Adaptive Knowledge Discovery Techniques for Data Mining

Qing Qing Zhou

A thesis submitted for the degree of Doctor of Philosophy at the University of Otago, Dunedin New Zealand

July 2003
Abstract

The ease of collection and the increasing availability of large data stores has led to demands for improved methods for analyzing these data and deriving significant knowledge that may be latent in these data stores. In particular there is hope that the use of new analytical techniques in connection with “data trawling”, or data mining operations may reveal hidden relationships that lie buried within these data sets. This research investigates various techniques for the task of discovering relevant features and inference rules from data sets. Following the three steps of a knowledge extraction process, namely pre-processing (feature selection), rule discovery process, and post-processing (rule refinement), the research attempts to address some current difficulties in these three steps and introduces and integrates a ‘market trading’ technique with existing techniques from the field of knowledge discovery and refinement with respect to data mining.

In connection with the pre-processing, a feature selection approach that employs neural networks is presented, and three associated pruning schemes that make automatic selection of the pruning threshold are proposed. The proposed neural network techniques are evaluated and compared with the $\chi^2$-statistic-based discretization algorithm, called Chi2, by experimenting with six practical applications.

The Chi2 algorithm is investigated as a technique for solving problems in intelligent spatial information systems and fuzzy systems. The case studies show that the Chi2-based spatial data filtering can successfully reduce the number of spatial data items and the number of features, and therefore neural network computation can be efficiently performed. A novel approach of employing the Chi2 algorithm to select membership functions for fuzzy systems is proposed. In connection with the applications of fuzzy neural networks (FuNN models), three experimental examinations are demonstrated that an automatic selection of the number and widths of the membership functions by the Chi2-based membership function selection method can lead to the improvement of the generalization ability of FuNN fuzzy neural networks.

In connection with the rule discovery and refinement process, a novel market-based rule learning (MBRL) system is developed and its capability of evolving and refining rules is investigated. As a classifier system-inspired model, it introduces a novel element by importing existing rule sets
generated by other rule extraction techniques into the system. This basic change not only makes the MBRL system begin with pre-established rule sets with a relatively limited complexity, rather than a random set, but also enhances the likelihood of being able to interpret the evolved rules. Moreover, the MBRL system produces various modifications in each of the layers of the structure. With the modifications introduced by the MBRL system, the problems existing in current classifier systems can be solved or lessened.

In this research, the MBRL system is proposed as a post-processing tool to be used with fuzzy neural networks (FuNN models) and the fuzzy neural network rule extraction technique, ReFuNN, in order to provide a general framework for fuzzy inference-based rule discovery. Similarly, as a post-processing tool, the MBRL system is also proposed to be used with feed-forward neural networks, and the feed-forward neural network rule extraction technique, NeuroLinear, in order to improve the quality of extracted rules from feed-forward neural networks. The experimental results show that the MBRL system is a potentially useful additional tool that can be used to refine (fuzzy) neural network extracted rules and possibly discover and add some new, better performance rules. As a result, it can lead to improved performance by increasing the accuracy of the rule inference performance and/or improving the comprehensibility of the rules.

By illustrating how the MBRL system succeeded in finding solutions for six learning examples from scratch, the MBRL system is shown to have potential as an alternative generic learning technique that can be used to complement, or be used as an alternative to, conventional connectionist models to accomplish complex computational tasks.
Acknowledgements

I would like to especially thank my supervisor, Professor Martin Purvis, for his vigilant supervision, his constant encouragement, and his friendship. He spent countless hours reviewing my work and provided me many insightful comments and constructive criticism. Without his enthusiasm and intelligence, this work would not have been possible. Over years, Martin has guided me into the scientific world, and taught me not only how to do research, but also have a balanced life. As a good friend, he has helped me to improve my English and offered me a great deal of help in many ways especially during my earlier days of life in New Zealand.

I would like to thank my co-supervisors, Dr. Stephen Cranefield and Professor Nikola Kasabov, for their continuing support, encouragement, and helpful comments. Thanks should go to Professor van den Herik - University of Maastricht; Dr. David Tuck - Industrial Research Limited; Professor George Benwell and Dr. Peter Whigham - University of Otago for reviewing and making suggestions to improve this thesis.

I would like to thank Mariusz Nowostawski for his valuable discussion and advice, and generously supplying the source code of Genetic Algorithms for my research. I would also like to thank Daniel Carter for proof-reading this thesis. Thanks should go to Dr. Simon Hales of Wellington School of Medicine, Dr. Corey Bradshaw of University of Tasmania, and Darrin Drumm of University of Otago, for providing the practical data sets for my experiments. I also appreciate the constant and prompt system support from Brendon Sly, Grame Roxburgh and Peter George.

The members of the Software Engineering and Collaborative Modelling laboratory have played an important role in my social life. I value the great friendship from Dr. Feng Zhang, Dr. Xiaodong Li, Hong shun Xu, Xiaoping Gong, Aurora Diaz, Rumen Raykov, Geoff Bush, Roy Ward, David June, Peter Hwang, Marcos De Oliveira, Simon Zwarts, and Gina Pan.

Last, and most important, I would like to thank my parents, my husband, Larry, and my baby son, Danny. Without the love, patience, support, and joy that they have provided, I would never be possible to complete this thesis.
# Table of Contents

Abstract .................................................. ii
Acknowledgements ........................................ iv
List of Figures ........................................... xi
List of Tables ............................................. xiii

## Part I Introduction and Background

### 1 Introduction ........................................ 1
   1.1 Motivation ........................................ 1
   1.2 Research Goals ................................... 6
   1.3 Evaluation Criteria ............................... 8
      1.3.1 Evaluation Criteria for Feature Selection Techniques ................. 8
      1.3.2 Evaluation Criteria for Rule Extraction and Refinement Techniques .... 9
   1.4 Outline of Thesis ................................ 11

### 2 Background .......................................... 14
   2.1 Introduction ...................................... 14
   2.2 Statistical Methodology ........................... 14
      2.2.1 Estimation .................................. 14
      2.2.2 Hypothesis Testing .......................... 15
   2.3 Neural Networks and Fuzzy Systems ................. 16
      2.3.1 Feed-forward Neural Networks .................. 17
      2.3.2 Fuzzy Systems ................................ 23
      2.3.3 Membership Function Construction Methods in Fuzzy Systems .......... 30
      2.3.4 Fuzzy Neural Networks ......................... 32
   2.4 Feature Selection Methods ........................ 35
      2.4.1 Overview ................................... 36
      2.4.2 A $\chi^2$ Statistic-based Discretization Algorithm - the Chi2 Algorithm ... 39
   2.5 Rule Extraction Methods ........................... 41
      2.5.1 Overview ................................... 41
      2.5.2 Decision Trees ................................ 43
2.5.3 X2R: A Fast Rule Generator .............................................. 45
2.5.4 NeuroLinear Approach .................................................. 46
2.5.5 ReFuNN Approach ......................................................... 50
2.6 Classifier Systems and Genetic Algorithms .............................. 54
  2.6.1 What is a Classifier System? ........................................ 55
  2.6.2 Genetic Algorithms ..................................................... 59
  2.6.3 Overview of Classifier Systems .................................... 64
  2.6.4 Limitations of Classifier Systems .................................. 67
2.7 Summary ................................................................. 69

3 The Problem Domains ..................................................... 70
  3.1 Introduction .............................................................. 70
  3.2 Iris Classification Data ................................................ 70
  3.3 Pima Indians Diabetes Data ........................................... 71
  3.4 Wine Recognition Data ................................................ 72
  3.5 Golf Course Problem ..................................................... 73
  3.6 Cook Islands Sea Cucumber Habitat Data ............................ 75
  3.7 New Zealand Asthma Incidence Data ................................ 76
  3.8 Data Preparation ........................................................ 78
  3.9 Summary ................................................................. 82

Part II Feature Selection

4 Feature Selection and Neural Network Analysis .......................... 83
  4.1 Introduction .............................................................. 83
  4.2 Feature Selection via Neural Networks ................................ 83
    4.2.1 Neural Network Training ......................................... 83
    4.2.2 Neural Network Pruning ......................................... 85
    4.2.3 Method Illustration: Iris Classification Data .................. 90
  4.3 Experiments .............................................................. 95
    4.3.1 Pima Indians Diabetes Data ..................................... 96
    4.3.2 Wine Recognition Data .......................................... 97
    4.3.3 Golf Course Problem ............................................. 99
Part III Rule Extraction and Refinement

6 A Market-based Rule Learning System .................................................. 132

6.1 Introduction ................................................................. 132

6.2 The Layers of the System .................................................. 133

6.2.1 Rule and Message System .......................................... 134

6.2.2 Apportionment of Credit Algorithm .............................. 134

6.2.3 The Genetic Algorithm ............................................. 138

6.3 The Properties of the System ........................................... 143

6.4 Analysis of the Behaviour of the System .............................. 146

6.4.1 Steady-state Behaviour ............................................. 146

6.4.2 Starting Parameters ................................................ 151

6.4.3 Time Complexity .................................................... 152

6.5 Summary ................................................................. 152
7 Rule Evolution and Refinement Using
the Market-based Rule Learning System ........................................ 154

7.1 Introduction ............................................................................. 154

7.2 Market-based Rule Evolution and Refinement Based on Extracted Rules
from Fuzzy Neural Networks ..................................................... 154

7.2.1 Fuzzy Rule and Input Information Encoding ....................... 155

7.2.2 Evolutionary Learning of Fuzzy Rules ................................. 158

7.2.3 Method Illustration: Iris Classification Data ...................... 160

7.3 Market-based Rule Evolution and Refinement Based on Extracted Rules
from Feed-forward Neural Networks ....................................... 163

7.3.1 NeuroLinear Rule Encoding ............................................. 164

7.3.2 Evolutionary Learning of NeuroLinear Rules ................... 166

7.3.3 Genetic Search within the System ................................. 169

7.3.4 Method Illustration: Iris Classification Data ...................... 171

7.4 Summary .............................................................................. 177

8 Experimental Implementation and Evaluation ............................. 179

8.1 Introduction ............................................................................. 179

8.2 Methodology ......................................................................... 179

8.3 Experiments with Existing Rule Extraction Methods .............. 181

8.3.1 Iris Classification ................................................................. 181

8.3.2 Pima Indians Diabetes Data ............................................. 183

8.3.3 Wine Recognition Data ...................................................... 184

8.3.4 Golf Course Problem ......................................................... 185

8.3.5 Cook Islands Sea Cucumber Habitat Data ....................... 187

8.3.6 New Zealand Asthma Incidence Data ............................. 188

8.4 Experiments with Market-based Rule Learning System .......... 189

8.4.1 Iris Classification ................................................................. 190

8.4.2 Pima Indians Diabetes Data ............................................. 191

8.4.3 Wine Recognition Data ...................................................... 194

8.4.4 Golf Course Problem ......................................................... 196
Part IV Conclusions

9 Conclusions ................................................................. 217
  9.1 Summary ................................................................. 217
  9.2 Main Contributions .................................................. 219
  9.3 Limitations ............................................................. 222
  9.4 Suggestions for Future Work ....................................... 223

Bibliography ...................................................................... 227

Attached CD:

Appendix A  Rules Extracted for the Six Problem Domains by the C4.5rules Method
Appendix B  Rules Extracted for the Six Problem Domains by the Rule Generator X2R
Appendix C  Fuzzy Rules Extracted for the Six Problem Domains by the ReFuNN Approach
Appendix D  Rules Extracted for the Six Problem Domains by the NeuroLinear Approach
Appendix E  Refined Fuzzy Rules for the Six Problems Domain by the Market-based Rule Learning System
Appendix F  Refined NeuroLinear Rules for the Six Problem Domains by the Market-based Rule Learning System
Appendix G  Evolved Rules for the Six Problem Domains by the Market-based Rule Learning System Starting with No Prior Knowledge
Appendix H  Parameters Used in the Experiments of Market-based Rule Evolution based on the ReFuNN-generated Fuzzy Rules
Appendix I  Parameters Used in the Experiments of Market-based Rule Evolution based on the NeuroLinear
Appendix J Parameters Used in the Experiments of Market-based Rule Evolution based on No Prior Knowledge
# List of Figures

1.1 A general model of knowledge discovery .............................................. 2  
2.1 A feed-forward neural network .............................................................. 18  
2.2 Nodes in a network .................................................................................. 18  
2.3 The logistic function ............................................................................... 19  
2.4 The hyperbolic tangent function ............................................................ 20  
2.5 Membership functions representing three fuzzy sets for the variable “age” .... 24  
2.6 An example of interpreting one fuzzy rule .............................................. 27  
2.7 An example of interpreting three fuzzy rules .......................................... 28  
2.8 The fuzzy neural network (FuNN) architecture ....................................... 33  
2.9 Initial and adjusted membership functions for a fuzzy variable used in a FuNN fuzzy neural network ......................................................... 35  
2.10 A decision tree ....................................................................................... 44  
2.11 Connection weights of a trained FuNN are interpreted as fuzzy rules by the ReFuNN algorithm ................................................................. 51  
2.12 Crossover (bit-string chromosome) .......................................................... 62  
3.1 Solution set for golf course suitability (produced by human expert) ............ 74  
4.1 The relationship between parameter $p$ and the number of existing weights $n$ .................. 89  
4.2 Difference between NBS approach and PBS approach ($P = 0.5, N = 100$) .... 90  
4.3 A network with 13 weights for Iris data classification ................................ 93  
4.4 Pruned network with 9 connections ....................................................... 95  
5.1 Region of blocks under analysis ............................................................. 114  
5.2 Attributes of altitude difference merged by the data filtering procedure ........ 115  
5.3 Attributes of altitude difference, rainfall and mean temperature merged by the data filtering procedure ......................................................... 116  
5.4 Membership functions for attribute `petal-width` of the Iris data set created by the Chi2-based approach ....................................................... 119  
5.5 Membership functions for the attribute `number of times pregnant` by (a) centre-based and (b) Chi2-based approach ................................. 122  
5.6 Membership functions for all the input variables of the Golf Course Problem by the centre-based approach ................................. 126
5.7 Membership function for suitability ................................................ 127
5.8 Membership functions for all the input variables of the Golf Course Problem produced by the Chi2-based approach .................................................. 128
5.9 (a) Centre-based FuNN solution on the whole data set (b) Chi2-based FuNN solution on the whole data set .................................................. 129
6.1 Model of a market-based rule learning system ............................... 133
6.2 The market-based trading algorithm in action ............................... 137
6.3 The genetic algorithm in an MBRL system .................................. 139
6.4 Illustration of Montana and Davis’s crossover method ................. 140
6.5 Illustration of Davis’s mutation method .................................... 141
6.6 Values of the payments for different payment coefficients and activation taxes in the steady state ($R_{ss} = 1$) ............................................... 148
6.7 Values of the strength for different payment coefficients and activation taxes in the steady state ($R_{ss} = 1$) ............................................... 148
6.8 Strengths in steady state for different values of life tax and number of inactive steps ($R_{ss} = 1$) ................................................................. 150
6.9 Strengths in steady state when life tax equals 0.002 and 0.04 ........... 150
7.1 The general model of market-based rule evolution and refinement based on fuzzy rules .......................................................... 154
7.2 Illustration of converting a fuzzy rule into a list that serves as a rule representation in the market-based rule learning system .................. 156
7.3 Illustration of converting input information into a list that serves as an environmental message that is provided as input to the market-based rule learning system ....... 157
7.4 Market-based rule learning system based on fuzzy rules .................. 159
7.5 Illustration of converting a NeuroLinear rule from rule base 1 into a list that serves as a rule representation in the market-based rule learning system .................. 165
7.6 Illustration of converting a NeuroLinear rule from rule base 2 into a list that serves as a rule representation in the market-based rule learning system .................. 166
7.7 An example of an environmental message for the market-based rule learning system based on the NeuroLinear rules .................. 166
7.8 Market-based rule evolution system based on NeuroLinear rules ........... 168
7.9 Illustration of crossover operation on individuals from rule base 1 .......... 171
## List of Tables

2.1 Intervals of hidden nodes ................................................. 48
2.2 The combinations of the discretized activation values at the three remaining hidden nodes in the pruned network trained for the Iris Data Classification data set. .......... 48
3.1 Attribute descriptions for the Iris data set .............................. 71
3.2 Class information for the Iris data set .................................. 71
3.3 Attribute descriptions for the Pima Indians Diabetes data set .............. 72
3.4 Class information for the Pima Indians Diabetes data set ................. 72
3.5 Attribute description for the Wine Recognition data set .................. 73
3.6 Class information for the Wine Recognition data set ..................... 73
3.7 Attribute descriptions for the Golf Course Problem ........................ 74
3.8 Class information for the Golf Course Problem ........................... 75
3.9 The related attributes of the habitat preference of the Sea Cucumber .... 76
3.10 Site condition class ranges for Cook Islands Sea cucumber Habitat data .................. 76
3.11 The related attributes of the New Zealand Asthma Incidence data set .... 77
3.12 Class information for the New Zealand Asthma Incidence data set ....... 77
3.13 Input Attributes of the Cook Islands Sea Cucumber Habitat data for neural network training .................................................. 80
3.14 Input attributes of the New Zealand Asthma Incidence data for neural network training .................................................. 80
4.1 The experimental results for the Iris data set using the neural network feature selection methods ............................................. 92
4.2 The predictive accuracies and tree size of C4.5 before and after Chi2 for the Iris data set ...................................................... 94
4.3 The summarized neural network results on the modified Iris data set for three pruning methods ................................................. 95
4.4 The experimental results for the Pima Indians Diabetes data set using Chi2 feature selection and C4.5 classification .................................. 96
4.5 The experimental results for the Pima Indians Diabetes data set using the neural network feature selection methods .......................... 97
4.6 The experimental results for the Wine Recognition data set using
4.7 The experimental results for the Wine Recognition data set using the neural network feature selection methods.

