and R. Jansen (1988). Knowledge in Context: a strategy for expert system maintenance. Second Australian Joint Artificial Intelligence Conference (AI88).
- Compton, P. and R. Jansen (1990). "A philosophical basis for knowledge acquisition." Knowledge Acquisition **2**: 241-257.
- Compton, P. and B. H. Kang (1993). Knowledge Histories. 2nd Australian Cognitive Science Conference.
- Compton, P., B. H. Kang, et al. (1993). Knowledge Acquisition Without Analysis. Knowledge Acquisition for Knowledge Based Systems, Berlin, Springer Verlag.
- Compton, P., P. Preston, et al. (1996). Knowledge based systems that have some idea of their limits. Tenth Knowledge Acquisition for Knowledge-Based Systems Workshop.
- Compton, P., P. Preston, et al. (1995). The Use of Simulated Experts in Evaluating Knowledge Acquisition. The 9th Knowledge Acquisition for Knowledge Based Systems Workshop, Department of Computer Science, University of Calgary, Calgary, Canada, SRDG Publications.
- Compton, P., P. Preston, et al. (1994). Local Patching Produces Compact Knowledge Bases. A Future for Knowledge Acquisition. L. Steels, S. G. and W. V. d. Velde. Berlin, Germany, Springer-Verlag**:** 104-117.
- Compton, P., Z. Ramadan, et al. (1998). A trade-off between domain knowledge and problem-solving method. 11th Banff Knowledge Acquisition for Knowledge Base System Workshop (KAW98), Canada, SRDG Publications.
- Compton, P. and D. Richards (1999). Extending ripple down rules. Proceedings of the Fourth Australian Knowledge Acquisition Workshop (AKAW99), Sydney, New South Wales University.

- Dazeley, R. and B. H. Kang (2003a). Rated MCRDR: Finding non-Linear Relationships Between Classifications in MCRDR. 3rd International Conference on Hybrid Intelligent Systems, Melbourne, Australia, IOS Press.
- Dazeley, R. and B. H. Kang (2003b). Weighted MCRDR: Deriving Information about Relationships between Classifications in MCRDR. The 16th Australian Joint Conference on Artificial Intelligence (AI'03), Perth, Australia, Springer-Verlag, Berlin Heidelberg New York.
- Dazeley, R. and B. H. Kang (2004a). An Augmented Hybrid System for Document Classification and Rating. PRICAI 2004: 8th Pacific Rim International Conference on Artificial Intelligence, Auckland, New Zealand, Springer.
- Dazeley, R. and B. H. Kang (2004b). Detecting the Knowledge Frontier: An Error Predicting Knowledge Based System. PKAW04: Pacific Rim Knowledge Acquisition Workshop, Auckland, New Zealand, Springer.
- Dazeley, R. and B. H. Kang (2004c). An Online Classification and Prediction Hybrid System for Knowledge Discovery in Databases. AISAT 2004: International Conference on Artificial Intelligence in Science and Technology, Hobart, Australia.
- Diaz-Agudo, B. and P. A. Gonzalez-Calero (2001). Classification Based Retrieval Using Formal Concept Analysis. ICCBR 2001, Springer-Verlag.
- Drake, B. J. and G. Beydoun (2000). Predicate Logic-based Incremental KA. Proceedings of the 6th Pacific-rim Knowledge Acquisition Workshop, Sydney, Australia.
- Edwards, G. (1996). Reflective Expert Systems in Clinical Pathology. Sydney, University of New South Wales.
- Edwards, G., P. Compton, et al. (1995a). New Paradigms for Expert Systems in Healthcare. Proceedings of the Fourth Health Informatics of NSW Conference, Primbee.
- Edwards, G., B. H. Kang, et al. (1995b). "Prudent expert systems with credentials: Managing the expertise of decision support systems." International Journal of Biomedical Computing **40**: 125-132.
- Estruch, V., C. Ferri, et al. (2003). Simple Mimetic Classifiers. Third International Conference, MLDM 2003, Leipzig, Germany,, Springer.
- Fahlman, S. E. and C. Lebiere (1990). The Cascade-Correlation Learning Architecture. Advances in Neural Information Processing Systems. D. S. Touretzky. Denver, Morgan Kaufmann, San Mateo. **2:** 524-532.
- Feigenbaum, E. A. (1977). The art of artificial intelligence: themes and case studies in knowledge engineering. (IJCAI) International Joint Conference on Artificial Intelligence.
- Fikes, R. and A. Farquhar (1997). Distributed Repositories of Highly Expressive Reusable Knowledge, Stanford KSL Technical Report 97-02.
- Frawley, W. J., G. Piatetsky-Shapiro, et al. (1992). "Knowledge Discovery in Databases: An Overview." AI Magazine **13**: 57 - 70.
- Frean, M. (1990). "The upstart algorithm: a method for constructing and training feedforward networks." Neural Computation **2**: 198 - 209.
- Freidson, E. (1994). Nourishing Professionalism. Professionalism Reborn: Theory, Prophecy and Policy. Chicago, University of Chicago Press**:** 199-216.
- Gaines, B. (1989a). Knowledge Acquisition: the Continuum Linking Machine Learning and Expertise Transfer. Proceedings of the 3rd European Workshop on Knowledge Acquisition for Knowledge-Based Systems Workshop, Paris.
- Gaines, B. (2000). Knowledge Science and Technology: Operationalizing the Enlightenment. Proceedings of the 6th Pacific Knowledge Acquisition Workshop, Sydney, Australia.

- Gaines, B. and M. L. G. Shaw (1990). Cognitive and Logical Foundations of Knowledge Acquisition. 5th AAAI Knowledge Acquisition for Knowledge Based Systems Workshop, Banff, Canada.
- Gaines, B. and M. L. G. Shaw (1993). "Knowledge Acquisition Tools Based on Personal Construct Psychology." Knowledge Engineering Review **8**(1): 49-85.
- Gaines, B. R. (1989b). An Ounce of Knowledge is Worth a Ton of Data: Quantitative Studies of the Trade-Off between Expertise and Data based on Statistical Well-Founded Empirical Induction. Proceedings of the 6th International Workshop on Machine Learning, San Mateo, California, Morgan Kaufmann.
- Gaines, B. R. (1991). The Tradeoff between knowledge and data in data acquisition. Knowledge Discovery in Databases. G. Piatetsky-Shapiro and W. Frawley. Cambridge, Massachusetts, MIT Press**:** 491-505.
- Gaines, B. R. and P. J. Compton (1992). Induction of Ripple Down Rules. Fifth Australian Conference on Artificial Intelligence (AI92), Hobart, World Scientific.
- Gaines, B. R. and P. J. Compton (1993). Induction of Ripple-Down Rules applied to Modelling Large Databases. Banff Knowledge Acquisition for Knowledge Base System Workshop.
- Gama, J. (1998). Combining Classifiers with Constructive Induction. Proceedings of ECML-98.
- Gennari, J. H., M. A. Musen, et al. (2002). The Evolution of Protégé: An Environment for Knowledge-Based Systems Development, Stanford Medical Informatics (SMI)**:** 32.
- Grosso, W. E., H. Eriksson, et al. (1999). Knowledge Modeling at the Millennium (The Design and Evolution of Protege-2000). 12th International Workshop on Knowledge Acquisition, Modeling and Management (KAW'99), Banff, Canada.
- Guha, R. V. and D. B. Lenat (1994). "Enabling agents to work together." Communications of the ACM **37**(7): 127-142.
- Han, J. and M. Kamber (2001). Data Mining: Concepts and Techniques. San Francisco, Morgan Kaufmann Publishers.
- Han, J. and C. Moraga (1995). The Influence of the Sigmoid Function Parameters on the Speed of Backpropagation Learning. Natural to Artificial Neural Computation, International Workshop on Artificial Neural Networks, Malaga, Spain, Springer-Verlag.
- Hebb, D. O. (1949). Organization of Behavior. New York, Wiley.
- Horn, K., P. Compton, et al. (1985). "An expert computer system for the interpretation of thyroid assays in a clinical laboratory." Aust. Comp. J. **17**(1): 7-11.
- Ignizio, J. P. (1991). Introduction to Expert Systems: The Development and Implementation of Rule-Based Expert Systems. New York, McGraw-Hill Inc.
- Jansen, B. and P. Compton (1988). The Knowledge Dictionary: A Relational Tool for the Maintenance of Expert Systems. Proceedings of the International Conference on Fifth Generation Computer Systems FGCS'88.
- Jansen, R. and P. Compton (1989). The Knowledge Dictionary: Storing Different Knowledge Representations. Proceedings 5th Australian Conference on Applications of Expert Systems.
- Jenkins, J. J. (1974). "Remember that old theory of memory? Well, forget it!" American Psychologist: 785-795.
- Kang, B. H. (1996). Validating Knowledge Acquisition: Multiple Classification Ripple Down Rules. Sydney, University of New South Wales.
- Kang, B. H. and P. Compton (1992). Knowledge Acquisition in Context : the Multiple Classification Problem. The 2nd Pacific Rim International Conference on Artificial Intelligence, Seoul, Korea.