4.8 The experimental results for the Golf Course Problem using Chi2 feature selection and C4.5 classification.

4.9 The experimental results for the Golf Course data using the neural network feature selection methods.

4.10 The experimental results for the Cook Islands Sea Cucumber data using Chi2 feature selection and C4.5 classification.

4.11 The experimental results for the Cook Islands Sea Cucumber data using the neural network feature selection methods.

4.12 Experimental parameter setting for the New Zealand Asthma Incidence data.

4.13 The experimental results for the New Zealand Asthma Incidence data using Chi2 feature selection and C4.5 classification.

4.14 The experimental results for the New Zealand Asthma Incidence data using the neural network feature selection methods.

4.15 The P-values associated with predictive accuracy when comparing the three neural network feature selection schemes with C4.5 procedure based on the Chi2-selected features.

4.16 A summary of number of neural network connections before and after neural network selection methods on six data sets.

4.17 A summary of number of features before and after applying the Chi2 algorithm and neural network feature selection approaches on six data sets.

5.1 Test results of the neural network after the data filtering procedure.

5.2 Fuzzy neural network configuration based on the centre-based approach for the Pima Indians Diabetes data set.

5.3 The number of intervals and interval boundaries after Phase 2 of the Chi2 method for the Pima Indians Diabetes data set.

5.4 Fuzzy neural network configuration based on the Chi2-based approach for the Pima Indians Diabetes data set.

5.5 Accuracy rates (%) of the fuzzy neural networks for the Pima Indians Diabetes data set.
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.6</td>
<td>The number of intervals and interval boundaries after Phase 2 of the Chi2 method for the Cook Islands Sea Cucumber data set</td>
</tr>
<tr>
<td>5.7</td>
<td>Accuracy rates (%) of the fuzzy neural networks for the Cook Islands Sea Cucumber data set</td>
</tr>
<tr>
<td>5.8</td>
<td>The number of intervals and interval boundaries after Phase 2 of the Chi2 method for the Golf Course Problem</td>
</tr>
<tr>
<td>5.9</td>
<td>The test result for the fuzzy neural network for the Golf Course Problem</td>
</tr>
<tr>
<td>6.1</td>
<td>Some parameters used in the experiments</td>
</tr>
<tr>
<td>7.1</td>
<td>The fuzzy rule base after encoding (Iris Classification)</td>
</tr>
<tr>
<td>7.2</td>
<td>The updated strengths of fuzzy rules after training with one example (Iris Classification)</td>
</tr>
<tr>
<td>7.3</td>
<td>The updated strengths of fuzzy rules after training with 135 examples (Iris Classification)</td>
</tr>
<tr>
<td>7.4</td>
<td>Values of the parameters used in the experiment of market-based rule evolution starting with the ReFuNN-generated fuzzy rules in connection with the Iris data set</td>
</tr>
<tr>
<td>7.5</td>
<td>A comparison of accuracy performance, and number of rules for the ReFuNN-generated rules before and after using the market-based procedure for one cross-validation trial of the Iris data set</td>
</tr>
<tr>
<td>7.6</td>
<td>NeuroLinear-generated rule base 1</td>
</tr>
<tr>
<td>7.7</td>
<td>NeuroLinear-generated rule base 2</td>
</tr>
<tr>
<td>7.8</td>
<td>Performance of the NeuroLinear-generated rules</td>
</tr>
<tr>
<td>7.9</td>
<td>Updated rule base 2</td>
</tr>
<tr>
<td>7.10</td>
<td>Parameters used in the experiment with market-based rule evolution starting with rule base 1 and rule base 2 generated by the NeuroLinear technique for Iris classification</td>
</tr>
<tr>
<td>7.11</td>
<td>A comparison of accuracy performance, and number of rules for the NeuroLinear-generated rules before and after using the market-based procedure for one cross-validation trial of the Iris data set</td>
</tr>
<tr>
<td>7.12</td>
<td>Updated rule base 1</td>
</tr>
<tr>
<td>7.13</td>
<td>A comparison of accuracy rule performance and number of rules before and after the market-based rule learning system was applied and started without prior knowledge for the Iris data set</td>
</tr>
</tbody>
</table>
7.14 The evolved rule base 1 generated by the MBRL system starting with no prior knowledge for the Iris data set ................................................................. 177
7.15 The evolved rule base 2 generated by the MBRL system starting with no prior knowledge for the Iris data set ................................................................. 177
8.1 Accuracy(%) of the rule sets extracted from the four existing rule extraction techniques in connection with the Iris data set ................................................. 182
8.2 The P-values associated with test accuracy (Iris data set) ............................... 182
8.3 Complexity (number of rules and number of antecedents per rule) of the rule sets extracted from the four existing rule extraction techniques in connection with the Iris data set ................................................................. 183
8.4 Accuracy(%) of the rule sets extracted from the four existing rule extraction techniques in connection with the Pima Indians Diabetes data set ......................... 183
8.5 The P-values associated with test accuracy (Pima Indians Diabetes data set) ....... 184
8.6 Complexity (number of rules and number of antecedents per rule) of the rule sets extracted from the four existing rule extraction techniques in connection with the Pima Indians Diabetes data set ................................................................. 184
8.7 Accuracy(%) of the rule sets extracted from the four existing rule extraction techniques applied to the Wine Recognition data set ................................................. 185
8.8 The P-values associated with test accuracy (Wine Recognition data set) ............. 185
8.9 Complexity (number of rules and number of antecedents per rule) of the rule sets extracted from the four existing rule extraction techniques applied to the Wine Recognition data set ................................................................. 185
8.10 Accuracy(%) of the rule sets extracted from the four existing rule extraction techniques applied to the Golf Course Problem ................................................................. 186
8.11 The P-values associated with test accuracy (Golf Course Problem) ................... 186
8.12 Complexity (number of rules and number of antecedents per rule) of the rule sets extracted from the four existing rule extraction techniques applied to the Golf Course Problem ................................................................. 186
8.13 Accuracy(%) of the rule sets extracted from the four existing rule extraction techniques in connection with the Cook Islands Sea Cucumber Habitat data set ............. 187
8.14 The P-values associated with test accuracy (Cook Islands Sea Cucumber Habitat data set) ................................................................. 187
8.15 Complexity (number of rules and number of antecedents per rule) of the rule sets extracted from the four existing rule extraction techniques in connection with the Cook Islands Sea Cucumber data set ............................................................... 188
8.16 Accuracy(%) of the rule sets extracted from the four existing rule extraction techniques in connection with the New Zealand Asthma Incidence data set ......................... 188
8.17 The P-values associated with test accuracy (New Zealand Asthma Incidence data set) ................................................................................................................................. 189
8.18 Complexity (number of rules and number of antecedents per rule) of the rule sets extracted from the four existing rule extraction techniques in connection with the New Zealand Asthma Incidence data set ......................................................... 189
8.19 Comparison of accuracy, number of rules, number of antecedents per rule, and their associated P-values for the ReFuNN-generated rules before and after using the market-based procedure in connection with the Iris data set ......................................................... 190
8.20 A comparison of accuracy, number of rules, number of antecedents per rule, and their associated P-values for the NeuroLinear-generated rules before and after using the market-based procedure applied to the Iris data set ................................................................. 191
8.21 In connection with the Pima Indians Diabetes data set, a comparison of accuracy performance, number of rules, number of antecedents per rule, and their associated P-values for the ReFuNN-generated rules before and after using the market-based procedure ................................................................. 192
8.22 In connection with the Pima Indians Diabetes data set, a comparison of accuracy, number of rules, number of antecedents per rule, and their associated P-values for the NeuroLinear-generated rules before and after using the market-based procedure ................................................................................................................................. 193
8.23 In connection with the Pima Indians Diabetes data set, a comparison of accuracy, number of rules, and their associated P-values for the randomly generated rules before and after using the market-based procedure ................................................................. 193
8.24 In connection with the Wine Recognition data set, a comparison of accuracy, number of rules, number of antecedents per rule, and their associated P-values for the ReFuNN-generated rules before and after using the market-based procedure ......................... 194
8.25 In connection with the Wine Recognition data set, a comparison of accuracy, number of rules, number of antecedents per rule, and their associated P-values for the NeuroLinear-
generated rules before and after using the market-based procedure .................. 195
8.26 In connection with the Wine Recognition data set, a comparison of accuracy, number of rules, and their associated P-values for the randomly generated rules before and after using the market-based procedure .................................................. 196
8.27 In connection with the Golf Course Problem, a comparison of accuracy, number of rules, number of antecedents per rule, and their associated P-values for the ReFuNN-generated rules before and after using the market-based procedure ....................... 196
8.28 In connection with the Golf Course Problem, a comparison of accuracy, number of rules, number of antecedents per rule, and their associated P-values for the NeuroLinear-generated rules before and after using the market-based procedure ....................... 197
8.29 In connection with the Golf Course Problem, a comparison of accuracy, number of rules, and their associated P-values for the randomly generated rules before and after using the market-based procedure .................................................. 198
8.30 In connection with the Cook Islands Sea Cucumber Habitat data set, a comparison of accuracy, number of rules, number of antecedents per rule, and their associated P-values for the ReFuNN-generated rules before and after using the market-based procedure .................................................. 198
8.31 In connection with the Cook Islands Sea Cucumber data set, a comparison of accuracy, number of rules, number of antecedents per rule, and their associated P-values for the NeuroLinear-generated rules before and after using the market-based procedure .................................................. 199
8.32 In connection with the Cook Islands Sea Cucumber data set, a comparison of accuracy performance, number of rules, and their associated P-values for the randomly generated rules before and after using the market-based procedure .................................................. 200
8.33 In connection with the New Zealand Asthma Incidence data set, a comparison of accuracy, number of rules, number of antecedents per rule, and their associated P-values for the ReFuNN-generated rules before and after using the market-based procedure .................................................. 203
8.34 In connection with the New Zealand Asthma Incidence data set, a comparison of accuracy, number of rules, number of antecedents per rule, and their associated P-values for the NeuroLinear-generated rules before and after using the market-based procedure .................................................. 204
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.35</td>
<td>In connection with the New Zealand Asthma Incidence data set, a comparison of accuracy, number of rules, and their associated P-values for the randomly generated rules before and after using the market-based procedure</td>
</tr>
<tr>
<td>8.36</td>
<td>Accuracy performance (%) of the rule sets extracted by different rule extraction techniques</td>
</tr>
<tr>
<td>8.37</td>
<td>Accuracy(%) of the rule sets before and after the market-based rule learning system was applied and started without prior knowledge</td>
</tr>
<tr>
<td>8.38</td>
<td>Number of rules extracted by different rule extraction techniques</td>
</tr>
<tr>
<td>8.39</td>
<td>Average number of antecedents per rule for rule sets (excluding default rules) extracted by different rule extraction techniques</td>
</tr>
<tr>
<td>8.40</td>
<td>Accuracy performance (%) on the test set of the learning models and of the rules extracted by different techniques</td>
</tr>
</tbody>
</table>
Part I
Introduction and Background

Part I describes the context of this research by presenting the research introduction and background, and comprises Chapters 1–3.
Chapter 1 Introduction

As computer power grows and data collection technologies advance, a plethora of data is generated in almost every field where computers are used. Human beings rely more and more on computers to accumulate, process, and make use of data. Without the aid of improved computing technologies, there is no doubt that huge amounts of data collected would never be examined. The ease of collection and the increasing availability of data stores has led to demands for new methods for analyzing these data and deriving significant knowledge that may be latent in these data stores. In particular there is hope that the use of new analytical techniques in connection with data-mining operations may reveal hidden relationships that lie buried within these data sets. In order to make raw data useful, it is necessary to represent, process, and extract knowledge for various applications. This research investigates adaptive knowledge discovery techniques for the task of data mining applications so that relevant features can be identified, and inference rules can be extracted and refined from the problem domains. In this chapter, Sections 1.1 and 1.2 provide the research motivation and the goal of the research. Section 1.3 describes sets of criteria to evaluate the feature selection, rule extraction, and rule refinement techniques used in the thesis, and Section 1.4 presents the outline of the thesis.

1.1 Motivation

The increasing interest in data mining, or the use of historical data to discover regularities and improve future decisions, follows from the confluence of several recent trends: the falling cost of data-storage devices and the increasing ease of collecting data over networks; the development of robust and efficient learning algorithms to process these data; and the falling cost of computational power, enabling the use of computationally intensive methods for data analysis. The field of data mining and knowledge discovery has already produced practical applications such as analyzing medical outcomes, detecting credit card fraud, predicting customer purchase behaviour, predicting the personal interests of Web users, and optimizing manufacturing processes (Mitchell 1999). It has also led to a set of scientific questions about how computers might automatically and effectively learn from past experience (Langley 1998).
Figure 1.1. A general model of knowledge discovery

The general model of knowledge discovery used in this research is shown in Figure 1.1. It consists of three basic steps:

1) Pre-processing – data are selected and prepared for the next step.
2) Discovery process – a discovery algorithm is chosen and rules are learned; here rules are typically represented in an IF-THEN format: IF the data that is given to the rule meets some specified condition, THEN an action of a particular type is taken.
3) Post-processing – rules are sifted or grouped for their better use and understanding, since often the number of rules is large and some rules are more “interesting” than others (Liu & Motoda 1998).

Many research problems still remain in these three steps. They are listed as follows:

Problems in Pre-processing
Feature selection is one of the long-existing methods for data pre-processing. Its objective is to select a minimal subset of features according to some reasonable criteria (Liu & Motoda 1998), such as whether selected features are adequate for predictive accuracy of a learning task or whether selected features help simplify the learned results so that they are more understandable. By choosing the minimal subset of features, irrelevant and redundant features can be removed according to certain criteria. Although feature selection has been the focus of interest for quite some time, solving real-world problems, such as analyzing spatial data sets is still a challenge (Purvis et al. 2001). For the large size of spatial information data sets, it may be advantageous to perform data filtering by means of feature selection to reduce the size of the data set without
sacrificing the discriminating power of the original data prior to carrying out further computational analysis.

For the problems existing in other techniques, such as the difficulty of selecting appropriate membership functions in a fuzzy system (Klir & Yuan 1995), feature selection techniques such as Chi2 algorithm (Liu & Setiono 1995a) may offer solutions.

Moreover, the most widely-used type of non-symbolic computational module, artificial neural networks, has been extensively used for data pattern recognition (Schoenenburg 1990; Migrin 1993; Shustorovich & Thrasher 1996). However, artificial neural networks can be further enhanced to reduce the number of irrelevant and redundant features by adapting network architectures so that a minimal neural network structure with fewer nodes and connection weights can be obtained. The main technique used here is neural network pruning.

**Problems in the Discovery Process**

Many approaches such as decision trees (Quinlan 1986) and the rule generator X2R (Liu & Tan 1995) exist for rule extraction from raw data, but they are noise-intolerant and, more importantly, cannot be applied to incomplete or corrupted data. Therefore, improved data analytical techniques are in demand.

In recent years there has been interest in this respect in the application of artificial intelligence techniques, such as expert systems, fuzzy systems and artificial neural networks (Purvis et al. 1996). Each of these approaches has specific advantages when used for data analysis. For example, rule-based expert systems facilitate the input and refinement of expert knowledge in problem-solving modules. They also enable straightforward tracing of the reasoning process by which a particular outcome is achieved. Fuzzy systems extend the notion of ordinary rule-based expert systems to allow for the treatment of vague and uncertain knowledge in a systematic fashion. These systems can be efficiently implemented and can incorporate human-like reasoning about qualitative information. Neural networks, in contrast with the above, need not be initially supplied with expert knowledge and can be applied to virtually any data set without presuppositions concerning the distribution of the training data (Zhuang & Engel 1990). They can be applied to incomplete or corrupted data and still yield acceptable results. Moreover, it has
been shown that a three-layer feed-forward neural network with a sufficient number of hidden nodes can approximate any continuous function to any desired accuracy (Cybenko 1989). In many cases, neural networks are the preferred learning method simply because they have a better generalization ability than those of competing algorithms. Several empirical studies have pointed out that there are some problem domains in which neural networks provide predictive accuracy that is superior to commonly used symbolic learning algorithms (Atlas et al. 1989; Fisher & Mckusick 1989; Weiss & Kapouleas 1989; Shavlik et al. 1991) and statistical techniques (Zhuang & Engel 1990). On the other hand, the reasoning associated with neural networks is essentially a “black box” whose operation is not evident to the user. For example, it is usually not possible to explain why a certain pattern has been classified in a particular fashion by the network. It would be advantageous if explicit rules comprehensible to humans could be extracted to describe the classification behaviour of the network, and efforts have been made to find effective algorithms to extract such rules from trained feed-forward neural networks (Towell & Shavlik 1993; Fu 1994; Setiono & Liu 1997b). In particular, Setiono and Liu have presented a promising approach, “NeuroLinear”, which extracts linear rules for problem domains with continuous and discrete input attributes. Apart from rule extraction, there are a number of other techniques (Wikel et al. 1996; Poel 1998; Primeaux 2001) for extracting information from trained neural networks in order to overcome the black box behaviour of neural networks. However, these techniques are outside the scope of this research, and hence will not be discussed in this thesis. Fuzzy neural networks (Lee & Lee 1974), which combine neural networks and fuzzy logic, have also attracted attention. Several techniques (Horikawa et al. 1992; Kasabov 1993a) have been presented to extract fuzzy rules from fuzzy neural networks.

With the various known rule extraction techniques in hand, it is important to evaluate the relative strengths and weaknesses of these techniques in the context of different data sets, so that an understanding of how these techniques can best be used is gained.

In the past years, there has been interest in studying complex interactive systems by simulating the economic behaviour of markets (Clearwater 1996). The ability of markets to facilitate resource allocation with very little information makes them an attractive solution for many complex problems. A market-based system offers many features that rule extraction and refinement systems could utilize (Zhou & Purvis 1999). For example, a fascinating aspect of a
market is that through the simple interactions of trading (buying and selling) among individual agents, a global optimization can be achieved, such as achieving stable prices or a fair allocation of resources. The goal of a rule discovery system is to discover a set of rules that, when applied to the input data, leads to satisfactory behaviour of the system. By adopting the concept of economic trading behaviour among individual commercial agents, the rule discovery system can be thought as an artificial market where individual rule agents are interacting and competing in order to achieve some overall global behaviour.

Holland was the first author to use economic terminology and insight in a multi-agent learning system. His classifier system work (Holland 1986) attracted considerable interest for a number of years. Holland’s classic classifier system is an on-line learning system that seeks to gain reinforcement from its environment based on an evolving set of simple string-encoded rules called classifiers. It distributes credit to a large number of sequentially acting classifiers by means of strictly local transactions among them. Via the credit assignment process, classifiers useful in gaining reinforcement are selected and propagated over others less useful, leading to improved system performance. While its initial appearance was promising, the evolution of the classifier system idea has not been rapid. Only a limited number of successful applications (Booker 1982; Goldberg 1983; Wilson 1985; Dorigo & Sirtori 1991; Roberts 1993) were subsequently reported. A classifier system has major drawbacks: it is difficult to interpret (Wilson 2000), hard to generate initial rule chains (Wilson & Goldberg 1989), and it is difficult to set initial system parameters (Richards 1995). As the result of these weaknesses, according to Wilson (1999), a classifier system is rarely the technique of choice in practical applications.

The market-based rule learning (MBRL) system proposed in this thesis is an adaptive rule-learning system inspired by a classifier system. Although it takes the main structure of a classifier system, it introduces the fundamental change by importing existing rule sets generated by other rule extraction techniques into the system. Moreover, it presents modifications in each of the layers of the structure. With the change and modifications introduced by the MBRL system, the problems existing in previous classifier systems can be solved or lessened. This will be discussed in detail in Chapter 6.
Problems in Post-processing

Simplicity and accuracy are two desirable features of the derived rule set (Craven 1996). It is important to have an additional post-processing tool in hand to refine extracted rules so that they are easier to comprehend and more effective. Many traditional rule-refinement systems follow the strategy of selecting a single rule to refine at each step of the process. This rule-by-rule procedure can be a bottleneck in the refinement process. Market-based techniques therefore can provide benefits to the post-processing stage (as well as to the discovery process as outlined above). They can avoid this bottleneck and may therefore offer advantages for efficiently discovering a simpler set of refined rules that still perform effectively.

In this thesis we apply the proposed market-based rule learning (MBRL) system to post-processing the output of (fuzzy) neural network rule extraction algorithms to generate human-understandable rules with high accuracy and/or small rule size.

1.2 Research Goals

The goal of this research is to investigate and develop new adaptive techniques for the task of discovering relevant features and inference rules from data. Using the three steps of the knowledge discovery process as a basis, the research focuses on tackling the research problems discussed in the previous section as follows:

Pre-processing (feature selection)

- With the existing difficulty of selecting appropriate membership functions in fuzzy systems, a novel approach of employing a $\chi^2$ statistic-based feature selection method, called the Chi2 algorithm (Liu & Setiono 1995a), to make automatic selection of the number and widths of the membership functions is proposed.
- In response to the challenge of applying feature selection for solving real-world problems involving large data sets, a Chi2-based spatial data filtering is proposed to reduce the number of spatial data items and the number of features. The resulting data set can then be more efficiently used in connection with neural network computation.
• With the aim of implementing feature selection via neural networks, three different pruning mechanisms are proposed and compared. These three neural network pruning mechanisms are evaluated by experimental comparisons to the Chi2 algorithm.

**Discovery Process**

• For the purpose of integrating a ‘market trading’ technique to the field of rule discovery and refinement with respect to data mining, market-based rule learning (MBRL) system inspired by classifier system is proposed. The proposed system is investigated by applying it to learning tasks when no prior knowledge is given.

• Four existing rule extraction approaches, including an established method based on C4.5 decision trees (Quinlan 1986), the rule generator X2R (Liu & Tan 1995), a fuzzy neural network rule extraction technique ReFuNN (Kasabov 1993a), and a feed-forward neural network rule extraction technique NeuroLinear (Setiono & Liu 1997b), are evaluated empirically in the context of problem domains that involve supervised classification learning.

**Post-processing**

• The market-based rule learning (MBRL) system is proposed in combination with the fuzzy neural network rule extraction algorithm, ReFuNN, to provide a general framework for fuzzy inference-based rule discovery.

• The MBRL system is proposed in combination with the feed-forward neural network rule extraction algorithm, NeuroLinear, to provide a framework for improving the quality of rules extracted from feed-forward neural networks.

• The use of the MBRL system as a post-processing tool is evaluated by conducting experiments on problem domains that involve supervised classification learning.

This thesis does not make the claim that the general framework comprising a market-based evolution system with (fuzzy) neural network rule extraction algorithms are ideal for all learning tasks. The aim is to provide an alternative approach to other approaches, that might be useful in solving some tasks when there is difficulty dealing with them using more conventional techniques.
In the thesis, a set of six selected data sets was used to conduct the experiments and evaluate the relevant techniques.

1.3 Evaluation Criteria
As described earlier, the goal of this research aims to investigate adaptive discovery techniques for data mining so that relevant features can be identified, and inference rules can be extracted and refined from the problem domains. Following the conventional three steps of the knowledge discovery process described earlier, feature selection techniques, rule extraction techniques and rule refinement techniques are investigated. In order to evaluate these techniques, we first identify a set of appropriate evaluation criteria.

1.3.1 Evaluation Criteria for Feature Selection Techniques
Three assessment measures are commonly used (Liu & Motoda 1998) to evaluate feature selection techniques. They are listed below.

- **Predictive accuracy.** Can the feature selection methods help to improve the predictive accuracy of a learning task? Predictive accuracy is a commonly used measure to assess whether the selected features can improve or maintain the system’s performance on unseen data. For the problem domains that involve supervised classification learning, the predictive accuracy of the system is defined based on the classification performance, i.e. the percentage of the correct classified examples out of the total number of testing examples, as follows:

  \[
  \text{Predictive Accuracy} = \frac{\text{Number of correct examples}}{\text{Total number of testing examples}}
  \]

- **Comprehensibility.** Can the feature selection methods help to reduce the complexity of representations? It is generally agreed that it is difficult for humans to find regularities and understand underlying relationships in data sets. There are various representations for expressing induced knowledge from learning algorithms. For example, there are learning algorithms that represent their learned knowledge as decision trees (Quinlan 1993), and neural networks (Rumelhart et al. 1986), Although it can be a subjective matter as to which representation is more understandable than another, it is reasonable to seek the simplest possible for a given
representation, where ‘simpler’ might mean a smaller number of structural elements or associations among the elements. In general one might expect that the simpler a representation, the easier it will be to comprehend. For example, other things being equal, we would prefer a smaller size of neural network than a larger one.

- **Data simplicity.** Can the feature selection methods remove noisy, irrelevant, and redundant features? The number of features selected is a measure for assessing the size of data: *the smaller, the better*, because a smaller number of features, implying less complexity of a data set, means fewer potential hypotheses, faster learning, and simpler results.

A feature selection method can be evaluated along these three dimensions. If a method can achieve the best scores in all three, it is judged to be the best method overall. Otherwise, one can compare along each dimension and check which method gets better scores in more dimensions. With these three dimensions in mind, two other important practical factors should also be considered:

- **Speed of feature selection methods.** How quickly can feature selection methods obtain results? This is particularly important when we deal with large data sets.

- **Generality of selected features.** The generality of selected features is not always the most important. However, in order to reduce the data collection effort, we want to know whether the same set of relevant features are selected by different techniques. For example, we can inspect whether the selected features from the neural network feature selection approaches overlap those selected by the Chi2 algorithm on the same application.

The feature selection techniques used in the thesis will be evaluated with respect to the above five criteria in Chapter 4.

**1.3.2 Evaluation Criteria for Rule Extraction and Refinement Techniques**

Four commonly used measurements to evaluate rule extraction and refinement techniques are as follows:
- **Predictive accuracy.** Typically the most important consideration in rule inductive learning is to induce a rule set that has a high level of predictive accuracy (Craven 1996). So whether the extracted or refined rule sets from the relevant techniques can make accurate predictions on previously unseen cases should be measured.

- **Comprehensibility.** Often the comprehensibility of learned rule sets is an important consideration. That is, does the technique extract (or refine) rule sets in such a way that they may be inspected and understood by experts in the domain? Measuring the comprehensibility of learned rule sets is a problematic issue. An underlying premise of the experiments conducted in this thesis is that *syntactic complexity* is a good indicator of comprehensibility. For a given representation language, other things being equal, simpler descriptions are better than complex descriptions. The psychological literature supports the notion that humans prefer simple concepts (Neisser & Weene 1962; Pinker 1979; Medin *et al.* 1987). The specific measures of syntactic complexity that are used to assess the comprehensibility of rule sets in this thesis are: (1) the number of rules in the rule set, and (2) the average number of antecedents per rule in the rule set.

- **Fidelity.** Fidelity is a quality attribute associated with rules that have been generated from a learning model. Rule fidelity is a measure of the agreement between the classification of the test set (unseen cases) by the learning model and by the rule set extracted from the learning model (Andrews *et al.* 1995). A rule set is considered to display a high level of fidelity if it can mimic the behaviour of the learning model, such as an artificial neural network, from which it was extracted by capturing a large proportion of the information embodied in the model. For example, consider a neural network (feed-forward neural network or fuzzy neural network) as a learning model, fidelity is measured by the difference in accuracy on test samples between the neural network extracted rules (NeuroLinear-generated rules or ReFuNN-generated rules) and neural network outputs.

- **Consistency.** Consistency measurement is another quality attribute associated with rules that have been generated from a learning model. Under Towell and Shavlik’s criteria (1993), an extracted rule set is deemed to be consistent if, under differing training sessions, a learning model generates rule sets which produce the same classification of unseen examples.
In addition to predictive accuracy, comprehensibility, fidelity and consistency, there is another criterion that is often an important consideration when evaluating rule extraction techniques:

- **Scalability.** Scalability is the ability of a rule extraction technique to be applicable to data sets with increasingly larger numbers of features and larger numbers of samples. The scalability of a neural network rule extraction algorithm defined by Craven and Shavlik (1999) is as follows: Scalability refers to how the running time of a rule extraction algorithm and the comprehensibility of its extracted rule sets vary as a function of such factors as network, feature-set and training-set size.

Scalability is a quality measurement for discovery process tools (rule extraction techniques). The scalability of post-processing tools (rule refinement techniques) does not play a crucial role for knowledge discovery and data mining applications. It is expected that a rule refinement technique only deal with existing rule sets extracted from other learning techniques and not with the original data set, and thus the rule refinement technique does not raise particular concerns about scalability.

The proposed MBRL system and four existing rule extraction techniques will be empirically evaluated against the criteria of predictive accuracy, comprehensibility, fidelity and consistency. The scalability criterion will be further discussed in Chapter 8.

### 1.4 Outline of Thesis

This thesis consists of four parts, each of which is made up of one to four chapters, a total of ten chapters. Each part revolves around a theme. Part I presents the research introduction and background. Part II is devoted to feature selection (pre-processing) issues. The issues concerning both rule extraction and rule refinement (discovery process and post-processing) are together described in Part III. The motivation for grouping rule extraction issues with rule refinement issues in one part is simply because the proposed MBRL system can be applied both in the rule extraction (discovery) step and the rule refinement (post-processing) step, and the existing rule extraction techniques are used to conduct the experiments for the purposes of experimental comparisons to the MBRL system. Part IV concludes the thesis.
The remainder of this thesis is organized as follows:

Part I provides context for understanding the contributions of the work presented in subsequent chapters. Chapter 2 provides background material for the rest of the thesis. It describes the statistical methods used in the experiments of this thesis, gives a brief introduction to neural networks, fuzzy systems and fuzzy neural networks, provides a review of related work in feature selection and rule extraction, and describes the field of classifier systems including genetic algorithms in detail. One existing statistic-based feature selection method, called the Chi2 algorithm, and four existing rule extraction methods: C4.5 decision trees, the rule generator X2R, the feed-forward neural network rule extraction approach NeuroLinear, and the fuzzy neural network rule extraction approach ReFuNN, are discussed and illustrated. Chapter 3 provides a description of the problem domains used in the experiments of the thesis.

Part II is devoted to feature selection (pre-processing) issues, concerning both statistical and connectionist methods. Chapters 4 and 5 belong to this part. In Chapter 4, how to achieve feature selection via neural networks is discussed. Three neural network pruning schemes are then proposed. A detailed analysis and comparison of experimental results using six data sets is presented. Chapter 5 proposes a Chi2-based spatial data filtering technique and a Chi2-based membership function selection method for fuzzy systems. Both of these methods demonstrate how techniques from separate approaches can be combined to yield better results.

Part III is about issues concerning rule extraction and refinement (in the discovery process and post-processing steps), and comprises Chapters 6 – 8. In Chapter 6, the architecture and computational processes of the proposed market-based rule learning (MBRL) system are described in detail, with emphasis on differences between the proposed system and the classic classifier system. The steady-state behaviour of the proposed system is analyzed mathematically and its time-complexity is evaluated. Chapter 7 focuses on how the proposed market-based rule learning (MBRL) system can be used as a rule-refinement tool to improve the quality of extracted rules from fuzzy neural networks and feed-forward neural networks. A description is given of how to encode and feed the input information and initial rule set to the system, how learning is carried out, and how the search is performed by the genetic algorithm. Chapter 8 first presents experiments with rule evolution and refinement using the MBRL system with six selected data
sets. For comparison and evaluation purposes, experiments on postprocessing the output of four existing rule-extraction techniques, including C4.5 decision trees, the rule generator X2R, the NeuroLinear approach, and the ReFuNN approach, are also described. In this chapter, an empirical evaluation of the proposed MBRL system and the selected existing rule-extraction techniques is also presented. The strengths and weaknesses of different rule generation and refinement methods are examined. The results provide evidence that can be used for better informed selection and usage of available rule extraction and refinement methods.