- Kang, B. H. and P. Compton (1994). Multiple Classification Ripple Down Rules. Third Japanese Knowledge Acquisition for Knowledge-Based Systems Workshop, Hatoyama, Japan, Japanese Society for Artificial Intelligence.
- Kang, B. H., P. Compton, et al. (1995). Multiple Classification Ripple Down Rules: Evaluation and Possibilities. The 9th Knowledge Acquisition for Knowledge Based Systems Workshop, Department of Computer Science, University of Calgary, Banff, Canada, SRDG Publications.
- Kang, B. H., W. Gambetta, et al. (1994). Verification and Validation with Ripple Down Rules. Validation and Verification of Knowledge Based Systems (AAAI-94 Workshops), Seattle, USA, AAAI Press.
- Kang, B. H., W. Gambetta, et al. (1996). "Verification and Validation with Ripple Down Rules." International Journal of Human-Computer Studies **44**(2): 257-268.
- Kang, B. H., P. Preston, et al. (1998). Simulated Expert Evaluation of Multiple Classification Ripple Down Rules. Eleventh Workshop on Knowledge Acquisition, Modeling and Management, Voyager Inn, Banff, Alberta, Canada.
- Karp, P. D., V. K. Chaudhri, et al. (1999). "A collaborative environment for authoring large knowledge bases." Journal of Intelligent Information Systems **13**: 155-194.
- Kelly, G. A. (1955). The Psychology of Personal Constructs. New York, Norton.
- Kelly, G. A. (1970). A brief introduction to personal construct theory. Perspectives in Personal Construct Theory. D. Bannister. London, Academic Press**:** 1-29.
- Khan, A. S. (2003). Incremental Knowledge Acquisition for Case-Based Reasoning. School of Computing Science and Engineering. Sydney, University of New South Wales (UNSW)**:** 200.
- Khan, A. S. and A. Hoffmann (2003). "Building a case-based diet recommendation system without a knowledge engineer." Artificial Intelligence in Medicine **27**(2): 155-179.
- Kidd, A. L. and W. P. Sharpe (1987). Goals for Expert Systems Research: An Analysis of Tasks and Domains. Expert Systems IV, Cambridge University Press**:** 146-152.
- Kildare, R. (2004). Ad-hoc online teams as complex systems: agents that cater for team interaction rules. Proceedings of the 7th Asia-Pacific Conference on Complex Systems, Cairnes, Australia.
- Kildare, R., R. Williams, et al. (2004). Expert Software Support for Ad-Hoc Teams - Enabling Online Interaction Rules. Proceedings of the 2nd International Conference on Artificial Intelligence in Science and Technology (AISAT 2004), Hobart, Australia, University of Tasmania.
- Killin, J. (1993). Managing and Maintaining an Operational KADS System. A Principled Approach to Knowledge-Based Development. G. Schreiber, B. J. Wielinga and J. Breuker. London, Academic Press. **11:** 457.
- Kim, Y. S., S. S. Park, et al. (2004a). Adaptive Web Document Classification with MCRDR. ITCC04: International Conference on Information Technology, Las Vegas, Nevada, USA.
- Kim, Y. S., S. S. Park, et al. (2004b). Incremental Knowledge Management of Web Community Groups on Web Portals. PAKM2004: Practical Aspects of Knowledge Management, Vienna, Austria.
- Leake, D. B., Ed. (1996). Case-Based Reasoning: Experiences, Lessons, and Future Directions. Menlo Park, CA, AAAI Press/MIT Press.
- Lee, M. R. (1996). From Multiple Representations to Casual Explanations. School of Computer Science and Engineering. Sydney, Australia, University of New South Wales.
- Lee, M. R. and P. Compton (1994). Context-Dependent Casual Explanations. Eighth International Workshop on Qualitative Reasoning About Physical Systems, Nara, Japan.

- Lee, M. R. and P. Compton (1995). From Heuristic Knowledge to Casual Explanations. AI'95 Eighth Australian Joint Conference on Artificial Intelligence, Singapore, World Scientific.
- Lee, M. R. and P. Compton (1996). From Heuristics to causality. Proceedings of the third world congress on expert systems, Seoul, Cognizant Communications.
- Lenat, D. B. (1995). "CYC: A Large-Scale Investment in Knowledge Infrastructure." Communications of the ACM **38**(11): 33-38.
- Lenat, D. B. and E. A. Feigenbaum (1988). On the threshold of knowledge. fourth Australian Conference on Applications of Expert Systems.
- Lenat, D. B. and E. A. Feigenbaum (1991). "On the threshold of knowledge." Artificial Intelligence **47**(1-3): 185-250.
- Lenat, D. B. and R. V. Guha (1990). Building large knowledge-based systems : representation and inference in the Cyc project. Reading, Massachusetts, Addison-Wesley Pub. Co.
- Lenat, D. B., R. V. Guha, et al. (1990). "Cyc: Toward Programs with Common Sense." Communications of the ACM **33**(8): 30-49.
- Liang, P. (1990). Problem decomposition and subgoaling in artificial neural networks. Proceedings IEEE Conference on Syst., Man and Cybernetics, Los Angeles, CA, IEEE Press, Piscataway, NJ.
- Liang, P. (1991). Design of artificial neural networks based on the principle of divide-and-conquer. Proceedings IEEE International Symposium on Circuits and Systems, Singapore, IEEE Press, Piscataway, NJ.
- Linster, M. E. (1992). Sisyphus'91: Models of Problem Solving. GMD (Gesellschaft fur Mathematik und Datenverarbeitung mbH).
- MacGregor, R. M. (1991). Using a description classifier to enhance deductive inference. Seventh IEEE Conference on AI Applications.
- Maes, P. (1990). "Designing Autonomous Agents." Robotics and Autonomous Systems **6**(1,2): 1-196.
- Mansuri, Y., J. G. Kim, et al. (1991a). A comparison of a manual knowledge acquisition method and an inductive learning method. Australian workshop on knowledge acquisition for knowledge based systems, Pokolbin, Australia.
- Mansuri, Y., J. G. Kim, et al. (1991b). An evaluation of Ripple-Down Rules. Proceedings of the IJCAI'91 Knowledge Acquisition Workshop, Pokolbin.
- Martinez-Bejar, R., V. R. Benjamins, et al. (1998a). A formal framework to build domain knowledge ontologies for ripple-down rules-based systems. 11th Banff Knowledge Acquisition for Knowledge Base System Workshop (KAW98), Canada, SRDG.
- Martinez-Bejar, R., V. R. Benjamins, et al. (1997). Designing operators for constructing domain knowledge ontologies. Knowledge Acquisition, Modeling and Management , Lecture Notes in Artificial Intelligence. E. Plaza and R. Benjamins. Berlin, Springer-Verlag**:** 159-173.
- Martinez-Bejar, R., F. Ibanez-Cruz, et al. (1999a). A reusable framework for incremental knowledge acquisition. Proceedings of the Fourth Australian Knowledge Acquisition Workshop (AKAW99), Sydney, Australia.
- Martinez-Bejar, R., F. Ibáñez-Cruz, et al. (1999b). "FMR: an incremental knowledge acquisition system for fuzzy domains." Lecture Notes in Artificial Intelligence. **1621**: 349-364.
- Martinez-Bejar, R., H. Shiraz, et al. (1998b). Using Ripple Down Rules-based Systems for Acquiring Fuzzy Domain Knowledge. 11th Banff Knowledge Acquisition For Knowledge-Based Systems Workshop (KAW98), Canada, SRDG publications.

- McCulloch, W. S. and W. A. Pitts (1943). "A logical calculas of the ideas immanent in neural nets." Bull. Math. Biophysics **5**: 115-133.
- Menzies, T. (1996). Assessing Responses to Situated Cognition. Tenth Knowledge Acquisition for Knowledge-Based Systems Workshop, Catelonia, Spain.
- Menzies, T. (1998). "Towards Situated Knowledge Acquisition." International Journal of Human-Computer Studies **49**: 867-893.
- Menzies, T. and J. Debenham (2000). Expert System Maintenance. Encyclopaedia of Computer Science and Technology. A. Kent and J. G. Williams, Marcell Dekker Inc. **42:** 35-54.
- Mezard, M. and J. Nadal (1990). "Learning in feedforward layered networks: the tiling algorithm." Jopurnal of Physics Association **22**: 2191 - 2203.
- Milne, A. A. (1926). Winnie-The-Pooh. reprint (2003), London, Egmont Books Limited.
- Minsky, M., L. and S. A. Papert (1969). Perceptrons.
- Moody, J. (1989). Fast learning in multi-resolution hierarchies. Advances in Neural Information Processing Systems. D. Touretzky. San Mateo, Morgan-Kaufmann**:** 29 - 39.
- Moody, J. and C. Darken (1988). Learning with localized receptive fields. Proceedings of the 1988 Connectionist Models Summer School, San Mateo, Morgan-Kaufmann.
- Moody, J. and C. Darken (1989). "Fast learning in networks of locally-tuned processing units." Neural Computation **1**(2): 281 - 294.
- Muggleton, S. (1987). Duce, An oracle-based approach to contructive induction. Proceedings of the Tenth International Joint Conference on Artificial Intelligence.
- Mulholland, M., D. Delzoppo, et al. (1993a). An Expert System For Ion Chromatography Implemented In Ripple Down Rules. Proceedings of the 12th Australian Symposium on Analytical Chemistry, Perth, Australia.
- Mulholland, M., P. Preston, et al. (1993b). "Teaching a computer ion chromatography." Chemistry in Australia **60**(5).
- Mulholland, M., P. Preston, et al. (1993c). The Use Of Machine Learning To Develop An Expert System For Ion Chromatography. Proceedings of the 12th Australian Symposium on Analytical Chemistry, Perth, Australia.
- Mulholland, M., P. Preston, et al. (1993d). An expert system for ion chromatography developed using machine learning and knowledge in context. Proceedings of the Sixth International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Edinburgh.
- Musen, M. A. (1987). "Use of a domain model to drive an interactive knowledge-editing tool." International Journal of Man-Machine Studies **26**: 105-121.
- Naidoo, Y. (1998). Prudence in RDR. School of Computer Science and Engineering. Sydney, Australia, University of New South Wales**:** 51.
- Newell, A. (1982). Computer structures : principles and examples. New York, McGraw-Hill.
- Newell, A. and H. A. Simon (1976). "Computer Science as empirical Inquiry: Symbols and Search." Communications of the ACM **19**(3): 113-126.
- Ollington, R. B. (2004). Localisation and Navigation: Applying Biological Principles to Mobile Robotics. School of Computing. Hobart, University of Tasmania.
- Ollington, R. B. and P. W. Vamplew (2003). Adaptive Response Function Neurons. Proceedings 2nd International Conference on Computational Intelligence, Robotics and Autonomous Systems, Singapore.

- Ollington, R. B. and P. W. Vamplew (2004). Learning Place Cells from Sonar Data. Proceedings AISAT2004: International Conference on Artificial Intelligence in Science and Technology, Hobart, Tasmania, Australia.
- Park, S. S. (2005). Discussion of MonClassifier changes. R. Dazeley. Hobart.
- Park, S. S., Y. S. Kim, et al. (2003). Web Information Management System: Personalization and Generation. IADRS International Conference WWW/Internet 2003, Algarve, Portugal.
- Park, S. S., Y. S. Kim, et al. (2004a). Personalized Web Document Classification using MCRDR. PKAW04: Pacific Rim Knowledge Acquisition Workshop, Auckland, New Zealand, Springer.
- Park, S. S., Y. S. Kim, et al. (2004b). Web Document Classification: Managing Context Change. IADIS International Conference WWW/Internet 2004, Madrid, Spain.
- Patel, V. and M. Ramoni (1997). Cognitive Models of Directional Inference in Expert Medical Reasoning. Expertise in Context. P. F. Feltovich and R. Hoffman. Cambridge, MA, MIT Press**:** 67-99.
- Piaget, J. (1970). Genetic Epistemology. New York, Columbia University Press, 1970.
- Pittman, K. and D. B. Lenat (1993). Representing knowledge in CYC-9, MCC Technical Report CYC-175-93P.
- Platt, J. (1991). "A Resource-Allocating Network for Function Interpolation." Neural Computation **3**: 213 - 225.
- Popper, K. R. (1963). Conjectures and Refutations. London, Routledge and Kegan Paul Ltd.
- Popper, K. R. (1968). Epistemology without a knowing subject. Logic, Methodology and Philosophy of Science III. B. V. Rootselaar. Amsterdam, North-Holland**:** 333-373.
- Preece, A. D. (1995). "Validation of knowledge-based systems: Current trends and issues." The Knowledge Engineering Review **10**(1): 69 - 71.
- Preston, P., G. Edwards, et al. (1993). A 1600 Rule Expert System Without Knowledge Engineers. Moving Towards Expert Systems Globally in the 21st Century (Proceedings of the Second World Congress on Expert Systems 1993), New York.
- Preston, P., G. Edwards, et al. (1994). A 2000 Rule Expert System Without a Knowledge Engineer. Proceedings of the 8th AAAI-Sponsored Banff Knowledge Acquisition for Knowledge-Based Systems Workshop, Banff, Canada.
- Puerta, A., J. Egar, et al. (1992). "A Multiple-Method Knowledge-Acquisition Shell for the Automatic Generation of Knowledge-Acquisition Tools." Knowledge Acquisition **4**(2): 171-196.
- Quinlan, J. R. (1987). "Simplifying decision trees." International Journal of Man-Machine Studies **27**(3): 221-234.
- Quinlan, J. R. (1993). C4.5: Programs for Machine Learning. San Mateo, California, Morgan Kaufmann Publishers.
- Ramesh, B. and V. Dhar (1992). "Supporting systems development by capturing deliberations during requirements engineering." IEEE Transactions on Software Engineering **18**: 498-510.
- Richards, D. (1998a). The Reuse of Knowledge in Ripple Down Rule Knowledge Based Systems. School of Computer Science and Engineering. Sydney, University of New South Wales**:** 336.
- Richards, D. (1998b). Ripple Down Rules with Formal Concept Analysis: A Comparison to Personal Construct Psychology. 11th Workshop on Knowledge Acquisition, Modeling and Management (KAW'98), Banff, Canada, SRDG Publications, Department of Computer Science, University of Calgary, Calgary, Canada,.

- Richards, D. (1999). The Reuse of Knowledge: Research, Issues, and the Ripple-Down Rules Approach. The Fourth Australian Knowledge Acquisition Workshop (AKAW99), Sydney, Australia, The University of New South Wales.
- Richards, D. (2000a). Giving Experts Ownership and Control over their Knowledge. Proceedings of OzCHI2000: Interfacing Reality in the New Millennium, University of Technology, Sydney, Australia.
- Richards, D. (2000b). Negotiating a Shared Conceptual Model using Groupware. Proceeding of the 11th Australasian Conference on Information Systems, Brisbane.
- Richards, D. (2000c). A Process Model for Requirements Elicitation,. Proceeding of the 11th Australasian Conference on Information Systems, Brisbane, Australia.
- Richards, D. (2000d). Reconciling conflicting sources of expertise: a framework and illustration. Proceedings of the 6th Pacific Knowledge Acquisition Workshop, Sydney, Australia.
- Richards, D. (2000e). The Visualisation of Multiple Views to Support Knowledge Reuse. Intelligent Information Processing Conference (IIP'2000) In conjunction with the 16th IFIP World Computer Congress WCC2000, Beijing, China.
- Richards, D. (2001). "Knowledge Based System Explanation: The Ripple Down Rules Alternative." International Journal of Knowledge and Information Systems **5**(1): 2-25.
- Richards, D. (2004). "Addressing the Ontology Acquisition Bottleneck through Reverse Ontological Engineering." Journal of Knowledge and Information Systems (KAIS) **6**: 402-427.
- Richards, D. and P. A. Busch (2000). Measuring , Formalising and Modelling Tacit Knowledge. International Congress on Intelligent Systems and Applications (ISA'2000).
- Richards, D. and P. A. Busch (2001). Acquiring and Applying Contextualised Tacit Knowledge. Australian Conference for Knowledge Management & Intelligent Decision Support (ACKMIDS' 2001), Melbourne, Australia.
- Richards, D., V. Chellen, et al. (1996). The Reuse of RDR Knowledge Bases: Using Machine Learning to Remove Repitition. Pacific Knowledge Acquisition Workshop (PKAW96), Sydney, Australia.
- Richards, D. and P. Compton (1997a). Combining Formal Concept Analysis and Ripple Down Rules to Support Reuse. Software Engineering and Knowledge Engineering (SEKE97), Madrid, Springer-Verlag.
- Richards, D. and P. Compton (1997b). Knowledge Acquisition First, Modelling Later. Proceedings of the Tenth European Knowledge Acquisition Workshop.
- Richards, D. and P. Compton (1999). Revisiting Sisyphus I- An Incremental Approach to Resource Allocations using Ripple Down Rules. 12th Banff Knowledge Acquisition for Knowledge Base System Workshop (KAW99), Canada, SRDG.
- Richards, D. and T. Menzies (1998). Extending the SISYPHUS III Experimentfrom a Knowledge Engineering Task to a Requirements Engineering Task. Task 11th Workshop on Knowledge Acquisition, Modeling and Management, Banff, Canada, SRDG Publications.
- Richards, D. and S. Simoff (2001). "Design ontology in context - a situated cognition approach to conceptual modelling." Special Issue on Conceptual Modelling in Design Computing in the International Journal of Artificial Intelligence in Engineering **15**: 121-136.
- Richards, D. and D. Zowghi (1999). Maintaining and comparing Requirments. Proceedings of the Fourth Australian Conference on Requirements Engineering ACRE'99, Macquarie University, Sydney, Australia.
- Rissanen, J. (1978). Modeling by shortest data description. Automatica**:** 465.