Part IV contains only Chapter 9, which summarizes the contributions of this thesis, limitations of the work presented, and proposes for future work.

The attached CD contains 10 appendices, which present the representative rules extracted or refined by different rule extraction and refinement techniques for each problem domain, and the parameters used in the experiments.
Chapter 2 Background

2.1 Introduction
This chapter provides background material for the remainder of the thesis. Section 2.2 describes the statistical methods used in the experiments. Section 2.3 provides a description of feed-forward neural networks, fuzzy systems, and fuzzy neural networks employed in the algorithms of the thesis. Section 2.4 gives an overview of feature selection methods, and, in particularly, describes a $\chi^2$ statistic-based feature selection algorithm, called the Chi2 algorithm (Liu & Setiono 1995a). This material is presented as the empirical evaluation of three pruning schemes associated with feature selection via neural networks presented in Chapter 4 involves experimental comparisons to the Chi2 algorithm. Moreover, the spatial data filtering and the membership function selection method proposed in Chapter 5 are based on the Chi2 algorithm. Section 2.5 surveys other work that has been done in the area of rule extraction, in particular, four existing rule extraction methods that are described and illustrated in detail. Section 2.6 gives detailed descriptions and discussions of classifier systems including genetic algorithms. This discussion provides the background appropriate for understanding the novel aspects of the market-based rule learning (MBRL) system presented in Chapter 6.

2.2 Statistical Methodology
In this thesis, statistical methods are used in the experimental evaluation of proposed techniques and various existing techniques in Chapters 4, 5, 7 and 8. This section briefly discusses two types of statistical methods that are important in later chapters: estimation and hypothesis testing.

2.2.1 Estimation
The task in an estimation problem is to determine the value of some parameters of interest. The experiments in this thesis, in particular, are concerned with estimating the predictive accuracy of various feature selection, rule extraction and rule refinement techniques, and with estimating the complexity of learning models and inference rules extracted from the models.

The basic method for estimating the predictive accuracy of a learning algorithm is to measure its accuracy on a set of examples that were not used during the learning process. Such a set is called
a test set. Unless the size of the available data set is quite large, a preferred method for accuracy estimation is to use cross validation (Stone 1974). In k-fold cross validation, the available data is partitioned into k separate sets of approximately equal size. The cross-validation procedure involves k iterations in which the learning method is given k-1 of the subsets to use as training data, and is tested on the set left out. Each iteration leaves out a different subset so that each is used as the test set exactly once. The cross-validation accuracy of the given method is simply the average of the accuracy measurements from the individual folds. In the experiments in the thesis, predictive accuracy, comprehensibility, and fidelity will be measured using 10-fold cross-validation.

2.2.2 Hypothesis Testing

Hypothesis testing involves evaluating an assertion about the distribution of a random variable. Such an assertion is termed a statistical hypothesis. Hypothesis testing is used in this thesis when evaluating the performance of various experimental techniques.

One type of hypothesis test that is commonly used when evaluating inductive learning techniques is whether two or more techniques have significantly different performances. For example, in Chapters 7 and 8, we will test the hypothesis that the inference rules produced by one technique in some domain are more accurate than those produced by another technique. A statistical test called paired-sample t-test (Sachs 1984) is used to compare the accuracy of one technique to another when the cross validation procedure is employed. In the paired-sample t-test, we first calculate the average of the differences in accuracy measurements \( \text{diff} \) for techniques A and B for k folds:

\[
\text{diff} = \frac{1}{k} \sum_{i=1}^{k} (\text{accuracy}_A^i - \text{accuracy}_B^i) \tag{2.1}
\]

We also calculate the standard deviation, \( s \), of this value. Here, \( \text{accuracy}_A^i \) is the measured accuracy for technique A on the \( i \)-th fold, and \( \text{accuracy}_B^i \) is the accuracy of technique B on the same fold. In order for this test to be valid, the two techniques must have used the same partition for cross validation. The test statistic is then:
The null hypothesis (that the two algorithms have the same level of accuracy) is rejected with 100(1 - \(a\))% confidence if:

\[
|t| \geq t_{a/2, k-1}
\]

where \(a\) is a predefined significance level, \(t_{a/2, k-1}\) defines the rejection region for the test, and it is calculated by using a \(t\) distribution with \(k-1\) degrees of freedom. For example, if using \(a = 0.05\) and \(k = 10\), then \(t_{0.025, 9} = 2.262\). Note that this is a two-tailed test, meaning that the null hypothesis can be rejected either if technique A is more accurate than technique B, or vice versa. It is proper to use a two-tailed test in this situation since we have no \textit{a priori} reason to believe that one technique is less accurate than the other.

A statistical term, \textit{p-value} (Dowdy & Wearden 1991), is often used in hypothesis tests, where one either rejects or fails to reject a null hypothesis. The \(p\)-value represents the probability of rejecting the null hypothesis when it is true. The smaller the \(p\)-value, the smaller is the probability that you would be making a mistake by rejecting the null hypothesis. Therefore, instead of deciding the rejection level beforehand and then conducting a \(t\)-test as outlined above, we can just compute and report the value of the test statistic and its associated \(p\)-value. Then, the reported result will be left to the reader to judge how significant it is. By convention, the most commonly used level of significance is 0.05 (Dowdy & Wearden 1991), that is, one can reject the null hypothesis when the \(p\)-value is less than 0.05. With respect to calculating the \(p\)-value, the reader should refer to (Mendenhall & Sincich 1995).

### 2.3 Neural Networks and Fuzzy Systems

The model era of neural networks began with the pioneering work of McCulloch and Pitts (1943). Since then, many other significant contributions (Hebb 1949; Rosenblatt 1958; Minsky & Papert 1969; von der Malsburg 1973; Grossberg 1976; Kohonen 1982; Hopfield 1982; Rumelhart \textit{et al.} 1986) have been made in the field. One of the most important development in the field of neural networks was the development of an algorithm called \textit{backpropagation algorithm} (Rumelhart \textit{et al.} 1986).
al. 1986) to train multilayered networks. Details of the backpropagation algorithm will be presented later in the section.

Neural network models are inspired by natural physiology and are an attempt to mimic the neurons and synaptic connections of the brain (Hertz et al. 1991). Biological neurons transmit electrochemical signals through neural pathways. Each neuron receives signals from other neurons through special junctions called synapses. Some inputs tend to excite the neurons, others inhibit them. When the cumulative effect exceeds a threshold, the neuron fires and sends a signal to other neurons. The notion of an artificial neuron, created by the artificial intelligence (AI) researchers and simulated on a computer, models these simple biological characteristics. Each artificial neuron is connected to a set of inputs; each input value is multiplied by a weight analogous to a synaptic strength, and the combination of these weighted values is what is acted upon by the artificial neuron.

There is a wide variety of neural network architectures and learning methods for both supervised and unsupervised inductive learning tasks (described below). The work in this thesis focuses on feed-forward neural networks and fuzzy neural networks applied to classification tasks, and therefore the discussion below is restricted to these two particular types of neural network architectures and supervised learning methods.

In order to fully understand fuzzy neural networks, a basic introduction to fuzzy sets, fuzzy logics and fuzzy systems are given. The current difficulty of constructing membership functions in fuzzy systems is also addressed.

2.3.1 Feed-forward Neural Networks

Figure 2.1 shows the architecture of a typical feed-forward neural network model. It has three layers: a layer of input nodes, an intermediate layer of nodes, and a layer of output nodes. Connections are from each input node to each intermediate layer node, and from each intermediate layer node to each output node in a feed-forward manner.
These nodes of different layers are analogous to biological neurons and are the processing units of the network. A node has an activation value, which is determined by a nonlinear activation function of the sum of its inputs, which are the weighted values of the outputs from other nodes that feed into it. Each internode connection has a weight (positive or negative), which is multiplied by the source node output to produce an input for the destination node. This is shown in Figure 2.2, where the two nodes on the left feed into the righthand node.

If the output value of each node is $X_1, X_2$, and $X$, and the values of the weights are $W_1$ and $W_2$, then $X$ is given by
$X = f \left( \sum_{i=1}^{2} X_i W_i + \theta \right)$  \hspace{1cm} (2.4)

where $f$ is the activation function, and $\theta$ is the bias for the rightmost node. The bias of a node, which is an adjustable parameter, can be thought of as the node’s predisposition to have a high (or low) activation before it receives any activation signals from other nodes. Usually a nonlinear function $f$ is characterised so that if $X_i$ is high, and $W_i$ is positive, a weighted value of $X_i$ will tend to increase $X$, whereas if $W_i$ is negative it will tend to decrease $X$. In this way positive weights are analogous to excitatory synapses, and negative weights are analogous to inhibitory synapses of biological neurons. One commonly used activation function (sometimes called a transfer function) is the logistic function:

$$f(x) = \frac{1}{1 + e^{-cx}}$$  \hspace{1cm} (2.5)

where $x$ denotes the input to a node, and $c$ is a constant. As shown in Figure 2.3, this function “squashes” the node’s net input to an activation value in the range $[0,1]$. The value of $c$ determines how steep the curve is.

![Figure 2.3. The logistic function](image)

Another commonly used activation function is the hyperbolic tangent function:
The main difference between the hyperbolic tangent function and the logistic function is with respect to their ranges. The hyperbolic tangent function has a maximum value of 1.0 and a minimum value of -1.0 as shown in Figure 2.4.

\[
f(x) = \frac{e^x - e^{-x}}{(e^x + e^{-x})}
\]  

(2.6)

Figure 2.4. The hyperbolic tangent function

Both the logistic function and the hyperbolic tangent function are sigmoidal functions.

As illustrated in Figure 2.1, a feed-forward neural network is composed of several layers of simple processing nodes. The state of a node at any given time is represented by its activation value, which is a real-valued number, typically in the range [0,1] or in the range [-1,1]. The input layer of a neural network contains nodes whose activations represent values for the features of the problem domain in which the network is being applied. Typically, a real-valued feature is represented by a single input node, and a discrete feature with \(n\) possible values is represented by \(n\) input nodes. The nodes in the output layer of a network represent the decisions made by the network. The role of the hidden nodes is to transform the input space into another space in which it is more profitable for the output units to make linear discriminations.

A feed-forward neural network works in the following way. The input values are presented to the input nodes, which feed into and activate the hidden layer(s). The values produced by the
activation functions at the hidden nodes, in turn, feed forward into the output nodes, and then
determine values of the output nodes.

The learning of neural networks described here is called *supervised learning*, where the model’s
outputs are compared directly with a set of known, correct values. In other words, an example
is presented to the input nodes, and the outputs are produced from the current parameters. The
outputs are then compared with a set of known, correct target values of a given example and an
error function is computed. The weights and biases associated with the internode connections are
then systematically adjusted so as to minimise the error function. Then the next “correct” training
example is presented to the network, and the error minimisation calculation is repeated. After
many passes through the whole training example set, the error usually converges to some value
and, provided that this error value is below some predefined threshold, the network can be said
to have learned a set of relationships.

The most widely used neural network learning method is the *backpropagation algorithm*
(Rumelhart *et al.* 1986). Learning in a neural network involves modifying the weights and biases
of the network in order to minimize an error function, which is a measure of how close the
network’s predictions are to the class labels for the examples in the training set. The error
function is normally defined as the sum of the squared errors:

\[
E = \frac{1}{2} \sum_i \sum_j (A_j - T_j)^2 \tag{2.7}
\]

Here \(i\) ranges over the examples in the training set, \(j\) ranges over the output nodes of the network,
\(T_j\) is the target value for the \(j\)-th output node for a given example, and \(A_j\) is the activation of the
\(j\)th output unit in response to the example.

For the feed-forward neural networks used in the thesis, the activation function implemented by
the network for all the nodes is continuous and differentiable. Therefore, the error function can
be minimized by calculating its partial derivatives with respect to each of the network’s
parameters, and making changes to the parameters as follows:

\[
\Delta \mathbf{w} \propto -\eta \nabla_{\mathbf{w}} E \tag{2.8}
\]
where $\vec{w}$ represents the vector of weights and bias in the network, $\eta$ is the learning rate that basically represents the search step size, and $\nabla_{\vec{w}} E$ is the gradient of the error function $E$ with respect to the vector of weights and bias in the network. Each parameter is updated according to the following rule:

$$w_{new} = w_{old} + \Delta \vec{w}$$  \hspace{1cm} (2.9)

The updated parameters define a new estimate of the outputs for which $E$ should be reduced. The gradient is recalculated at this new point, and the parameters are updated again. The process is repeated until it is considered to have converged to a satisfactory point, which is either when $E$ cannot be made any smaller, or when the outputs are stabilized and sufficiently close to the targets.

Various optimization algorithms can also be used to minimize the error function. Quasi-Newton algorithms (Haykin 1999), such as the BFGS method, treat the supervised training of a neural network as a problem in numerical optimization. At each iteration of the algorithm, a positive definite matrix that is an approximation of the inverse of the Hessian of the function to be minimized is computed. The positive definiteness of this matrix ensures that a descent direction can be generated. Given a descent direction, a step size is computed via an inexact line search algorithm. It has been shown that Quasi-Newton algorithms, such as the BFGS method can speed up the training process significantly (Watrous 1987). They do however require more storage for the matrix, which may be a disadvantage in a parallel implementation. Details of the BFGS algorithm can be found in Dennis & Schnabel (1983). Standard gradient descent augmented with a momentum term is sometimes used to minimize the error function, as is the Conjugate-Gradient method. Rinnooy Kan and Timmer (1989) have found that for a moderate number of weights, various Quasi-Newton algorithms are more efficient; for a large number of weights, various Conjugate-Gradient algorithms are more efficient.

Often, network training is stopped before a local minimum in the error function is reached. The motivation underlying these techniques of early stopping is that over-fitting may occur if the network is trained to the training data too closely. Over-fitting is a phenomenon which indicates that the neural network has too closely approximated (learned) the training data, which may
contain noise. In such a case the network can not generalize well on new examples. One method for estimating a good stopping point is to use a validation set of data that is distinct from the training set data to monitor the predictive accuracy of the network as it is being trained. Instead of stopping training based on the error-function minimum for the training data, this procedure saves the weights from the iteration of the optimization method that results in the lowest value of the error function for the validation set data. All the neural-network related experiments described in this thesis use this method of employing separate training and validation data sets.

The ability of feed-forward neural networks to represent complicated and highly nonlinear relationships, as well as to generalize the recognition behaviour to new situations has attracted considerable interest. Although many real-world applications of this technique attest to its viability, it also has some disadvantages, such as the fact that neural network knowledge is coded as a large number of numerical weights, and their semantics (in terms of the problem to be solved) are not explicit. Two issues are important for getting past these disadvantages: how to represent and how to extract neural network knowledge contained in the trained set of weights.

2.3.2 Fuzzy Systems

The notion of fuzzy sets, fuzzy logic and fuzzy inference systems were introduced by Zadeh (1965, 1971, 1973, 1974). Zadeh (1972) also proposed the initial idea of applying fuzzy sets to control problems. However, the actual research on fuzzy controllers was initiated by Mamdani (1976). Mamdani’s work influenced many others to explore the applicability of fuzzy controllers to various control problems (Holmblad & Ostergaard 1982; Maiers & Sherif 1985; Sugeno & Park 1993; Zhang et al. 1993). In recent years, research on fuzzy systems has been focussing on investigating connections between fuzzy systems and neural networks, as well as exploring the integration of rule-based and model-based approaches in fuzzy control, which are exemplified by the work of Filev (1991; 1992) and Sugeno and Yasakawa (1993).

A fuzzy inference system is a process of formulating the mapping from a given input to an output using fuzzy logic. The mapping then provides a basis from which decisions can be made, or patterns discerned. Fuzzy logic starts with the concept of a fuzzy set. A fuzzy set is a set without a crisp, clearly defined boundary. Unlike a classical set, which is a container that wholly includes or wholly excludes any given element, in fuzzy sets an element can belong to a set partially. The
The degree of membership is defined through a generalized characteristic function called the *membership function*. The values of the membership function are real numbers in the interval [0,1], where 0 means that the element is not a member of the set, 1 means that it belongs to the set entirely, and values in between represent that it belongs to the set with a partial degree of membership. Figure 2.5 shows three membership functions representing three fuzzy sets labelled as “young”, “middle” and “old”, all of them being fuzzy values of a variable “age”. As we can see, the age 30 belongs to the fuzzy set “middle” to a degree of 0.7, to the “young” set to a degree of 0.3 and to the “old” set to a degree of 0.