- Rosenblatt, F. (1958). "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain." Psychological Review **65**(6): 386-408.
- Rumerlhart, D. E. and J. L. McClelland (1986). Parallel distributed processing: Explorations in the microstructure of cognition. Volume 1. Cambridge, MA, MIT Press.
- Russell, S. J. and P. Norvig (1995). Artificial Intelligence: Modern Approach. Prentice Hall.
- Scheffer, T. (1996). Algebraic Foundation and Improved Methods of Induction of Ripple Down Rules. Proceedings of the Pacific Knowledge Acquisition Workshop (PKAW'96).
- Schreiber, G. (1993). Operationalizing Models of Expertise. A Principled Approach to Knowledge-Based Development. G. Schreiber, B. J. Wielinga and J. Breuker. London, Academic Press. **11:** 457.
- Schreiber, G., B. Wielinga, et al. (1993). KADS: A Principled Approach to Knowledge-Based System Development. Elsevier.
- Shaw, M. L. G. (1988). Validation in a Knowledge Acquisition System with Multiple Experts. (FGCS) Fifth Generation Computer Systems 1988, Tokyo, Japan, OHMSHA Ltd. Tokyo and Springer-Verlag.
- Shaw, M. L. G. and B. Gaines (1992). "Kelly's "Geometry of Psychological Space" and its significance for Cognitive Modeling." The New Psychologist.
- Shiraz, G. M., P. Compton, et al. (1998). FROCH: A Fuzzy Expert System with Easy Maintenance. International Conference on Intelligent Systems and Cybernetics, San Diego, CA, SRDG Publications. USA.
- Shiraz, G. M. and C. A. Sammut (1998). Acquiring Control Knowledge from Examples Using Ripple-Down Rules and Machine Learning. 11th Workshop on Knowledge Acquisition, Modeling and Management, Banff, Canada, SRDG Publications, Department of Computer Science, University of Calgary, Calgary, Canada.
- Shiraz, G. M. and C. A. Summut (1995). An incremental Method for Learning to Control Dynamic Systems. The Machine Learning Workshop of the IJCAI-95, Montreal, Canada.
- Shiraz, G. M. and C. A. Summut (1997). Combining Knowledge Acquisition and Machine Learning to Control Dynamic Systems. The Fifteenth International Joint Conference on Artificial Intelligence (IJCAI-97), Nagoya, Japan.
- Soininen, T. and M. Stumptner (2003). "Editorial." Artificial Intelligence for Engineering Design, Analysis and Manufacturing **17**(1).
- Stumme, G., R. Studer, et al. (2000). Towards an Order-Theoretical Foundation for Maintaining and Merging Ontologies. Proceedings of Referenzmodellierung 2000, Siegen, Germany.
- Suchman, L. A. (1987). Plans and Situated Actions: The Problem of Human-Machine Communication, Cambridge Press.
- Suryanto, H. (2004). Learning and Discovery in Incremental Knowledge Acquisition: Ripple Down Rules. Department of Artificial Intelligence, School of Computer Science and Engineering. Sydney, Australia, University of New South Wales**:** 170.
- Suryanto, H. and P. Compton (2000a). Discovery of Class Relations in Exception Structured Knowledge Bases. Conceptual Structures: Logical, Linguistic, and Computational Issues (ICCS-2000), Darmstadt Germany, Springer.
- Suryanto, H. and P. Compton (2000b). Learning classification taxonomies from a classification knowledge based system. Proceedings of the First Workshop on Ontology Learning OL'2000, Berlin, Germany.
- Suryanto, H. and P. Compton (2001). Discovery of Ontologies from Knowledge Bases. Proceedingsof the First International Conference on Knowledge Capture, New York, USA.

- Suryanto, H. and P. Compton (2002). Intermediate Concept Discovery in Ripple-Down Rule Knowledge Bases. Working Notes in the 2002 Pacific Rim Knowledge Acquisition Workshop (PKAW 2002), Tokyo, Japan.
- Suryanto, H. and P. Compton (2003). Invented Predicates to Reduce Knowledge Acquisition Effort. IJCAI workshop, Acapulco, Mexico.
- Suryanto, H. and P. Compton (2004). Invented predicates to reduce knowledge acquisition. 14th International Conference on Knowledge Engineering and Knowledge Management (EKAW04), Whittlebury Hall, Northamptonshire, UK.
- Sutton, R. and A. Barto (1998). Reinforcement Learning: An Introduction. Cambridge, Massachusetts, A Bradford Book, The MIT Press.
- Swartout, W. and J. Moore (1993). Explanation in Second generation ES. Second Generation Expert Systems. J. M. David, J. P. Krivine and R. Simmons. Berlin, Springer**:** 543-585.
- Tesauro, G. J. (1994). "Temporal Difference Learning and TD-Gammon." Communications of the ACM **38**: 58-68.
- Vamplew, P. (1996). Recognition of Sign Language Using Neural Networks. School of Computing. Hobart, University of Tasmania**:** 215.
- Vamplew, P. W. and A. Adams (1992). Missing Values in a Backpropagation Neural Net. Proceedings of ACNN'92: The Third Australian Conference on Neural Networks, Canberra, Australia.
- Vamplew, P. W., D. Clark, et al. (1996). Techniques for Dealing with Missing Values in Feedforward Networks. ACNN'96: Proceedings of the Seventh Australian Conference on Neural Networks, Canberra, Australia.
- Vera, A. and H. A. Simon (1993a). "Situated Action: A Response to Reviewers." Cognitive Science **17**: 77-86.
- Vera, A. and H. A. Simon (1993b). "Situated Action: A Symbolic Interpretation." Cognitive Science **17**: 7-48.
- Vera, A. and H. A. Simon (1993c). "Situated Action: Reply to William Clancey." Cognitive Science **17**: 117-133.
- Wada, T., T. Horiuchi, et al. (1999). Characterization of Default Knowledge in Ripple Down Rules Method. Methodologies for Knowledge Discovery and Data Mining: Third Pacific-Asia Conference, PAKDD-99, Beijing, China, Springer.
- Wada, T., T. Horiuchi, et al. (2000). Integrating Inductive Learning and Knowledge Acquisition in the Ripple Down Rules Method. Proceedings of the 6th Pacific-Rim Knowledge Acquisition Workshop, Sydney, Australia.
- Wada, T., T. Horiuchi, et al. (2001a). "A Description Length-Based Decision Criterion for Default Knowledge in the Ripple Down Rules Method." Knowledge and Information Systems **3**(2): 146-167.
- Wada, T., H. Motoda, et al. (2001b). Knowledge Acquisition from Both Human Expert and Data. PAKDD 2001, 5th Pacific-Asia Conference, Hong Kong, China, Springer.
- Wada, T., T. Yoshida, et al. (2002a). Extension of the RDR Method That Can Adapt to Environmental Changes and Acquire Knowledge from Both Experts and Data. PRICAI 2002: Trends in Artificial Intelligence : 7th Pacific Rim International Conference on Artificial Intelligence, Tokyo, Japan, Springer.
- Wada, T., T. Yoshida, et al. (2002b). Reorganize Knowledge Base of RDR for Adaption to Environmental Changes. Working Notes in the 2002 Pacific Rim Knowledge Acquisition Workshop (PKAW 2002), Tokyo, Japan.
- Waldrop, M. M. (1990). Fast, cheap, and out of control. Research News.

- Webster, M. (1995). The relationship between models and Heuristics. School of Computer Science and Engineering. Sydney, Australia, Univerity of New South Wales.
- Widmer, G. and M. Kubat (1996). "Learning in the Presence of Concept Drift and Hidden Contexts." Machine Learning **23**(1): 69 - 101.
- Widrow, B. and M. E. Hoff (1960). "Adaptive switchingcircuits." IRE WESTCON Connection Record **4**: 96-104.
- Wielinga, B. J., T. Schreiber, A., et al. (1992). "KADS: a modeling approach to knowledge engineering." Knowledge Acquisition **4**(1): 5-54.
- Wille, R. (1981). Restructuring Lattice Theory: An Approach Based on Hierarchies of Concepts. Ordered Sets: Proceedings of the NATO Advanced Study Institute held at Banff, Canada. I. Rival, D. Reidel Publishing**:** 445-472.
- Wille, R. (1989). Knowledge acquisition by methods of formal concept analysis. Data analysis, learning symbolic and numeric knowledge, Antibes, Nova Science Publishers, Inc.
- Wille, R. (1992). "Concept Lattices and Conceptual Knowledge Systems." Computers Mathematical Applications **23**(6-9): 493-515.
- Wolpert, D. H. (1992). "Stacked Generalization." Neural Networks **5**(2): 241-259.
- WordNet (2003). WordNet, Princeton University. **2006:** http://dictionary.reference.com/.
- Zadeh, L. A. (1965). "Fuzzy sets, Information and Control." Information and Control **8**: 338-353.
- Zurada, J. M. (1992). Introduction to Artificial Neural Systems. St Paul, West Publishing Company.

Appendix A: Chapter Quotes

Each chapter in this thesis began with a quote from the famous series of children's books about Winnie the Pooh, by A. A. Milne. Pooh's philosophy on life and simple interpretation of the world around him provides great and simple examples of knowledge inference that are both important and straightforward. These misinterpretations of the world lead toward many of the adventures in the 100 acre wood.