**Figure 2.5.** Membership functions representing three fuzzy sets for the variable “age”

The only condition a membership function must satisfy is that it must vary between 0 and 1. The function itself can be an arbitrary curve whose shape we can define as a function that suits us from the point of view of simplicity, convenience, speed, and efficiency. The simplest membership functions are the *triangular* membership functions, which use three points and straight lines to form triangles as shown in Figure 2.5. The *trapezoidal* membership function has a flat top and is just a truncated triangle curve. Other commonly used membership functions include the Gaussian distribution function, the sigmoid curve, and quadratic and cubic polynomial curves. For detailed information on any of these membership functions, the reader should refer to Kilr and Yuan (1995).
Now how are these fuzzy concepts used in connection with logic? Suppose there are two fuzzy sets \( A \) and \( B \). A common prescription for logical operations using fuzzy concepts is as follows:

- Intersection (AND):
  \[
  \mu_{A \cap B}(x) = \text{minimum} \left( \mu_A(x), \mu_B(x) \right)
  \]  
  (2.10)

- Union (OR):
  \[
  \mu_{A \cup B}(x) = \text{maximum} \left( \mu_A(x), \mu_B(x) \right)
  \]  
  (2.11)

- Set complement (NOT):
  \[
  \mu_{\neg A}(x) = 1.0 - \mu_A(x)
  \]  
  (2.12)

where \( \mu_A(x) \) is the membership degree of \( x \) in \( A \), and \( \mu_B(x) \) is the membership degree of \( x \) in \( B \).

Some researchers in fuzzy logic have explored the use of other interpretations of the AND and OR operations. For example, the OR operation can also be implemented by the probabilistic OR method, which is defined by the following equation:

\[
\mu_{A \cup B}(x) = \mu_A(x) + \mu_B(x) - \mu_A(x) \times \mu_B(x)
\]  
 (2.13)

Fuzzy sets and fuzzy operations are the subjects and verbs of fuzzy logic. If-then rule statements are used to formulate the conditional statements that comprise fuzzy logic. A single fuzzy if-then rule assumes the form

\[
\text{if } x \text{ is } A \text{ then } y \text{ is } B
\]

where \( A \) and \( B \) are fuzzy values defined by fuzzy sets. An example of such a rule might be

\[
\text{if service is good then tip is average}
\]

Note that \textit{good} is represented as a fuzzy set, and so the antecedent is an interpretation that returns a single membership degree between 0 and 1. On the other hand, \textit{average} is also represented as a fuzzy set, and the consequent is an assignment that assigns the entire fuzzy set \textit{average} to the output variable \textit{tip}. This set will later be defuzzified, assigning one value to the output. The concept of defuzzification is described later in this section.
The antecedent of a rule can have multiple parts

\[
\text{if service is excellent or food is delicious then tip is generous}
\]

In this case all parts of the antecedent are calculated simultaneously and resolved to a single number using the logical operators described earlier. The consequent of a rule can also have multiple parts:

\[
\text{if temperature is cold then hot water valve is open and cold water valve is shut}
\]

in which case all consequents are affected equally by the result of the antecedent. How is the consequent affected by the antecedent? The consequent specifies a fuzzy set be assigned to the output. The implication function then modifies that fuzzy set to the degree specified by the antecedent. The most common way to modify the output fuzzy set is truncation using the min function (where the fuzzy set is “chopped off” as shown in Figure 2.6).
If service is excellent or food is delicious then tip = generous

1. Fuzzify inputs

- service (crisp) with u(service==excellent) = 0.0
- food (crisp) with u(food==delicious) = 0.7

2. Apply OR operator (max)

If (0.0 or 0.7) then tip = generous

max(0.0, 0.7) = 0.7

3. Apply implication operator (min)

If (0.7) then tip = generous

min(0.7, generous)

Figure 2.6. An example of interpreting one fuzzy rule
Figures 2.6 and 2.7 show the examples of interpreting one or more fuzzy rules in a fuzzy inference system. The process consists of five parts: fuzzify inputs, apply fuzzy operator to multiple part antecedents, apply implication method, aggregate all outputs, and defuzzify. The details are as follows:

1. **Fuzzify inputs**: Resolve all fuzzy statements in the antecedent to a degree of membership between 0 and 1. In other word, this step involves taking crisp numerical input values and determining the degree to which they belong to each of the appropriate fuzzy sets via membership functions. For example, Figure 2.6 shows how well the food at our hypothetical restaurant (rated on a scale of 0 to 10) qualifies with respect to the fuzzy variable “delicious”. In this case, the food was rated as an 8, which, given the graphical definition of “delicious”, corresponds to $\mu = 0.7$ for the “delicious” membership function.
2. Apply fuzzy operator to multiple part antecedents: Once the inputs have been fuzzified, we know the degree to which each part of the antecedent has been satisfied for each rule. If the antecedent of a given rule has more than one part, the fuzzy operator is applied to obtain one number that represents the result of the antecedent for that rule. This number will then be applied to the output function. The input to the fuzzy operator is two or more membership values from fuzzified input variables. The output is a single truth value. This is the degree of support for the rule. Figure 2.6 shows an example of the OR operator max at work. The two different pieces of the antecedent (service is excellent and food is delicious) yielded the fuzzy membership values 0.0 and 0.7 respectively. The fuzzy OR operator simply selects the maximum of the two values, 0.7, and the fuzzy operation for this antecedent is completed.

3. Apply implication method: Use the degree of support for the entire rule to shape the output fuzzy set. The consequent of a fuzzy rule assigns an entire fuzzy set to the output. This fuzzy set is represented by a membership function that is chosen to indicate the qualities of the consequent. If the antecedent is only partially true, (i.e., is assigned a value less than 1), then the output fuzzy set is truncated according to the implication method.

4. Aggregate all outputs: In general, one rule by itself has a limited effectiveness. What is needed are two or more rules that can work together. The output of each rule is a fuzzy set. The output fuzzy sets for each rule are then aggregated into a single output fuzzy set. Two commonly used aggregation methods are: max (taking the maximum over all of the output sets of rules), and sum (simply the sum of each rule’s output set) (Dyckhoff & Pedrycz 1984; Dubois & Prade 1985). In Figure 2.7, three fuzzy rules have been placed together to show how the output of each rule is combined, or aggregated, into a single fuzzy set by using the max method.

5. Defuzzify: The input for the defuzzification process is a fuzzy set (the aggregate output fuzzy set) and the output is a single number. The aggregate of a fuzzy set encompasses a range of output values, and so must be defuzzified in order to resolve a single output value from the set. Two of the common defuzzification methods are the centroid and maximum methods. In the centroid method, the crisp value of the output variable is computed by finding the variable value of the center of gravity of the aggregate output fuzzy set. In the maximum method, one of the variable values at which the aggregate output fuzzy set has its maximum truth value is chosen
as the crisp value for output variable. There are several variations of the maximum method that differ only in what they do when there is more than one variable value at which this maximum truth value occurs. One of these, the average-of-maximum method, returns the average of the variable values at which the maximum truth value occurs. Figure 2.7 shows how a crisp value (tip = 16%) is generated by the average-of-maximum method.

2.3.3 Membership Function Construction Methods in Fuzzy Systems

Since the notion of fuzzy sets was introduced by Zadeh, one of the main difficulties in the field has been with the construction of membership functions (Bilgic & Turksen 1997). Almost all membership function construction methods are based on an expert’s judgement and can be classified into direct methods and indirect methods (Kilr & Yuan 1995). In direct methods, experts are expected to give answers to questions of various kinds that explicitly pertain to the constructed membership function. In indirect methods, experts are required to answer simpler questions, easier to answer and less sensitive to the various biases of subjective judgement, which pertain to the constructed membership function only implicitly. The answers are subject to further processing. Both direct and indirect methods can be further classified to methods that involve one expert and methods that require multiple experts. This results in four principle classes of methods for constructing membership functions: direct methods/one expert, direct methods/multiple experts, indirect methods/one expert, and indirect methods/multiple experts. Among these four principle classes of methods, the following six methods have been commonly used (Norwich & Turksen 1982; Chameau & Santamarina 1987; Turksen 1991): Polling (Hersh & Carmazza 1976), Direct Rating (Hersh & Carmazza 1976), Reverse Rating (Turksen 1988), Interval Estimation (Chameau & Santamarina 1987), Membership Exemplification (Hersh & Carmazza 1976), and Pairwise Comparison (Kochen & Badre 1974).

Unless a membership function can be defined in terms of a suitable similarity function with respect to an ideal prototype, the above methods have one fundamental disadvantage (Kilr & Yuan 1997): they require an expert (or experts) to give answers that are precise (or close enough) to capture subjective judgements. This is often infeasible for complex concepts. In addition, the answers given by the experts are always somewhat arbitrary, which can often lead to inappropriate membership functions. It would be preferable to automatically construct membership functions from the available data. Some efforts have been made in this direction:
Constructions based on mathematical functions. Methods for transforming membership functions from mathematical functions were explored by Devi and Sarma (1985), Civanlar and Trussell (1986), and Dubois and Prade (1986). An advantage of these methods is that the membership function matches the sample data exactly. Its disadvantage is that the complexity of the resulting mathematical function increases with the number of data samples.

Constructions by neural networks. Takagi & Hayashi (1991) discuss a neural network that generates nonlinear, multi-dimensional membership functions which is a membership function generating module of a larger system that utilized fuzzy logic. Yamakawa and Furukawa (1992) presented an algorithm for learning membership functions using a model of the fuzzy neuron. Their method uses example-based learning and optimization of cross-detecting lines. They assign trapezoidal membership functions and automatically come up with its parameters.

Constructions by a genetic algorithm (GA). The work carried by Mang et al. (1995) proposed an algorithm to use the genetic algorithm (Goldberg 1989a) for adaptation of membership functions. The method uses the GA to make small changes to the width and centre positions of the membership functions.

An often mentioned drawback of neural networks and genetic algorithms is their non-deterministic nature. When the neural network or the genetic algorithm is initialized with different configurations, it is likely that the training or search process terminates at a different local minimum. Thus different membership functions can be produced when neural network-based or GA-based construction methods are employed. These different membership functions may cause confusion for humans. Further more, both the GA and neural networks can be computationally expensive.

In Chapter 5, a novel approach of employing a $\chi^2$ statistic-based feature selection algorithm to automatically construct membership functions is presented. It provides an alternative approach to other approaches and might be more appropriate and efficient for constructing membership functions for some tasks.
2.3.4 Fuzzy Neural Networks

The interconnection between fuzzy systems and neural networks was originally recognized by Kosko (1992) and further investigated by Buckley (1993), Buckley and Hayashi (1993), and Buckley et al. (1993). These researches suggest that neural networks are eminently suited for approximating fuzzy controllers and other types of fuzzy systems, as well as for implementing these approximations in appropriate hardware. Although classical neural networks can be employed for this purpose, attempts have been made to develop alternative neural networks, more attuned to the various procedures of approximate reasoning. These alternative neural networks are usually referred to as fuzzy neural networks. Various fuzzy neural networks (Lee & Lee 1974; Keller & Hunt 1985; Pedrycz 1991; Carpenter et al. 1992; Gupta & Qi 1992; Horikawa et al. 1992; Pal & Mitra 1992; Simpson 1992; Yager 1992; Hayashi et al. 1993; Kasabov 1996) have been suggested, which use neural network architectures to simulate fuzzy inference systems.

The series of steps to implement a basic fuzzy neural network system are: (1) convert real-valued data into a fuzzified representation; (2) train the neural network with the fuzzified information; and then (3) de-fuzzify the result to produce real values of the desired output. After the system is trained to satisfaction, fuzzy rules can be extracted from the trained neural network.

The model FuNN (Kasabov 1996) that is employed in this thesis consists of five layers of nodes: an input variable layer; a condition elements layer (input fuzzy membership function); a rule layer; an action elements layer (output fuzzy membership function); and an output layer. These elements are shown schematically in Figure 2.8. Ordinarily, triangular membership functions have been employed as models for assigning weights for the second and fourth layers.
The input layer represents the input variables. In the condition elements layer, each node represents a fuzzy variable of an input variable, such as “delicious” or “rancid”. The condition element layer performs fuzzification. The activation values of the nodes in the condition layer represent the membership degrees to which the input value belongs to each of the appropriate fuzzy sets of the respective input variables. In the simplified scheme shown in Figure 2.8, each of the two inputs can be fuzzified in the condition layer by showing the degree of their membership in a fuzzy set (such as the degree to which they are “delicious” or “rancid”). The number of fuzzy variables need not be the same for the various inputs. Thus inputs could be connected to two condition layer nodes (as shown in Figure 2.8, X1 is connected to A1 and A2, and X2 is connected to B1 and B2), and other inputs could be connected to, say, four condition layer nodes.

In the rule layer, one node represents a single complex fuzzy rule. The function of the rule layer is to apply the fuzzy operators to multiple part antecedents. The semantic meaning of the activation value of a node in this layer is that it represents the degree to which the input data matches the antecedent part of the fuzzy rule represented by this node.

A node of the action elements layer represents a fuzzy variable of an output variable. The function of this layer is to apply the implication method. The activation of the node represents
the degree to which this membership function is supported by all fuzzy rules together, so this is the level to which the membership function for this fuzzy variable is “chopped off” according to the rules and current facts. The output layer combines the output fuzzy sets for each rule into a single fuzzy set, and then performs a defuzzification which converts the fuzzy set to a single real number.

There are two versions of the FuNN according to the mode of training and adaptation (Ruan 1997):

1. Fixed version: when the FuNN fuzzy neural network is trained, the weights connecting the outer layers (input layers and output layer) do not change; a backpropagation algorithm is used to adjust the connection weights of the remaining part of the FuNN (condition elements layer, rule layer and action elements layer) in order to achieve the desired performance.

2. Adaptive version: the weights connecting the outer layers are also adjusted; in this mode all the connection weights change during training. Adjustment of these outer layers means that the triangular membership functions that are used in a FuNN to represent fuzzified descriptions of the input variables in the condition elements layer and output variables in the action elements layer are adjusted during the training of the net, but this is normally a separate process.

Three methods (Kasabov 1993b) for producing the membership functions employed by FuNN are as follows:

- **A Fixed Centre-based Membership Function Selection Approach**

A centre-based MF selection approach can be implemented by using three-point triangular membership functions as shown in Figure 2.5. The membership functions are spaced equally according to the minimum and maximum values of the input data. With these triangular membership functions each input value will belong to no more than two fuzzy sets, and their membership degrees will always sum to one. Though this approach is straightforward, the division into equally spaced membership functions may be naïve and inappropriate for some data sets.
Manual Adjustment of Centres of Membership Functions

For triangle-shaped MFs, the centres can be adjusted manually. As shown in Figure 2.9, heavy solid lines represent the initial membership functions of a variable $x$, and light solid lines represent the asymmetrical membership functions after adaptation. In this case a rigid partitioning is often used to establish a region within which each centre can move but not cross. The adjustment must satisfy the restrictions imposed on the membership functions (the movements of their centres cannot take them out of the membership function’s partition). Generally, manual alterations only adjust the membership functions slightly.

Genetic Algorithm for Adaptation of Membership Functions

The fuzzy membership functions in a FuNN can also be adjusted by an approach based on a genetic algorithm. Since triangular membership functions are employed, only the centres need to be represented in the chromosome of the GA module, making the computation not too expensive. The permitted amount of movement of these centres is also restricted by setting fixed limits along the horizontal axis (see Figure 2.9).

2.4 Feature Selection Methods

Features, which are also called attributes, properties, or characteristics, can have discrete or continuous (numeric) values. A collection of features with their values forms a flat data file that
describes an application in which each line describes an instance. An instance is a pair \((x, f(x))\), where \(x\) is an \(N\)-dimensional feature vector, and \(f(x)\) is one of the predefined classes (categories). In real-world applications, some of the \(N\) original features are often completely irrelevant/redundant to the target concept \(f(x)\). An irrelevant feature does not affect the target concept in any way, and a redundant feature does not add anything new to the target concept. Feature selection is a process that picks a subset of features that are relevant to the target concept. The rest of this section offers an overview of the various feature selection methods, and particularly discusses a \(\chi^2\) statistic-based discretization algorithm, called the Chi2 algorithm.

2.4.1 Overview

Most feature selection methods can be grouped into two categories: exhaustive or heuristic search of an optimal set of \(M\) attributes. For example, Almuallim and Dietterich’s FOCUS algorithm (1994) starts with an empty feature set and carries out exhaustive search until it finds a minimal combination of features that is sufficient to construct a hypothesis consistent with a given set of examples. It works on binary, noise-free data. Its time complexity is \(O(\min(N^M, 2^N))\). They proposed three heuristic algorithms to speed up the search.