Milne, A. A. (1926). Winnie-The-Pooh. reprint (2003), London, Egmont Books Limited.

Appendix B: Notation

This Appendix details all the various notations used in equations throughout this thesis. Most notations used are standardised with the existing literature, however, some were changed to avoid clashes. A couple of notations are used in different places to represent different terms; it should be clear from a notation's context as to which of the definitions apply. These are ordered according to the order they appear in the thesis.

Notation	Description
net	The *weighted sum* of the inputs to a perceptron prior to having the thresholding function applied.
n	Number of elements (such as the number of nodes)
j	Index for the current *input source* (either an input or hidden node)
w	The *weight* attached to the arc from an input node to the current node. A width modifier in the Multi-Class-Prediction simulated expert.
x	The value of *input* into a particular node of an ANN
f(net)	The *threshold function* applied to the weighted sum.
e	Natural constant $\left(1+1/n\right)^n = 2.71828$ (rounded to 5 decimal places)
k	A positive *constant* that controls the spread of the sigmoid function.
δ	The amount of *error* at a particular neuron.
p	A particular input *pattern*. The *percentage ratio* of hidden nodes to input nodes. A percentage of 200 will result in 2 hidden nodes being added for every additional input node. Percentage of membership in the Multi-Class-Prediction simulated expert.
o	Index for the current *output* node.

h	Index for the current *hidden* node.
$f'(net)$	The *derivative* of the *thresholding function* applied to the weighted sum at a particular neuron.
v^r	The *value* of the *reward* applied to a particular neuron.
v^c	The *value* of the output node's original *conclusion*.
l	Index for the current *output source* (either an hidden or output node)
t	A moment in *time*.
η	A network's *gain* term (often referred to as *learning rate*).
α	A network's *moment* term.
s_k	A *function* in *k-dimensional space*
λ	A particular *coefficient* that is learned for each basis function
ϕ	A *basis function*.
y	The *centre* of a hyperellipse
σ	A positively valued *shaping parameter* used to maintain control over the acceptable distance from the centre of a basis function.
d	A *distance* measure – usually using Euclidean distance.
i	Index for the current *input* node.
c	A Case The centre value for a given class in the Multi-Class-Prediction simulated expert.
list	A list of Classifications found by the MCRDR inferencing process
$\bar{x}$	*Input array* for neural network
$\bar{v}$	A *vector* of neurons' output *values*.
R	A set of *rules*.
$\wp$	The power set function.
R^*	A set of all the *known rules*.

R^T	A set of *terminating rules*.
C	A set of *classifications*.
C^*	A set of all the *known classifications*.
A	A set of *Attributes* from fired rules.
A^*	A set of all the *known Attributes*.
F	A set of features. A feature is a rule, class or attribute.
F^*	A set of all the *known Features*.
$F^\#$	A set containing *all* the *features* both known and unknown.
P	A set of rules that fired within a particular path. All paths contain the root node, the terminating node and all the nodes in between.
r^d	The depth of a rule within a particular path of fired rules followed. $r^d=0$ at the root node and $r^d=\|\|P\|\|$ at the terminating rule.
r^v	The activation *value* for the input neuron associated with a particular rule.
r	A single *rule*.
v	A single neurons output *value*. Short hand for the more specific notation identified above v^o.
RM_w	Rated MCRDR using the weighted average
ζ	*Step-distance* modifier. Allows partial steps to be taken rather than full steps.
m	The *number* of *features* added to the system during a single case's correction phase.
$RM_{l(\varepsilon)}$	Rated MCRDR using a linear network and random initialisation.
$RM_{l(\Delta)}$	Rated MCRDR using a linear network and the single-step-Δ-initialisation-rule.
Δ_l	Initialisation rule for a single layered linear network
ws	The weighted sum at a particular node.
δ_{ws}	The error in the weighted sum at a particular node.

T_{ws}	The *target* required for a particular *weighted sum*. The target is the reward after it has been passed back through the inverse of the thresholding function.
$RM_{bp(\varepsilon)}$	Rated MCRDR using a non-linear network and random initialisation.
$RM_{bp(\Delta)}$	Rated MCRDR using a non-linear network and the single-step-Δ-initialisation-rule.
Δ_{bp}	Initialisation rule for a two layered non-linear network
q	Index for the current *hidden* node in Δ_{bp}-rule.
RM_{RBF}	Rated MCRDR using an RBF network – only creating hidden nodes when a new input is added.
RM_{RBF+}	Rated MCRDR using an RBF network – creates hidden nodes when a new input is added and when needed. Also uses other techniques to aid learning.
t^a	The actual calculated total for a class in the Multi-Class-Prediction simulated expert.
t^m	The maximum total possible in the Multi-Class-Prediction simulated expert.
RM_p	Rated MCRDR applied to prudence analysis
$RM_{p(p)}$	Rated MCRDR using the prediction method for prudence analysis
$RM_{p(c)}$	Rated MCRDR using the classification method for prudence analysis
η_s	The shortcut connections *gain* term (often referred to as *learning rate*).
θ	A *threshold* value used in the RBF+ based technique. If the error exceeds this value, then an investigation should be done into whether a new hidden node should be added.
θ_t	The threshold at time *t* indicating what level the output must exceed to prevent a warning to be given.
θ_s	The threshold at the start of the knowledge base creation phase.

θ_η	The threshold training rate. Indicates the amount of change made to the threshold after each case.
θ_m	The minimum (class based method) or maximum (prediction based method) that the threshold can not exceed.

Appendix C: Acronyms

This Appendix lists all the acronyms used throughout this thesis in alphabetical order.

AA	Attribute Association
ADALINE	*Ada*ptive *Lin*ear Combiner
AI	Artificial Intelligence
ANN	Artificial Neural Network
ARFN	Adaptive Response Function Neurons
BP	Backpropagation
BPA	Backpropagation Algorithm
CA	Class Association
CBR	Case Based Reasoning
CC	Cascade Correlation
CMOD	*C*asual *Mod*elling
CPN	Counterpropagation Network
CQL	Concurrent-Q-Learning
DNF	Disjunctive Normal Form
DRDR	Dynamic RDR
DRPA	Decreasing Rule Path Association
ES	Expert System
ETS	Expertise Transfer System
FCA	Formal Concept Analysis
FMR	Fuzzy MCRDR
FN	False Negative
FP	False Positive
FRDR	Fuzzy RDR
FROCH	Fuzzy ROCH
GENICA	*Gen*eralization using *I*ntra *C*onstruction and *A*bsorption
GKB	Generic Knowledge Base
HBF	Hyper Basis Function
IF	Information Filtering
IONICS	Iterative Organisation of Novel Ion Chromatography Solutions

ISC	Intermediate Situation Cognition
KA	Knowledge Acquisition
KADS	Knowledge Acquisition and Design Structuring
KB	Knowledge Base
KBS	Knowledge Based System
KDD	Knowledge Discovery in Databases
KE	Knowledge Engineering
KM	Knowledge Maintenance
KR	Knowledge Representation
KSS0	Knowledge Support System Zero
LDRDR	Learning Dynamic RDR
MADALINE	Many ADALINE
MCRDR	Multiple Classification Ripple Down Rules
MCRDR/FCA	A hybrid Combination of MCRDR and FCA
MDLP	Minimum Description Length Principle
ML	Machine Learning
NRDR	Nested RDR
OKBC	Open Knowledge Base Connectivity
OL	Ontology Learning
ONCOCIN	*Onco*logic MY*CIN*
OODBMS	Object Oriented Database Management System
PCP	Personal Construct Psychology
PDP	Parallel Distributed Processing System
PEIRS	Pathology Expert Interpretative Reporting System
PRDR	Possible RDR
PRICAI	Pacific Rim International Conference in Artificial Intelligence
PSM	Problem Solving Method
PWIMS	Personalized Web Information Management System
RAN	Resource Allocating Network
RBF	Radial Basis Function
RDR	Ripple-Down Rules
REMAP	*Re*presentation and *Ma*intenance of *P*rocess knowledge
RL	Reinforcement Learning

RM	Rated MCRDR.
ROCH	RDR-Oriented Conceptual Hierarchies
RPA	Rule Path Association
RRDR	Recursive RDR
SC	Situation Cognition
SCRDR	Single Classification RDR
SITCRC	Smart Internet Technology Co-operative Research Centre
SQL	Structured Query Language
SVM	Support Vector Machine
TD	Temporal Difference Learning
TLU	Threshold Logic Unit
TN	True Negative
TP	True Positive
TRA	Terminating Rule Association
TTT	Tic-Tac-Toe
V&V	Validation and Verification
WM	Weighted MCRDR (old acronym now referred to as RM)
WWW	World Wide Web
XRDR	Simple RDR

Appendix D: Parameters used in Experiments

This Appendix lists all the necessary parameters used in each experiment throughout this thesis. The table below lists these parameters according to experiment in the order they appear in the thesis.

Experiment Description	Dataset	Algorithm	ζ	η	η_s	*Association Type*	p	k	σ	θ	θ_s	θ_r	θ_m	*# of expert conditions*
Classification Generalisation	Multi Linear	RM_w	**0.1**	-	-	**DRPA**	-	-	-	-	-	-	-	-
		$RM_{l(\varepsilon)}$	-	**0.1**	-	**RPA**	-	**1.0**	-	-	-	-	-	-
		$RM_{l(\Delta)}$	**0.1**	**0.1**	-	**DRPA**	-	**1.0**	-	-	-	-	-	-
		$RM_{bp(\varepsilon)}$	-	**0.1**	-	**RPA**	**3.0**	**1.0**	-	-	-	-	-	-
		$RM_{bp(\Delta)}$	**0.1**	**0.1**	**0.1**	**DRPA**	**1.0**	**1.0**	-	-	-	-	-	-
		RM_{rbf}	-	**0.1**	-	**RPA**	-	**1.0**	**15**	-	-	-	-	-
		RM_{rbf+}	-	**0.1**	-	**RPA**	-	**1.0**	**13**	**0.3**	-	-	-	-
		MCRDR	-	-	-	-	-	-	-	-	-	-	-	-
		ANN	-	**0.1**	-	-	**40***	**1.0**	-	-	-	-	-	-
	Multi Non-Linear	RM_w	**0.1**	-	-	**DRPA**	-	-	-	-	-	-	-	-
		$RM_{l(\varepsilon)}$	-	**0.1**	-	**RPA**	-	**1.0**	-	-	-	-	-	-
		$RM_{l(\Delta)}$	**0.1**	**0.1**	-	**DRPA**	-	**1.0**	-	-	-	-	-	-
		$RM_{bp(\varepsilon)}$	-	**0.1**	-	**RPA**	**3.0**	**1.0**	-	-	-	-	-	-
		$RM_{bp(\Delta)}$	**0.1**	**0.1**	**0.1**	**DRPA**	**1.0**	**1.0**	-	-	-	-	-	-
		RM_{rbf}	-	**0.1**	-	**RPA**	-	**1.0**	**15**	-	-	-	-	-
		RM_{rbf+}	-	**0.1**	-	**RPA**	-	**1.0**	**13**	**0.3**	-	-	-	-
		MCRDR	-	-	-	-	-	-	-	-	-	-	-	-
		ANN	-	**0.1**	-	-	**40***	**1.0**	-	-	-	-	-	-
	Chess	RM_w	**0.1**	-	-	**TRA**	-	-	-	-	-	-	-	**2**
		$RM_{l(\varepsilon)}$	-	**0.1**	-	**TRA**	-	**1.0**	-	-	-	-	-	**2**
		$RM_{l(\Delta)}$	**0.1**	**0.1**	-	**TRA**	-	**1.0**	-	-	-	-	-	**2**
		$RM_{bp(\varepsilon)}$	-	**0.1**	-	**TRA**	**1.0**	**1.0**	-	-	-	-	-	**2**
		$RM_{bp(\Delta)}$	**0.1**	**0.1**	**0.1**	**TRA**	**1.0**	**1.0**	-	-	-	-	-	**2**
		RM_{rbf}	-	**0.1**	-	**TRA**	-	**1.0**	**2**	-	-	-	-	**2**
		RM_{rbf+}	-	**0.1**	-	**TRA**	-	**1.0**	**2**	**0.3**	-	-	-	**2**
		MCRDR	-	-	-	-	-	-	-	-	-	-	-	**2**
		ANN	-	**0.05**	-	-	**300***	**1.0**	-	-	-	-	-	-
	TTT	RM_w	**1.0**	-	-	**TRA**	-	-	-	-	-	-	-	**3**
		$RM_{l(\varepsilon)}$	-	**0.2**	-	**TRA**	-	**1.0**	-	-	-	-	-	**3**
		$RM_{l(\Delta)}$	**1.0**	**0.2**	-	**TRA**	-	**1.0**	-	-	-	-	-	**3**
		$RM_{bp(\varepsilon)}$	-	**0.2**	-	**TRA**	**1.0**	**1.0**	-	-	-	-	-	**3**
		$RM_{bp(\Delta)}$	**1.0**	**0.2**	**0.2**	**TRA**	**1.0**	**1.0**	-	-	-	-	-	**3**
		RM_{rbf}	-	**0.2**	-	**TRA**	-	**1.0**	**2**	-	-	-	-	**3**
		RM_{rbf+}	-	**0.2**	-	**TRA**	-	**1.0**	**2**	**0.3**	-	-	-	**3**
		MCRDR	-	-	-	-	-	-	-	-	-	-	-	**3**
		ANN	-	**0.1**	-	-	**15***	**1.0**	-	-	-	-	-	-

* *This represents the number of hidden nodes used in the network on this task. It does not represent the hidden percentage like the other values in the same column.*

Experiment Description	Dataset	Algorithm	ζ	η	η_s	*Association Type*	p	k	σ	θ	θ_s	θ_r	θ_m	*# of expert conditions*
Classification Generalisation	Nursery	RM_w	**0.6**	-	-	**RPA**	-	-	-	-	-	-	-	-
		$RM_{l(\varepsilon)}$	-	**0.2**	-	**RPA**	-	**1.0**	-	-	-	-	-	-
		$RM_{l(\Delta)}$	**0.6**	**0.2**	-	**RPA**	-	**1.0**	-	-	-	-	-	-
		$RM_{bp(\varepsilon)}$	-	**0.2**	-	**RPA**	**1.0**	**1.0**	-	-	-	-	-	-
		$RM_{bp(\Delta)}$	**0.6**	**0.2**	**0.2**	**RPA**	**1.0**	**1.0**	-	-	-	-	-	-
		RM_{rbf}	-	**0.1**	-	**TRA**	-	**1.0**	**2**	-	-	-	-	-
		RM_{rbf+}	-	**0.1**	-	**TRA**	-	**1.0**	**2**	**0.2**	-	-	-	-
		MCRDR	-	-	-	-	-	-	-	-	-	-	-	-
		ANN	-	**0.1**	-	-	**100***	**1.0**	-	-	-	-	-	-
	Audiology	RM_w	**0.5**	-	-	**TRA**	-	-	-	-	-	-	-	-
		$RM_{l(\varepsilon)}$	-	**0.01**	-	**TRA**	-	**1.0**	-	-	-	-	-	-
		$RM_{l(\Delta)}$	**0.5**	**0.01**	-	**TRA**	-	**1.0**	-	-	-	-	-	-
		$RM_{bp(\varepsilon)}$	-	**0.01**	-	**TRA**	**2.0**	**1.0**	-	-	-	-	-	-
		$RM_{bp(\Delta)}$	**0.5**	**0.01**	**0.01**	**TRA**	**2.0**	**1.0**	-	-	-	-	-	-
		RM_{rbf}	-	**0.1**	-	**TRA**	-	**1.0**	**3**	-	-	-	-	-
		RM_{rbf+}	-	**0.1**	-	**TRA**	-	**1.0**	**3**	**0.2**	-	-	-	-
		MCRDR	-	-	-	-	-	-	-	-	-	-	-	-
		ANN	-	**0.01**	-	-	**1***	**1.0**	-	-	-	-	-	-
	Car	RM_w	**0.45**	-	-	**TRA**	-	-	-	-	-	-	-	**2**
		$RM_{l(\varepsilon)}$	-	**0.2**	-	**TRA**	-	**1.0**	-	-	-	-	-	**2**
		$RM_{l(\Delta)}$	**0.45**	**0.2**	-	**TRA**	-	**1.0**	-	-	-	-	-	**2**
		$RM_{bp(\varepsilon)}$	-	**0.2**	-	**TRA**	**3.0**	**1.0**	-	-	-	-	-	**2**
		$RM_{bp(\Delta)}$	**0.45**	**0.2**	**0.2**	**TRA**	**3.0**	**1.0**	-	-	-	-	-	**2**
		RM_{rbf}	-	**0.1**	-	**TRA**	-	**1.0**	**3**	-	-	-	-	**2**
		RM_{rbf+}	-	**0.1**	-	**TRA**	-	**1.0**	**3**	**0.2**	-	-	-	**2**
		MCRDR	-	-	-	-	-	-	-	-	-	-	-	**2**
		ANN	-	**0.1**	-	-	**50***	**1.0**	-	-	-	-	-	-
Classification Online	Multi Non-Linear	RM_w	**0.1**	**-**	**-**	**DRPA**	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**-**
		$RM_{l(\varepsilon)}$	**-**	**0.1**	**-**	**RPA**	**-**	**1.0**	**-**	**-**	**-**	**-**	**-**	**-**
		$RM_{l(\Delta)}$	**0.1**	**0.1**	**-**	**DRPA**	**-**	**1.0**	**-**	**-**	**-**	**-**	**-**	**-**
		$RM_{bp(\varepsilon)}$	**-**	**0.1**	**-**	**RPA**	**3.0**	**1.0**	**-**	**-**	**-**	**-**	**-**	**-**
		$RM_{bp(\Delta)}$	**0.1**	**0.1**	**0.1**	**DRPA**	**1.0**	**1.0**	**-**	**-**	**-**	**-**	**-**	**-**
		RM_{rbf}	**-**	**0.1**	**-**	**RPA**	**-**	**1.0**	**15**	**-**	**-**	**-**	**-**	**-**
		RM_{rbf+}	**-**	**0.1**	**-**	**RPA**	**-**	**1.0**	**13**	**0.3**	**-**	**-**	**-**	**-**
		MCRDR	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**-**
		ANN	**-**	**0.1**	**-**	**-**	**40***	**1.0**	**-**	**-**	**-**	**-**	**-**	**-**
	Chess	RM_w	**0.1**	**-**	**-**	**TRA**	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**2**
		$RM_{l(\varepsilon)}$	-	**0.1**	-	**TRA**	-	**1.0**	-	-	-	-	-	**2**
		$RM_{l(\Delta)}$	**0.1**	**0.1**	-	**TRA**	-	**1.0**	-	-	-	-	-	**2**
		$RM_{bp(\varepsilon)}$	-	**0.1**	-	**TRA**	**1.0**	**1.0**	-	-	-	-	-	**2**
		$RM_{bp(\Delta)}$	**0.1**	**0.1**	**0.1**	**TRA**	**1.0**	**1.0**	-	-	-	-	-	**2**
		RM_{rbf}	-	**0.1**	-	**TRA**	-	**1.0**	**2**	-	-	-	-	**2**
		RM_{rbf+}	-	**0.1**	-	**TRA**	-	**1.0**	**2**	**0.3**	-	-	-	**2**
		MCRDR	-	-	-	-	-	-	-	-	-	-	-	**2**
		ANN	-	**0.05**	-	-	**300***	**1.0**	-	-	-	-	-	-