There are many heuristic feature selection algorithms. The Relief algorithm (Kira & Rendell 1992) assigns a ‘relevance’ weight to each feature, which is meant to denote the relevance of the feature to the target concept. Relief samples instances randomly from the training set and updates the relevance values based on the difference between the selected instance and the two nearest instances of the same and opposite classes. According to Kira and Rendell, Relief assumes two-class classification problems and does not help with redundant features. If most of the given features are relevant to the concept, it would select most of them even though only a fraction maybe necessary for concept description. The PRESET algorithm (Modrzejewski 1993) is another heuristic feature selector that assumes a noise-free binary domain and uses the theory of Rough Sets (Pawlak 1991) to heuristically rank the features. Since PRESET does not try to explore all combinations of the features, it is certain that it will fail on problems whose attributes are highly correlated where the combinations of a few attributes do not help in finding the relevant attributes.
According to Liu & Motoda (1998), the exhaustive search approach may be infeasible in practice and the heuristic search approach can reduce the search time significantly, but may fail on hard problems and cannot remove redundant attributes.

There are other approaches to determine the most significant feature(s) through information measures (Shannon 1948; Wiener 1948). From the information measures point of view, information is a way of measuring the uncertainty of the receiver when he receives all messages. If the receiver knows what is coming, his expected surprise level (uncertainty) is low; if he does not know at all what is coming, a reasonable assumption is that all messages have almost equal probabilities to come, his expected surprise is high. In the context of classification, messages are classes. An information measure $U$ is referred to as the uncertainty function concerning the true class, and is defined so that larger values for $U$ represent higher levels of uncertainty. A feature should be selected if it can reduce uncertainty. A commonly used uncertainty function is Shannon’s entropy (Shannon 1948). Many other uncertainty functions have also been suggested. They were briefly reviewed by Ben-Bassat (1982). The idea of information measures is used in ID3 (Quinlan 1986) and C4.5 (Quinlan 1993) for selecting a feature to grow a decision tree.

In recent years, the broad interest in data pre-processing has lead to another active research field: feature transformation (Liu & Motoda 1998). One of the variants of feature transformation is feature construction. Feature construction is a process that discovers missing information about the relationships between features and augments the space of features by inferring or creating additional features (Matheus 1991; Wnek & Michalski 1994). For example, a two dimensional problem (say, $A_1 = \text{width}$ and $A_2 = \text{length}$) may be transformed to a one-dimensional problem ($B_1 = \text{area}$) after $B_1$ is discovered.

Some learning algorithms have built-in feature selection, for example, ID3 (Quinlan 1986), FRINGE (Pagallo & Haussler 1990) and C4.5 (Quinlan 1993). The experimental results in (Almuallim & Dietterich 1994) suggest that one should not rely on decision tree induction methods for feature selection, since these learning algorithms (e.g., ID3 or FRINGE) select irrelevant features on some data sets, which can lead to a deterioration of predictive accuracy.
The use of neural networks as a feature selector was reported by Setiono and Liu (1997a). They proposed a method that uses a three-layer feed-forward neural network as a tool to determine which attributes are to be discarded. The neural network is trained with the complete set of attributes as input. For each attribute in the network, the accuracy of the network is computed with all the weights of the connections associated with this attribute set to zero. The attribute that gives the smallest decrease in the network accuracy is removed. The network is then retrained and the process is repeated. A network pruning algorithm (Setiono 1997) is the foundation of the proposed method. By adding a penalty term to the error function of the network, redundant network connections can be distinguished from relevant ones by their small weights when the network training process has been completed. Setiono’s pruning algorithm and other penalty term-based pruning algorithms (Hanson & Pratt 1989; Ji et al. 1990; Weigend et al. 1991) require the value of the pruning-threshold $\eta^2$ to be specified manually: a value of $\eta^2$ that is too large or too small will over- or under-prune the weights of the network. An extreme example of under-prune is when all weights remain in the network. Over-prune will lead to the elimination of too many weights, thus causing the accuracy level of the network to drop too much. It is therefore ideal to let the weights determine what value $\eta^2$ should take in order to avoid over- or under-removal of the weights. In Chapter 4, three different mechanisms to automatically select $\eta^2$ are proposed.

Feature selection algorithms can also be divided based on the data types on which they operate. Many feature selection algorithms (Ross et al. 1994; Kira & Rendell 1992) have been crafted to work effectively on discrete data or, even more strictly, on binary data. In order to work with data having numeric attributes, a common practice for those algorithms is to discretize the data before conducting feature selection. The Chi2 algorithm (Liu & Setiono 1995a) provides a way to select features directly from numeric attributes while discretizing them. The Chi2 algorithm stems from Kerber’s ChiMerge (Kerber 1992) algorithm which is designed to discretize numeric attributes based on the $\chi^2$ statistic. ChiMerge requires a user to specify a proper significance level ($\alpha$) which is used for merging values of all the attributes. The Chi2 algorithm extends ChiMerge by automatically selecting $\alpha$ until further merging is discontinued by the stopping criteria. Furthermore, the Chi2 algorithm's capability to perform not only discretization but also feature selection has made it a big step forward from ChiMerge. The Chi2 algorithm is described in detail below.
2.4.2 A χ² Statistic-based Discretization Algorithm - the Chi2 Algorithm

The Chi2 algorithm is a general algorithm that uses the χ² statistic to discretize numeric attributes repeatedly until some inconsistencies are found in the discretized data. By repeated application of the approach, a reduced set of features can be achieved.

The method reduces the complexity of data in two dimensions: a vertical dimension (number of instances) and a horizontal dimension (number of features). The χ² statistic is employed to continue discretizing the numeric attributes until the discriminating power of the original continuous-valued data cannot be maintained. This step significantly reduces the possible data space from a continuum to discreteness according to the characteristics of the data by merging attribute values into sets of intervals. After discretization, duplicates may occur in the data. Removing these duplicates reduces the amount of data. Hence, the original database, if viewed as a large table, is shortened in its vertical dimension. Horizontal reduction is achieved by attribute selection which is accomplished by retaining only those attributes having more than one discrete value.

The goal of the Chi2 algorithm is gradually to place the data items into a contiguous set of intervals, each of which contains a set of items. For the purpose of later analysis, all the data items in a given interval are considered to be essentially the same. Initially each data item is considered to be in its own, unique, interval. As the algorithm is applied, adjacent intervals are merged and so that they will include several data items.

The Chi2 algorithm is oriented around the χ² statistic and consists of two phases. In the first phase each attribute i is associated with some significance level, say 0.5 at the outset. The data values of this attribute are sorted, and each value is considered to be a resident of an (initially single-valued) interval. Then the χ² value is calculated for each pair of adjacent intervals. The formula for computing the χ² value is:

\[
χ^2 = \sum_{i=1}^{2} \sum_{j=1}^{k} \frac{(A_{ij} - E_{ij})^2}{E_{ij}}
\] (2.14)
where

\[ k = \text{number of classes}, \]
\[ A_{ij} = \text{number of patterns in the } i\text{-th interval, } j\text{-th class}, \]
\[ R_i = \text{number of patterns in the } i\text{-th interval}, \]
\[ C_j = \text{number of patterns in the } j\text{-th class}, \]
\[ N = \text{total number of patterns}, \]
\[ E_{ij} = \text{expected frequency of } A_{ij}, \quad E_{ij} = R_i \times C_j / N. \]

Starting with the lowest \( \chi^2 \) values, adjacent intervals are merged until all pairs of intervals have \( \chi^2 \) values greater than the \( \chi^2 \) value associated with the current significance level for the given attribute. This is done for each attribute. These merged intervals now represent a discretization of the data set. With the reduced number of intervals, it is possible that there are some inconsistencies (two identical data elements associated with different output class values) in the data set. If the number of observed inconsistencies remains below a user set value, \( \delta \), the above process is repeated with a decremented significance level for the attributes (and hence a larger tolerated \( \chi^2 \) value). At the end of the first phase, the data set is discretized, and the number of data elements has been reduced.

In the second phase of the Chi2 algorithm, each attribute \( i \) is associated with an individual significance level, \( \text{sigLevel}[i] \), and takes its turn in the merging process. Consistency checking is conducted after each attribute’s merging; if the consistency constraint is exceeded, attribute \( i \) will not participate in further merging. The second phase ends when no attributes can be further merged. If during this process an attribute has been merged to a single interval, then it means that attribute (feature) is not useful for data discrimination and can be dropped from further consideration. By this reduction, feature selection has been achieved.

There are some limitations of the Chi2 algorithm (Liu & Setiono 1995b). It can only be used to discretize data and select features for supervised learning tasks, since class information is vital in the use of the \( \chi^2 \) statistic. In addition, it only works on numeric attributes; if there are mixed (numeric and discrete) attributes, the Chi2 algorithm can be specified to operate only on the numeric attributes for discretization and feature selection.
Experimental results (Liu & Setiono 1995b) have shown that, as a tool of data pre-processing, Chi2 serves as a useful means for feature selection and the discretization of numeric attributes. Its effectiveness will be further demonstrated by means of experiments on six practical data sets reported in Chapter 4.

2.5 Rule Extraction Methods

It is often desirable to have a set of meaningful and coherent rules that reveal hidden relationships that lie buried within data sets. Such rules are a form of knowledge that can be verified by human experts, and then passed on and expanded. The elaboration of such rules extracted from data can lead to improved theories concerning the application domain.

2.5.1 Overview

There are many ways to characterize rule extraction algorithms. Since neural networks have been widely applied to solve learning problems, one approach used here is to divide rule extraction algorithms into two classes based upon whether or not neural network technology is used.

In the past few years, attempts have been made at finding effective algorithms to extract rules from trained feed-forward neural networks:

- The KT algorithm (Fu 1994) searches for subsets of connections to a network node such that summed weights exceed the bias associated with that node. It is assumed that the node’s activation value is close to either 0 or 1. By searching for the proper subsets of the input connections, sets of rules are generated to describe under what conditions the node’s activation will take one of the two values.

- The M-of-N algorithm (Towell & Shavlik 1993) clusters the weights of a trained network into equivalence classes. The complexity of the network is reduced by eliminating unneeded clusters and by setting all weights in each remaining cluster to the average of the cluster’s weights. Rules with weighted antecedents are obtained from the remaining links of the simplified network.

- More recently, a method that uses sampling and queries was proposed (Craven & Shavlik 1994). Instead of searching for rules from the network, the problem of rule extraction is
viewed as a learning task. Given a trained network, this method uses queries to induce a decision tree that approximates the function represented by the model.

- Thrun (1995) described a rule extraction algorithm called Validity Interval Analysis (VIA) that divides the activation range of each node into intervals. The problem of determining the boundary of these intervals is considered as a linear programming problem. Thrun uses two methods to explore a space of rules: specific-to-general and general-to-specific. For tasks with numeric features, he uses the specific-to-general method that starts with training examples as seeds for rules. Each of these initial rules describes a point in the instance space. The rules are iteratively generalized by converting one of the rule’s literals into an interval, or by increasing the bounds of an interval in an existing rule.

- Setiono and Liu (1995) developed a rule extraction algorithm which was based on the fact that it is generally possible to replace the continuous activation values of hidden nodes by a small number of discrete values. Rule extraction is processed in two steps. First, rules that describe the network outputs in terms of the discretized activation values of the hidden nodes are generated. Second, rules that describe each discretized hidden node activation value in terms of the network inputs are constructed. By merging the rules obtained in these two steps, a set of rules that relates the inputs to the outputs of the network is obtained. While the algorithm can generate symbolic rules that mimic the predicted outcome of the original network, it works only for data with binary inputs (the implicit assumption is that the various hidden node activation values are determined by only a small number of input values, and excludes problems with continuous attributes where there can be infinitely many possible values taken by these attributes.). A further development of the algorithm, called NeuroLinear, was presented by Setiono and Liu (1997b). The NeuroLinear algorithm follows a two-step rule extraction process which generates rules that describe the relationship from the hidden layer to the output layer and from the input layer to the hidden layer. Because each rule condition is given in the form of a linear inequality

$$\sum_i c_i x_i < \eta$$  \hspace{1cm} (2.15)

where $c_i$ is a real coefficient, $x_i$ the value of the attribute $i$, and $\eta$ a threshold, continuous attributes can be used as inputs without requiring discretization. A detailed description of the algorithm will be given later in Section 2.5.4.
Parallel to the development of techniques for extracting rules from trained feed-forward neural networks, there are several techniques available to extract fuzzy rules from fuzzy neural networks. Horikawa et al. (1992) developed three types of fuzzy neural networks which can automatically identify the underlying fuzzy rules and tune the corresponding membership functions by modifying the connection weights of the neural networks using the backpropagation algorithm. Another method, ReFuNN, is a simple approach for rules extraction from a trained FuNN module. This will be explained later in Section 2.5.5.

Many existing approaches are available to extract knowledge directly from raw data without using neural networks. Shigeo and Lan (1995) proposed a method for extracting fuzzy rules directly from numerical input-output data for pattern classification. The GABIL system (De Jong et al. 1990) uses a genetic algorithm to learn and refine concept classification rules from its interaction with the environment. Decision trees involve a recursive partitioning of the feature space, based on a set of rules that are learned by an analysis of the training set. A tree structure is developed where, at each branching, a specific decision rule is implemented, which may involve one or more combinations of the attribute inputs. A new input vector then “travels” from the root node down through successive branches until it is placed in a specific class. In essence then, the classification is determined by describing the path from the root node of the tree to a leaf node – each nodal set of rules progressively refining the classification in a hierarchical manner. The X2R algorithm (Liu & Tan 1995) is a straightforward technique for generating a set of rules based on a data set with discrete attributes. Continuous-valued data must first be discretized before applying the X2R algorithm. Since Decision trees, the X2R algorithm, the NeuroLinear approach, and the ReFuNN approach have been demonstrated to be practical tools to analyze data (Bradshaw et al. 2001; Bradshaw et al. 2002; Zhou et al. 2001), and will be applied extensively in the experiments reported in the later chapters, more detailed descriptions will be given in the following four sections. Their applications to solving real-world problems will be presented in Chapter 8.

2.5.2 Decision Trees
In this section, we focus on the most widely used decision tree induction algorithm, C4.5. Figure 2.10 depicts an example decision tree for the problem domain of heart disease diagnosis which concerns medical cases of heart diseases (UCI 1998). As shown in the figure, a decision tree is
a rooted, directed acyclic graph consisting of a set of internal nodes (depicted as rectangles) and a set of leaves (depicted as ovals). Each internal node in a decision tree has an associated logical test based on the features in the domain. When classifying an example, the role of an internal node is to send the example down one of the outgoing branches of the node. The decision as to which branch an example is sent down is determined by the logical expression at the node. For C4.5, this expression considers a single feature, and thus the outcome of the test is determined by the value of that feature in the given example. In some decision trees, the test may be a function of several features. In Figure 2.10, each internal node tests a single feature, and the outgoing branches are labelled with the possible outcomes for a given test. For example, the root of the tree looks at the feature “rest ECG”, which has three possible values: “abnormal”, “normal”, and “hypertrophy”. Similarly, the rightmost child of the root node tests the real-valued feature “cholesterol” against a threshold of 200.

The classification procedure involves starting at the root of the tree, and then traversing a path through the tree that is determined by the outcomes of the tests at the internal nodes encountered along the path. The leaves of a decision tree do not have logical tests, but instead have associated class labels; in the figure, leaves are labelled either with the class “disease” or with “okay”. When an example reaches a leaf, the class associated with the leaf is the prediction made by the decision tree for that example.
After C4.5 has grown a tree, it then tries to simplify it by pruning away various subtrees and replacing them with leaves. C4.5's pruning method is called *Error-based Pruning* (Quinlan 1986), and it considers replacing each internal node by either a leaf or one of the node’s branches. In order to decide if a change should be made, C4.5 computes a confidence interval around the resubstitution (training) error rate of the node. A change should be made to a subtree if the resulting resubstitution error rate for the modified subtree is within a $C\%$ confidence interval of the unmodified subtree’s error rate, where $C$ is a parameter of the algorithm that determines how conservative the pruning process should be. Apart from error-based pruning, there are a number of other techniques (Breiman *et al.* 1984; Niblett & Bratko 1986; Mingers 1987; Quinlan 1987) for pruning decision trees; a comparative analysis of these techniques can be found in Esposito *et al.* (1997).

After a tree has been pruned, a collection of rules can be generated from the tree. The key ideas are:

- Every path from the root of a tree to a leaf gives one initial rule. The left-hand side of the rule contains all the conditions established by the path, and the right hand side specifies the class at the leaf.
- Each such rule is simplified by removing conditions that do not seem helpful for discriminating the nominated class from other classes, using a pessimistic estimate of the accuracy of the rule.
- For each class in turn, all the simplified rules for that class are sifted to remove rules that do not contribute to the accuracy of the set of rules as a whole.
- The set of rules for the classes are then ordered to minimize false positive errors and a default class is chosen.

### 2.5.3 X2R: A Fast Rule Generator

For a data collection that represents a set of discrete inputs and outputs, one can generate a set of rules that summarize this data set. The X2R algorithm (Liu & Tan 1995) is a straightforward technique for generating such a set of rules. In order to use it, one must have discrete inputs and outputs, so if one is dealing with continuous-valued data, the data must first be discretized by some standard technique. The X2R algorithm proceeds in three steps:
1. Generate a rule to cover the most frequently occurring input pattern. This is the shortest rule that can differentiate the input pattern from input patterns associated with other output classes. Then remove this input pattern from further consideration and iteratively repeat this step.

2. The generated rules are then grouped in terms of their output class labels.

3. For each rule cluster, remove redundant rules and drop more specific rules in favor of more general rules for the cluster. A default rule is chosen in the case when no rule can be applied to an input pattern.

This procedure generates an ordered set of rules (the rules must be applied in a specific order).

2.5.4 NeuroLinear Approach
The NeuroLinear approach (Setiono & Liu 1997b) is an algorithm to extract oblique decision rules from trained feed-forward neural networks and thereby derive knowledge about the mapping of inputs to outputs that is more readily understandable. The NeuroLinear rule extraction is performed in the following steps:

1. Select and train a feed-forward, multi-layer perceptron neural network to meet a prespecified accuracy requirement. Use the hyperbolic tangent function as the hidden layer activation function. The pruning algorithms can then be used to remove the redundant connections while maintaining the network’s performance.

2. Use the Chi2 algorithm to discretize the activation values of the hidden neural network nodes.

3. Extract a set of rules that reflects the relationship from the hidden layer to the output layer in the network.

The rules that describe the network outputs in terms of the discretized network activation values (also named as “the discrete intermediate outputs”) are extracted. The X2R approach can then further generate a set of order-sensitive rules which cover the most frequently occurring patterns presenting in the rules.
4. Extract a second set of rules that reflects the relationship from the input layer to the hidden layer in the network.

The rules that describe each discretized hidden node activation value in terms of the network’s inputs are extracted. Let \( N \) be the number of intervals found by the Chi2 algorithm for the activation values of hidden node \( H \). There are \( N+1 \) boundaries to form these \( N \) intervals: \(-1 = m_0 < m_1 < ... < m_N = 1\). The activation value of hidden node \( H \) falls into the interval \([m_{j-1}, m_j]\) if the sum of its weighted inputs satisfy

\[
tanh^{-1}(m_{j-1}) \leq \text{the sum of the weighted inputs} < \tanh^{-1}(m_j)
\] (2.16)

where \( \tanh^{-1}(x) \) is the inverse of the hyperbolic tangent function

\[
\tanh^{-1}(x) = \log((1 + x) / (1 - x)) / 2
\] (2.17)

5. With the two sets of rules, it is sometimes convenient to combine them into a single set of rules (from input values to output values).

For illustration purposes, next it is described how rules are extracted by NeuroLinear for a well-known Iris Data Classification (Fisher 1936). For a detailed description of the Iris Data Classification data set, the reader should refer to Chapter 3.

10-fold cross validation was applied to the 150 samples of iris data, which leads to the training set and the testing set for each partition containing 135 samples and 15 samples, respectively. For illustration purposes, one partition was randomly selected for the extraction of rules using NeuroLinear. A standard fully connected three layer feed-forward neural network was used. The number of input nodes was 4, and the number of nodes in the output layer was 3, as the three output attributes 0, 1, and 2 were coded as \{1, 0, 0\}, \{0, 1, 0\}, and \{0, 0, 1\}. We label the inputs \textit{sepal length}, \textit{sepal width}, \textit{petal length}, and \textit{petal width} as I0, I1, I2 and I3, respectively. The BFGS algorithm was used as the neural network training method. After the training and pruning
process, only 2 of the original 5 hidden nodes remained. The neural network made correct predictions of 98.52% on the training set and 93.33% on the test set.

Then the clusterization procedure, Chi2, was used to discretize the hidden node activation values associated with all 135 training sample. It found 2 intervals in each of the 2 remaining hidden nodes. The intervals of the 2 remaining hidden nodes were generated and are shown in Table 2.1.

**Table 2.1. Intervals of hidden nodes**

<table>
<thead>
<tr>
<th>Hidden Node</th>
<th>Intervals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hidden node 1</td>
<td>[-1, 0.50), [0.50, 1]</td>
</tr>
<tr>
<td>Hidden node 2</td>
<td>[-1, 0.75), [0.75, 1]</td>
</tr>
</tbody>
</table>

A unique integer value between 1 and 4 was given to represent one of these 4 intervals associated with hidden nodes. For example, 1 represents the first interval [-1, 0.50) at hidden node 1, 2 represents the second interval [0.50, 1] at hidden node 1, and 4 represents the last interval [0.75, 1] at hidden node 2. Thus, a new data set with 2 columns of discrete values was generated as shown in Table 2.2. The output classes for these samples were their original class labels, which were values ranging from 0 to 2. After removing all duplicates, only 4 unique samples remained. The frequencies associated with each unique sample are also shown in Table 2.2.

**Table 2.2. The combinations of the discretized activation values at the two remaining hidden nodes in the pruned network trained for the Iris Data Classification data set**

<table>
<thead>
<tr>
<th>Discretized Activation Values (Discrete Intermediate Outputs)</th>
<th>Output Class</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Hidden node 1</strong> (a₁)</td>
<td><strong>Hidden node 2</strong> (a₂)</td>
<td>0</td>
</tr>
<tr>
<td>unique sample 1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>unique sample 2</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>unique sample 3</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>unique sample 4</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>


Table 2.2 can be interpreted as a set of rules which describes the internal relationship from the hidden layer to the output layer in the neural network. For example, the first row in the Table 2.2 can be interpreted as:

**If** the discrete intermediate output \( a_1 = 2 \)

**and**

the discrete intermediate output \( a_2 = 3 \)

**then** output class = 0

where \( a_1 \) and \( a_2 \) represent the discrete activation value of the first hidden node, and the discrete activation value of the second hidden node, respectively. Then the above rule can be rewritten in its original term as:

**If** \( a_1 \in [0.50, 1] \) and \( a_2 \in [-1, 0.75) \) **then** output class = 0

An order-sensitive rule set (the rules should be fired in sequence) can be further generated by means of the X2R algorithm based on the frequencies of the samples shown in Table 2.2. The rules are as follows:

1. **If** the discrete intermediate output \( a_2 = 3 \)

**then** output class = 0
2. **If** the discrete intermediate output \( a_1 = 2 \)

**then** output class = 1
3. **Default rule** (output class = 2).

By using the weights of the pruned network and the boundaries of the intervals found by the Chi2 algorithm, another set of rules which maps from the original input attributes to the discrete activation values (discrete intermediate output) was generated. For instance, the condition the discrete intermediate output \( a_1 = 2 \) is satisfied by a sample if and only if the network’s first hidden node activation value is located in the interval [0.50, 1]. An activation value of a sample will be in this interval if and only if the weighted sum of its input is greater than or equal to \( \tanh^{-1}(0.50) = 0.55 \). Similarly, the condition the discrete intermediate output \( a_2 = 3 \) is satisfied by a sample if and only if the second hidden node activation values of a sample will fall
in the interval \([-1, 0.75)\), and this will occur if and only if the weighted sum of the sample’s input is less than \(\tanh^{-1}(0.75) = 0.97\).

Therefore, for hidden node 1, from the network connections we find that

**If** \(4.95 \times I_1 - 8.93 \times I_2 - 12.68 \times I_3 + 13.00 < 0.55\)
\[\text{then the discrete intermediate output } a_1 = 1\]

**If** \(4.95 \times I_1 - 8.93 \times I_2 - 12.68 \times I_3 + 13.00 \geq 0.55\)
\[\text{then the discrete intermediate output } a_1 = 2\]

Similarly, for hidden node 2, we have that

**If** \(2.58 \times I_3 < 0.97\)
\[\text{then the discrete intermediate output } a_2 = 3\]

**If** \(2.58 \times I_3 \geq 0.97\)
\[\text{then the discrete intermediate output } a_2 = 4\]

With these two levels of rule sets described above, a single set of rules from input values to output values can be easily obtained by combining them together. An example rule is shown below:

**If** \(2.58 \times I_3 < 0.97\) \text{ then output class } = 0

**2.5.5 ReFuNN Approach**

The ReFuNN (rules extraction from a fuzzy neural network) approach (Kasabov 1993a) is a simple method for extracting weighted fuzzy rules from FuNN module as described in Section 2.3.4 and illustrated in Figure 2.8. A set of rules \(\{r_j\}\) is extracted from a trained FuNN module as follows. All the connections to an action element neuron \(C_j\) that contribute significantly to its possible activation (their absolute values are above a defined threshold \(Th_a\)), are retained and their corresponding nodes \(R_j\) in the rule layer are analysed further. Only condition element nodes which support activating the chosen rule neuron \(R_j\) will be used in the antecedent part of a rule \(r_j\) (the absolute values of connection weights are above a threshold \(Th_c\)). The weights of the
connections between the condition element neurons and the rule nodes are taken as relative degrees of importance of the antecedent fuzzy propositions. The weights of the connections between a rule node \( R_j \) and an action elements node \( C_j \) define values for the certainty degree \( CF_j \). An example is shown below (see Figure 2.11). All input values range among the values A, B, C, D, and E, with A being the lowest value and E being the highest value. Both \( Th_a \) and \( Th_c \) are set at 2 for this illustration. The negative weights are represented by using “not” in the rule.

If  \(<\text{ALTITUDE is not A 16.482}>\text{ and } <\text{RAINFALL is not C 2.186}>\text{ and }<\text{TEMPERATURE is B 12.592}>\text{ and } <\text{DISTANCE is not C 8.566} >\text{ and } <\text{DISTANCE is D 5.019} >\)
then  \(<\text{SUITABILITY is A 13.469}>\text{ and } <\text{SUITABILITY is not B 19.502} >\).

**Figure 2.11.** Connection weights of a trained FuNN are interpreted as fuzzy rules by the ReFuNN algorithm

In order to generate a fuzzy rule set that achieves better inference performance and involves simpler rules (with fewer rule components) than that produced by the above method, a second
fuzzy rule extraction method was developed (Purvis et al. 1999). According to this method the FuNN is trained as before. Then all node connections that have an absolute value of weight below a certain threshold value $\delta$ were constrained to be zero. Then the fuzzy network is retrained. This process can be repeated, if necessary. Weighted fuzzy rules are then extracted. Like the first fuzzy rule extraction method, rule components in the consequent part are only derived from those connections to the action elements layer that have their weights above a certain threshold value $Th_a$. However for their corresponding rule nodes in the rule layer, only the strongest input connections from the condition elements layer are used in the antecedent part of a rule, with all other input connections constrained to be zero. When rules are extracted, each node in the rule layer represents a single fuzzy rule, which has only the strongest connection from each fuzzy input variable represented in its antecedent part. This will result in fewer components in the antecedent part of the rule than the above-described method.

A representative rule from this method shown below is taken from Purvis et al. (1999) when the threshold value $\delta$ was set to 0.5:

If $<$ALTITUDE is A 9.014$>$ and $<$RAINFALL is A 9.841$>$

and $<$TEMPERATURE is not B 5.095$>$ and $<$DISTANCE is A 10.63$>$

then $<$SUITABILITY is D 6.13$>$

The experimental results in Purvis et al. (1999) have shown that the second fuzzy rule extraction method has a superior performance over the first one in terms of number of extracted rules and predictive accuracy. In this thesis, for all experiments conducted using the ReFuNN approach, the second fuzzy rule extraction method was used.

The fuzzy inference procedure, which is used in connection with the derived fuzzy rule sets, is explained below:

* The overall degree of matching for the left-hand side of each rule is calculated, which is a weighted sum of the membership values to which input data belong to all its antecedent elements. A rule fires if and only if the overall matching degree of its antecedent part is positive.
• Then the degree to which each of the output membership functions is inferred collectively by all the rules is determined by calculating a weighted sum of all the certainty degrees associated with that output membership function from the activated rules.

This is illustrated in the following example, which we term, the Golf Course Problem. The description of this problem can be found in Chapter 3. The research goal here is to find fuzzy rules concerning four attributes (Altitude, Rainfall, Temperature, and Distance-To-Urban-Centre) that will identify ‘good’ sites for locating a golf course.

Suppose, for the Golf Course Problem, the altitude of a block is 267.8 meters, the rainfall is 2,400 mm per annum, the temperature is 10.5 degrees, and the distance is 260 kilometers. The membership function values of the input variables to which these data belong was found to be:

\[
\begin{align*}
\mu_A: \text{very low (Altitude)} &= 0.3 \\
\mu_B: \text{low (Altitude)} &= 0.7 \\
\mu_{C, \text{medium}}: \text{medium (Altitude)} &= \mu_{D, \text{high}}: \text{high (Altitude)} = \mu_{E, \text{very high}}: \text{very high (Altitude)} = 0 \\
\mu_A: \text{very light (Rainfall)} &= \mu_B: \text{light (Rainfall)} = \mu_{C, \text{medium}}: \text{medium (Rainfall)} = 0 \\
\mu_D: \text{heavy (Rainfall)} &= 0.9 \\
\mu_{E, \text{very heavy}}: \text{very heavy (Rainfall)} &= 0.1 \\
\mu_A: \text{very low (Temperature)} &= 1.0 \\
\mu_B: \text{low (Temperature)} &= \mu_{C, \text{medium}}: \text{medium (Temperature)} = \mu_{D, \text{high}}: \text{high (Temperature)} = \mu_{E, \text{very high}}: \text{very high (Temperature)} = 0 \\
\mu_A: \text{very near (Distance)} &= \mu_B: \text{near (Distance)} = \mu_{C, \text{somewhat distant}}: \text{distant (Distance)} = \mu_{D, \text{distant}}: \text{very distant (Distance)} = 0 \\
\mu_{E, \text{very distant}}: \text{very distant (Distance)} &= 1.0
\end{align*}
\]

Two extracted fuzzy rules are as follows (output variable Suitability ranges from A to E, with A representing Very Unsuitable and E representing Very Suitable):

\[
\begin{align*}
\text{If} & \quad <\text{ALTITUDE is not A} 16.482> \quad \text{and} \quad <\text{RAINFALL is not C} 2.186> \\
& \quad \text{and} \quad <\text{RAINFALL is E} 2.423> \quad \text{and} \quad <\text{TEMPERATURE is A} 12.592> \\
& \quad \text{and} \quad <\text{TEMPERATURE is B} 5.095> \quad \text{and} \quad <\text{DISTANCE is not C} 8.566> \\
& \quad \text{and} \quad <\text{DISTANCE is D} 5.019> \\
\text{then} & \quad <\text{SUITABILITY is A} 13.469> \quad \text{and} \quad <\text{SUITABILITY is not B} 19.502>
\end{align*}
\]

\[
\begin{align*}
\text{If} & \quad <\text{ALTITUDE is B} 3.393> \quad \text{and} \quad <\text{ALTITUDE is C} 4.722> \\
& \quad \text{and} \quad <\text{RAINFALL is D} 8.021> \quad \text{and} \quad <\text{RAINFALL is E} 3.045>
\end{align*}
\]
and <TEMPERATURE is D 20.815> and <DISTANCE is not C 12.624>
and <DISTANCE is D 2.345>
then <SUITSABILITY is not A 14.100> and <SUITSABILITY is B 16.672>
and <SUITSABILITY is E 8.627>

Then, overall degrees of matching of the left-hand side of the above two rules are calculated by the following summations:

\[-16.482 \times 0.3 + 2.186 \times 0 + 2.423 \times 0.1 + 12.592 \times 1.0 + 5.095 \times 0 - 8.566 \times 0 + 5.019 \times 0 = 7.8897\]
\[3.393 \times 0.7 + 4.722 \times 0 + 8.021 \times 0.9 + 3.045 \times 0.1 + 20.815 \times 0 - 12.624 \times 0 + 2.345 \times 1.0 = 12.2435\]

which are all positive, so both rules will fire. When this is done, an overall degree of <Suitability is A: very unsuitable> is calculated as 13.496 - 14.1 = -0.604, <Suitability is B: unsuitable> is calculated as -19.502 + 16.672 = -2.83, and <Suitability is E: very suitable> = 8.627. If the logistic function of equation (2.5) is applied at the action elements layer, then the membership function values for the above three fuzzy variables are as follows:

\[\mu_{A: \text{very unsuitable}} \text{ (Suitability)} = 0.353\]
\[\mu_{B: \text{unsuitable}} \text{ (Suitability)} = 0.057\]
\[\mu_{E: \text{very suitable}} \text{ (Suitability)} = 0.9998\]

If the maximum (also called winner-take-all) technique is employed as the defuzzification method, the final crisp solution 4 is obtained. That means, the selected block is an excellent site to build a golf course according to the fuzzy rules.