Experiment Description	Dataset	Algorithm	ζ	η	η_s	*Association Type*	p	k	σ	θ	θ_s	θ_r	θ_m	*# of expert conditions*
Classification Online	TTT	RM_w	1.0	-	-	TRA	-	-	-	-	-	-	-	3
		$RM_{l(\varepsilon)}$	-	0.2	-	TRA	-	1.0	-	-	-	-	-	3
		$RM_{l(\Delta)}$	1.0	0.2	-	TRA	-	1.0	-	-	-	-	-	3
		$RM_{bp(\varepsilon)}$	-	0.2	-	TRA	1.0	1.0	-	-	-	-	-	3
		$RM_{bp(\Delta)}$	1.0	0.2	0.2	TRA	1.0	1.0	-	-	-	-	-	3
		RM_{rbf}	-	0.2	-	TRA	-	1.0	2	-	-	-	-	3
		RM_{rbf+}	-	0.2	-	TRA	-	1.0	2	0.3	-	-	-	3
		MCRDR	-	-	-	-	-	-	-	-	-	-	-	3
		ANN	-	0.1	-	-	15*	1.0	-	-	-	-	-	-
	Nursery	RM_w	0.6	-	-	RPA	-	-	-	-	-	-	-	-
		$RM_{l(\varepsilon)}$	-	0.2	-	RPA	-	1.0	-	-	-	-	-	-
		$RM_{l(\Delta)}$	0.6	0.2	-	RPA	-	1.0	-	-	-	-	-	-
		$RM_{bp(\varepsilon)}$	-	0.2	-	RPA	1.0	1.0	-	-	-	-	-	-
		$RM_{bp(\Delta)}$	0.6	0.2	0.2	RPA	1.0	1.0	-	-	-	-	-	-
		RM_{rbf}	-	0.1	-	TRA	-	1.0	2	-	-	-	-	-
		RM_{rbf+}	-	0.1	-	TRA	-	1.0	2	0.2	-	-	-	-
		MCRDR	-	-	-	-	-	-	-	-	-	-	-	-
		ANN	-	0.1	-	-	100*	1.0	-	-	-	-	-	-
	Audiology	RM_w	0.5	-	-	TRA	-	-	-	-	-	-	-	-
		$RM_{l(\varepsilon)}$	-	0.01	-	TRA	-	1.0	-	-	-	-	-	-
		$RM_{l(\Delta)}$	0.5	0.01	-	TRA	-	1.0	-	-	-	-	-	-
		$RM_{bp(\varepsilon)}$	-	0.01	-	TRA	2.0	1.0	-	-	-	-	-	-
		$RM_{bp(\Delta)}$	0.5	0.01	0.01	TRA	2.0	1.0	-	-	-	-	-	-
		RM_{rbf}	-	0.1	-	TRA	-	1.0	3	-	-	-	-	-
		RM_{rbf+}	-	0.1	-	TRA	-	1.0	3	0.2	-	-	-	-
		MCRDR	-	-	-	-	-	-	-	-	-	-	-	-
		ANN	-	0.01	-	-	1*	1.0	-	-	-	-	-	-
	Car	RM_w	0.45	-	-	TRA	-	-	-	-	-	-	-	2
		$RM_{l(\varepsilon)}$	-	0.2	-	TRA	-	1.0	-	-	-	-	-	2
		$RM_{l(\Delta)}$	0.45	0.2	-	TRA	-	1.0	-	-	-	-	-	2
		$RM_{bp(\varepsilon)}$	-	0.2	-	TRA	3.0	1.0	-	-	-	-	-	2
		$RM_{bp(\Delta)}$	0.45	0.2	0.2	TRA	3.0	1.0	-	-	-	-	-	2
		RM_{rbf}	-	0.1	-	TRA	-	1.0	3	-	-	-	-	2
		RM_{rbf+}	-	0.1	-	TRA	-	1.0	3	0.2	-	-	-	2
		MCRDR	-	-	-	-	-	-	-	-	-	-	-	2
		ANN	-	0.1	-	-	50*	1.0	-	-	-	-	-	-
Prediction Generalisation	Multi Prediction	RM_w	0.8	-	-	TRA	-	-	-	-	-	-	-	-
		$RM_{l(\varepsilon)}$	-	0.1	-	TRA	-	1.0	-	-	-	-	-	-
		$RM_{l(\Delta)}$	0.2	0.1	-	TRA	-	1.0	-	-	-	-	-	-
		$RM_{bp(\varepsilon)}$	-	0.1	-	TRA	3.0	1.0	-	-	-	-	-	-
		$RM_{bp(\Delta)}$	0.4	0.1	0.1	TRA	3.0	1.0	-	-	-	-	-	-
		RM_{rbf}	-	0.1	-	TRA	-	1.0	11	-	-	-	-	-
		RM_{rbf+}	-	0.1	-	TRA	-	1.0	12	0.2	-	-	-	-
		ANN	-	0.1	-	-	40*	1.0	-	-	-	-	-	-

Experiment Description	Dataset	Algorithm	ζ	η	η_s	*Association Type*	p	k	σ	θ	θ_s	θ_r	θ_m	*# of expert conditions*
Prediction Online	Multi Prediction	RM_w	**0.8**	-	-	**TRA**	-	-	-	-	-	-	-	-
		$RM_{l(\varepsilon)}$	-	**0.1**	-	**TRA**	-	**1.0**	-	-	-	-	-	-
		$RM_{l(\Delta)}$	**0.2**	**0.1**	-	**TRA**	-	**1.0**	-	-	-	-	-	-
		$RM_{bp(\varepsilon)}$	-	**0.1**	-	**TRA**	**3.0**	**1.0**	-	-	-	-	-	-
		$RM_{bp(\Delta)}$	**0.4**	**0.1**	**0.1**	**TRA**	**3.0**	**1.0**	-	-	-	-	-	-
		RM_{rbf}	-	**0.1**	-	**TRA**	-	**1.0**	**11**	-	-	-	-	-
		RM_{rbf+}	-	**0.1**	-	**TRA**	-	**1.0**	**12**	**0.2**	-	-	-	-
		ANN	-	**0.1**	-	-	**40***	**1.0**	-	-	-	-	-	-
Prudence	GARVAN	$RM_{p(p)}$	**0.61**	**0.2**	**0.2**	**DRPA**	**1.0**	**1.0**	-	-	**0.01**	**0.001**	**0.1**	**2**
		$RM_{p(c)}$	**0.2**	**0.2**	**0.2**	**RPA**	**2.0**	**1.0**	-	-	**0.4**	**0.0001**	**0.0**	**2**
		$RM_{p(c)}$	**0.2**	**0.2**	**0.2**	**RPA**	**2.0**	**1.0**	-	-	**0.4**	**0.0003**	**0.0**	**2**
		$RM_{p(c)}$	**0.2**	**0.2**	**0.2**	**RPA**	**2.0**	**1.0**	-	-	**0.4**	**0.0005**	**0.0**	**2**
		$RM_{p(c)}$	**0.2**	**0.2**	**0.2**	**RPA**	**2.0**	**1.0**	-	-	**0.4**	**0.0007**	**0.0**	**2**
		$RM_{p(c)}$	**0.2**	**0.2**	**0.2**	**RPA**	**2.0**	**1.0**	-	-	**0.4**	**0.001**	**0.0**	**2**
		$RM_{p(c)}$	**0.2**	**0.2**	**0.2**	**RPA**	**2.0**	**1.0**	-	-	**0.4**	**0.002**	**0.0**	**2**
		$RM_{p(c)}$	**0.2**	**0.2**	**0.2**	**RPA**	**2.0**	**1.0**	-	-	**0.4**	**0.003**	**0.0**	**2**
		$RM_{p(c)}$	**0.2**	**0.2**	**0.2**	**RPA**	**2.0**	**1.0**	-	-	**0.4**	**0.005**	**0.0**	**2**
		$RM_{p(c)}$	**0.2**	**0.2**	**0.2**	**RPA**	**2.0**	**1.0**	-	-	**0.4**	**0.01**	**0.0**	**2**
		$RM_{p(c)}$	**0.2**	**0.2**	**0.2**	**RPA**	**2.0**	**1.0**	-	-	**0.4**	**0.03**	**0.0**	**2**
	Multi	$RM_{p(p)}$	**0.5**	**0.15**	**0.15**	**TRA**	**2.0**	**1.0**	-	-	**-0.1**	**0.005**	**0.49**	-
		$RM_{p(c)}$	**0.2**	**0.2**	**0.2**	**DRPA**	**2.0**	**1.0**	-	-	**0.4**	**0.00001**	**0.0**	-
		$RM_{p(c)}$	**0.2**	**0.2**	**0.2**	**DRPA**	**2.0**	**1.0**	-	-	**0.4**	**0.0001**	**0.0**	-
		$RM_{p(c)}$	**0.2**	**0.2**	**0.2**	**DRPA**	**2.0**	**1.0**	-	-	**0.4**	**0.001**	**0.0**	-
		$RM_{p(c)}$	**0.2**	**0.2**	**0.2**	**DRPA**	**2.0**	**1.0**	-	-	**0.4**	**0.003**	**0.0**	-
		$RM_{p(c)}$	**0.2**	**0.2**	**0.2**	**DRPA**	**2.0**	**1.0**	-	-	**0.4**	**0.01**	**0.0**	-
		$RM_{p(c)}$	**0.2**	**0.2**	**0.2**	**DRPA**	**2.0**	**1.0**	-	-	**0.4**	**0.03**	**0.0**	-
		$RM_{p(c)}$	**0.2**	**0.2**	**0.2**	**DRPA**	**2.0**	**1.0**	-	-	**0.4**	**0.05**	**0.0**	-
		$RM_{p(c)}$	**0.2**	**0.2**	**0.2**	**DRPA**	**2.0**	**1.0**	-	-	**0.4**	**0.07**	**0.0**	-
		$RM_{p(c)}$	**0.2**	**0.2**	**0.2**	**DRPA**	**2.0**	**1.0**	-	-	**0.4**	**0.1**	**0.0**	-
		$RM_{p(c)}$	**0.2**	**0.2**	**0.2**	**DRPA**	**2.0**	**1.0**	-	-	**0.4**	**0.5**	**0.0**	-
	Chess	$RM_{p(p)}$	**0.7**	**0.25**	**0.25**	**TRA**	**1.0**	**1.0**	-	-	**0.04**	**0.005**	**0.2**	**2**
		$RM_{p(c)}$	**0.3**	**0.2**	**0.2**	**DRPA**	**2.0**	**1.0**	-	-	**0.4**	**0.001**	**0.0**	**2**
		$RM_{p(c)}$	**0.3**	**0.2**	**0.2**	**DRPA**	**2.0**	**1.0**	-	-	**0.4**	**0.003**	**0.0**	**2**
		$RM_{p(c)}$	**0.3**	**0.2**	**0.2**	**DRPA**	**2.0**	**1.0**	-	-	**0.4**	**0.008**	**0.0**	**2**
		$RM_{p(c)}$	**0.3**	**0.2**	**0.2**	**DRPA**	**2.0**	**1.0**	-	-	**0.4**	**0.012**	**0.0**	**2**
		$RM_{p(c)}$	**0.3**	**0.2**	**0.2**	**DRPA**	**2.0**	**1.0**	-	-	**0.4**	**0.014**	**0.0**	**2**
		$RM_{p(c)}$	**0.3**	**0.2**	**0.2**	**DRPA**	**2.0**	**1.0**	-	-	**0.4**	**0.018**	**0.0**	**2**
		$RM_{p(c)}$	**0.3**	**0.2**	**0.2**	**DRPA**	**2.0**	**1.0**	-	-	**0.4**	**0.02**	**0.0**	**2**
		$RM_{p(c)}$	**0.3**	**0.2**	**0.2**	**DRPA**	**2.0**	**1.0**	-	-	**0.4**	**0.025**	**0.0**	**2**
		$RM_{p(c)}$	**0.3**	**0.2**	**0.2**	**DRPA**	**2.0**	**1.0**	-	-	**0.4**	**0.03**	**0.0**	**2**
		$RM_{p(c)}$	**0.3**	**0.2**	**0.2**	**DRPA**	**2.0**	**1.0**	-	-	**0.4**	**0.05**	**0.0**	**2**
	TTT	$RM_{p(p)}$	**0.6**	**0.2**	**0.2**	**DRPA**	**2.0**	**1.0**	-	-	**0.01**	**0.02**	**0.24**	**2**
		$RM_{p(c)}$	**0.05**	**0.2**	**0.2**	**DRPA**	**3.0**	**1.0**	-	-	**0.4**	**0.004**	**0.0**	**2**
		$RM_{p(c)}$	**0.05**	**0.2**	**0.2**	**DRPA**	**3.0**	**1.0**	-	-	**0.4**	**0.006**	**0.0**	**2**
		$RM_{p(c)}$	**0.05**	**0.2**	**0.2**	**DRPA**	**3.0**	**1.0**	-	-	**0.4**	**0.008**	**0.0**	**2**
		$RM_{p(c)}$	**0.05**	**0.2**	**0.2**	**DRPA**	**3.0**	**1.0**	-	-	**0.4**	**0.009**	**0.0**	**2**
		$RM_{p(c)}$	**0.05**	**0.2**	**0.2**	**DRPA**	**3.0**	**1.0**	-	-	**0.4**	**0.01**	**0.0**	**2**
		$RM_{p(c)}$	**0.05**	**0.2**	**0.2**	**DRPA**	**3.0**	**1.0**	-	-	**0.4**	**0.015**	**0.0**	**2**
		$RM_{p(c)}$	**0.05**	**0.2**	**0.2**	**DRPA**	**3.0**	**1.0**	-	-	**0.4**	**0.02**	**0.0**	**2**
		$RM_{p(c)}$	**0.05**	**0.2**	**0.2**	**DRPA**	**3.0**	**1.0**	-	-	**0.4**	**0.03**	**0.0**	**2**
		$RM_{p(c)}$	**0.05**	**0.2**	**0.2**	**DRPA**	**3.0**	**1.0**	-	-	**0.4**	**0.05**	**0.0**	**2**
		$RM_{p(c)}$	**0.05**	**0.2**	**0.2**	**DRPA**	**3.0**	**1.0**	-	-	**0.4**	**0.1**	**0.0**	**2**

Experiment Description	Dataset	Algorithm	ζ	η	η_s	*Association Type*	p	k	σ	θ	θ_s	θ_r	θ_m	*# of expert conditions*
Prudence	eHealth	$RM_{p(p)}$	**0.1**	**0.2**	**0.2**	**TRA**	**3.5**	**1.0**	-	-	**0.0001**	**0.0002**	**0.4**	*
		$RM_{p(c)}$	**0.3**	**0.2**	**0.2**	**TRA**	**3.5**	**1.0**	-	-	**0.4**	**0.0001**	**0.0**	*
		$RM_{p(c)}$	**0.3**	**0.2**	**0.2**	**TRA**	**3.5**	**1.0**	-	-	**0.4**	**0.0005**	**0.0**	*
		$RM_{p(c)}$	**0.3**	**0.2**	**0.2**	**TRA**	**3.5**	**1.0**	-	-	**0.4**	**0.001**	**0.0**	*
		$RM_{p(c)}$	**0.3**	**0.2**	**0.2**	**TRA**	**3.5**	**1.0**	-	-	**0.4**	**0.002**	**0.0**	*
		$RM_{p(c)}$	**0.3**	**0.2**	**0.2**	**TRA**	**3.5**	**1.0**	-	-	**0.4**	**0.004**	**0.0**	*
		$RM_{p(c)}$	**0.3**	**0.2**	**0.2**	**TRA**	**3.5**	**1.0**	-	-	**0.4**	**0.0045**	**0.0**	*
		$RM_{p(c)}$	**0.3**	**0.2**	**0.2**	**TRA**	**3.5**	**1.0**	-	-	**0.4**	**0.007**	**0.0**	*
		$RM_{p(c)}$	**0.3**	**0.2**	**0.2**	**TRA**	**3.5**	**1.0**	-	-	**0.4**	**0.01**	**0.0**	*
		$RM_{p(c)}$	**0.3**	**0.2**	**0.2**	**TRA**	**3.5**	**1.0**	-	-	**0.4**	**0.05**	**0.0**	*
		$RM_{p(c)}$	**0.3**	**0.2**	**0.2**	**TRA**	**3.5**	**1.0**	-	-	**0.4**	**0.1**	**0.0**	*

* *The number of expert conditions is covered by how many must be selected to reach the next non-null conclusion.*