With the use of the ReFuNN approach, only those weights which have their values above a certain threshold are taken as rule elements. This conversion can cause loss of information, which may lead to less satisfactory performance.

### 2.6 Classifier Systems and Genetic Algorithms

As described earlier, a market-based rule learning (MBRL) system proposed in this thesis is a rule-learning system inspired by a classifier system. In this section, we first introduce what a
classifier system is. Then an overview of classifier systems is provided. Finally, current
difficulties existing in classifier systems are outlined.

Since 1978, two approaches have been developed in the field of classifier systems: the Michigan
Approach (Holland & Reitman 1978) and the Pitt Approach (DeJong 1988). This study employs
the Michigan approach classifier system because it is the classical approach, having proven itself
and undergone more development.

2.6.1 What is a Classifier System?
A classifier system is a machine learning system that learns syntactically simple string-encoded
rules (called classifiers) to guide its performance in an arbitrary environment (Holland & Reitman
1978). A classifier system is a type of reinforcement learning system (Minsky 1963; Michie
1974) that has three major components: a rule and message system, apportionment of credit
algorithm, and the genetic algorithm.

These three components are described in more detail as follows:

- **Rule and Message System**
  Each classifier consists of a rule or conditional statement whose constituents are words drawn
  from the ternary alphabet (0,1,#). It has one or more conditions as the *antecedent*, an action
  statement as the *consequent*, and an associated strength. The rule portion has the following
template:

  IF <condition1>&<condition2>&...<conditionN>
  Then <action>

  where,
  <condition> is encoded as a string from the alphabet {0, 1, #}
  <action> is encoded as a string from the alphabet {0, 1}

  The “#” symbol acts as a wild card or “don’t care” indicator in the condition, matching either
  a 0 or 1 -- the more “don’t care” symbols, the more general the rule. The measure used to
  quantify this is called *specificity*. The specificity of a classifier is the number of non # symbols
in the antecedent. If a classifier’s antecedent consists of all # characters then the specificity is zero, if there are no # characters in the antecedent then the specificity is equal to the antecedent’s string length.

The strength portion of the classifier gives a measure of the rule’s past performance in the environment in which it is learning. That is, the higher a classifier’s strength, then the better it has performed and the more likely it is to be used when the condition matches an environmental message and reproduce when the GA is applied. The strength values are relative; therefore, a range limit is set.

The messages, generated either from the environment or from the action of other classifiers, match the condition part of the classifier rule. Therefore, an action is a type of message, with the consequence of an action being the modification of the environment or attempted matching with other classifiers. This type of message posting system is also sometimes known as a Blackboard System (Nii 1986).

Those classifier systems in which classifiers only match messages from the environment and whose generated actions only modify the environment are called single-layer learning systems.

- **Apportionment of Credit Algorithm – the Bucket Brigade**

  The apportionment of credit algorithm deals with the modifications in strengths of classifiers as the classifier system learns. In a classic classifier system, the bucket brigade algorithm (Holland 1986) serves this purpose. In the bucket brigade, strength modifications occur via three interrelated mechanisms:

  - Auction
  - Reinforcement & punishment
  - Taxation

  In operation the classifier system receives messages from the environment or from the action of other classifiers. All the classifiers that match one (or more) of the messages compete, by
submitting a bid, in an auction to determine victorious classifiers that will either directly effect the environment or have the right to perform their actions that might be able to activate other classifiers. The victorious classifiers will be directly or indirectly ‘beneficial’ or ‘detrimental’ to the environment. With this feedback, the apportionment of credit algorithm appropriately uses reinforcement and punishment to increase or decrease the strengths of the victorious classifiers. Finally, taxation is levied on each classifier per iteration and on each classifier that submits a bid during an auction (so that inactive classifiers gradually get weeded out).

An auction is performed among all the classifiers which have an antecedent that matches at least one of the messages. With the matching classifier pool determined, the auction commences. Each classifier participating in the auction submits a bid, which is a function of the classifier’s strength and specificity. Only the bid of the victorious classifiers is paid, so only the victorious classifiers have their strengths decreased by the amount of their winning bids. The bid of classifier \( i \) at iteration \( t \), \( B_i(t) \), is calculated as:

\[
B_i(t) = C_{bid} \times \text{bid}_1 + \text{bid}_2 \times \text{specificity} \times S_i(t) \tag{2.18}
\]

where

- \( C_{bid} \) is a classifier bid coefficient that determines what proportion of a classifier’s strength will be bid and possibly lost on a single step.
- \( \text{bid}_1, \text{bid}_2 \) are bid coefficients associated with a classifier’s specificity.
- \( \text{specificity} \) is the number of non # symbols in the antecedent of classifier \( i \).
- \( S_i(t) \) is the strength of classifier \( i \) at step \( t \).

The strength \( S_i(t+1) \) of a classifier \( i \) at the end of iteration \( t \) is:

\[
S_i(t + 1) = S_i(t) - B_i(t) + T_i(t) + R_i(t) \tag{2.19}
\]

where

- \( S_i(t) \) is the strength of classifier \( i \) at beginning of iteration \( t \).
- \( B_i(t) \) is the classifier’s bid during iteration \( t \) (as defined by equation 2.18). only paid if victorious.
- \( R_i(t) \) is the reward given by the activated classifiers or reward from the environment.
\( T_i(t) \) represents the taxes paid by the classifier \( i \). The amount of taxes can simply be proportional to the classifier’s strength. Therefore, \( T_i(t) = C_{\text{tax}} \cdot S_i(t) \), where \( C_{\text{tax}} \) is a tax coefficient.

Classifier \( i \) only makes a bid payment if victorious in the auction. The reward factor, \( R_i(t) \), is only non-zero if the classifier has won the auction and was able to activate other classifiers or affect the environment on the previous iteration. The reward for the action at iteration \( t \) will not be applied until iteration \( t + 1 \).

Taxation occurs to prevent the classifier population from being cluttered with artificially high strength classifiers of little or no utility. There are two types of taxes: life tax and bid tax. The life tax is a fixed rate tax applied to every classifier on every iteration. This is done to reduce the strength of classifiers that rarely or never are matched and therefore provide little or no utility. These classifiers would otherwise survive and hinder the functioning of the genetic algorithm. The bid tax is a fixed rate tax that is applied to each classifier that bids during the iteration. One reason for a bid tax is to penalize overly general classifiers, i.e., classifiers that bid on every step but perhaps seldom win because they have a low specificity which leads to low bids and so a low chance of winning the auction.

As will be discussed later, new classifiers are inserted into the population at the average strength of their parent, thus the tax rate must be set to ensure that inactive classifiers are degraded sufficiently before the application of the genetic algorithm. If this is not done, relatively inactive classifiers can retain an unrealistically high level of strength and ultimately reach reproduction disproportionately, thereby cluttering future populations with large numbers of overrated inactive classifiers.

With all the apportionment of credit mechanisms defined, the complete strength equation is shown below:

\[
S_i(t + 1) = S_i(t) - C_{\text{life tax}} \cdot S_i(t) - B_i(t) - R_i(t) - C_{\text{bid tax}} \cdot S_i(t) \tag{2.20}
\]

Recall that,
\( B_i(t) \) is only paid if classifier \( i \) wins the auction.

\( R_i(t) \) will only be non-zero if classifier \( i \) won the auction and was able to activate other classifiers or effect the environment on iteration \( t-1 \).

\( C_{\text{bidtax}} S_i(t) \) is only paid if classifier \( i \) bids in the auction (irrespective of whether \( i \) wins the auction or not).

\( C_{\text{lifetax}} \) is the life tax coefficient

\( C_{\text{bidtax}} \) is the bid tax coefficient

### Genetic Algorithms

Genetic algorithms (GAs), which were introduced by Holland (1975), are used as a classifier discovery mechanism that generates new classifiers which ultimately may replace existing poorer performing classifiers if the newly discovered classifiers prove to be better ones.

Discussion of GAs is provided in the next section.

#### 2.6.2 Genetic Algorithms

A genetic algorithm (GA) is a stochastic search algorithm based on the mechanics of natural selection (Darwin 1897) and population genetics (Mettler et al. 1988). Genetic algorithms, as Goldberg (1989a) states and demonstrates, are theoretically and empirically proven to provide robust search in complex spaces.

How does a genetic algorithm work and what makes it different from other search methods? First of all a genetic algorithm does not work with a single object in the search space. It works with an entire population of different objects. The individuals in the population are assigned a fitness value. This value is an indication of how good a solution a certain individual is. Fitness is calculated by an objective evaluation function. This evaluation function is the only domain-specific part of the genetic search. The genetic search process itself is not guided by domain-specific knowledge, so it is almost universally applicable, especially in domains where very little is known about what is being searched for.

Using the population of the current generation and the fitness of its members, the next generation is created. This operates a fashion that is analogous to natural evolution: the best performing
individuals have the biggest chance to produce offspring. Offspring are like their parents, but not always precisely like them. Some of the newly made individuals eventually replace older less-fit ones. By using this approach, a large part of a search space can be randomly covered in a relatively short period of time.

Each individual in a GA population is encoded as a set of genes in chromosomes. In the simplest form of the GA, bit strings play the role of chromosomes, with individual bits playing the role of genes. For many applications, it is more natural to use an alphabet of many characters, integer or floating-point numbers, to encode chromosomes. Goldberg’s argument (Goldberg 1989b) suggests that a GA should exhibit poorer performance with multiple-character encoding than with binary encoding. However, this has been questioned by Antonisse (1989). Several empirical comparisons between binary encoding and multiple-character or real-valued encoding have shown better performance for the latter (Janikow & Michalewicz 1991, Wright 1991). But the performance depends very much on the problem and the details of the GA being used, and at present there are no rigorous guidelines for predicting which encoding will work best. In Holland’s classifier system, rules are represented by bit strings consisting of zeroes, ones and wildcards, so the GA operates on the bit strings.

There are three basic actions in the genetic algorithm: selection, creation of a new individual from parents and the replacement of older, poorly performing individuals.

Selection of individuals for procreation is the first step. A common selection method in GAs is fitness-proportionate selection, in which the number of times an individual is expected to reproduce is equal to its fitness divided by the average of fitnesses in the population (Mitchell 1996). Two commonly used methods of implementing fitness-proportionate selection are: roulette-wheel selection (Goldberg 1989a) and rank-based selection (Baker 1985). In roulette-wheel selection every individual is assigned a certain probability which is proportional to its fitness. The probabilities of all individuals should add up to one. Candidates for procreation are then selected according to these probabilities. It get its name, because it is somewhat like spinning a roulette wheel. In rank-based selection individuals are ordered according to their fitness. Individuals are then assigned fixed probabilities according to their rank, that is, their position in the list. Candidates are again selected with these probabilities. The difference between
rank-based selection and roulette-wheel selection is not very great. When using roulette-wheel selection, one should be careful that the ratio between the highest and lowest probability should not be too great. Otherwise certain individuals will very quickly start to dominate the population. Although rank-based selection has to sort all the individuals first, it is much easier to control the different probabilities of the individuals in the population. In addition, ranking avoids giving the far largest share of offspring to a small group of highly fit individuals, and thus reduces the selection pressure when the fitness variance is high. It also keeps up selection pressure when the fitness variance is low: the ratio of probabilities of individuals ranked $i$ and $i+1$ will be the same whether their absolute fitness differences are high or low. Tournament selection (Goldberg & Deb 1991) is another commonly used method that does not require the calculation of statistical probability over the entire population. Two individuals are chosen at random from the population. A random number $r$ is then chosen between 0 and 1. If $r < k$ (where $k$ is a parameter, for example 0.75), the fitter of the two individuals is selected to be a parent; otherwise the less fit individual is selected. The two are then returned to the original population and can be selected again. Technical comparisons of different selection methods are provided in Goldberg and Deb (1991), Back and Hoffmeister (1991), de la Maza and Tidor (1993), and Hancock (1994). However, as is the case for encodings, at present there are no rigorous guidelines on which methods should be used for which problems. In fact, they often have similar properties, and therefore the selection method is not critical to the success of the work. In Holland’s classic classifier system, the selection process is performed using roulette-wheel selection, where each classifier’s strength value is used as its fitness.

The next step that happens in the operation of the genetic algorithm is the creation of new individuals from the selected parents. This is usually done by three techniques: crossover, mutation and inversion. In the research presented in this thesis, inversion is not used. Crossover is the mixing of two parents. The chromosomes of the parents must be the same length. A random point (between two genes) in one of the parents is chosen. For the bit-string chromosome representation shown in Figure 2.12, the two children are created by joining the genes before the cut point of the one parent and the genes after the cut point of the other parent.
Mutation is a very simple operation. According to Goldberg (1989a), mutation is the occasional (with small probability) random alteration of a string position. Its purpose is to introduce new solutions into the population, which were not present at the outset or were lost during the search. For a bit-string chromosome representation, mutation simply means randomly changing a symbol value of 0, 1, and # to another value from the same set.

Crossover and mutation are only used with a certain probability, and are thus not used with every child: sometimes a child is just a copy of one of its parents.

The last step of the genetic algorithm is the replacement of old, poorly performing individuals. Supposing that we work with a fixed population size, it is necessary to replace old individuals in order to accommodate the new ones. The replacement method chosen in the classic classifier systems was called the crowding method by De Jong (1975). Crowding consists of finding a weak individual that is very similar to the newborn child as the candidate for replacement. In a classifier system, new classifiers are inserted into the population at the average strength of their parents. The purpose of the crowding method is for the genetic algorithm to optimize not only the fitness of the chromosomes but also their diversity. Goldberg and Richardson (1987) accomplished a similar result using an explicit “fitness sharing” function: each individual’s fitness was decreased by the presence of other population members, where the amount of decrease due to each other population member was an explicit increasing function of the similarity between the two individuals. Thus, individuals that were similar to many other individuals were punished, and individuals that were different were rewarded. Goldberg and Richardson showed that in some cases this could induce appropriate “speciation”, allowing the population members to converge towards several peaks in the fitness landscape, rather than all
converge to the same peak. There are other ways to promote diversity, and the reader can further investigate them in Deb and Goldberg (1989), Eshelman (1991), and Hillis (1992).

For the various parameters associated with a genetic algorithm, such as population size, crossover probability, mutation probability, there is a great deal of discussion in the literature concerning parameter settings and approaches to parameter adaptation. There are no conclusive results on what is best; most people use what has worked well in previously reported cases (Mitchell 1996). De Jong (1975) performed an early systematic study of how varying parameters affected the GA’s on-line search performance on a small suite of test functions. De Jong’s experiments indicated that the best population size was 50-100 individuals, the best crossover probability was around 0.6, and the best mutation rate was 0.001. These settings became widely used in the GA community (Mitchell 1996), even though it was not clear how well the GA would perform with these settings on problems outside De Jong’s test suite. Grefenstette (1986) conducted experiments that used the GA as an optimization procedure to optimize the parameters for another GA. In Grefenstette’s experiments, the “meta-level GA” evolved a population of 50 GA parameter sets for the problems in De Jong’s test suite. Each individual encoded six GA parameters, including population size, crossover probability, mutation probability, etc. Grefenstette’s experiments showed that the best population size was 30, the best crossover probability was 0.95, and the best mutation rate was 0.01. These parameters gave a small but significant improvement in on-line performance over De Jong’s settings. Schaffer, Caruana, Eshelman, and Das (1989) spent over a year systematically testing a wide range of parameter combinations. They found that the best settings for population size, crossover probability, and mutation probability were independent of the problem in their test suite. These settings were similar to those found by Grefenstette: population size 20–30, crossover probability 0.75-0.95, and mutation probability 0.005-0.01. Compared with other studies that have argued for larger population sizes (e.g., Goldberg 1989c), De Jong, Grefenstette, and Schaffer et al. demonstrated that a very small population size was better. The setting of population size in classifier systems has traditionally followed Goldberg’s principle “more is better”. For example, when Holme’s (1996) EpiCS was applied for investigation of epidemiologic surveillance data, the population size was fixed at 1000.
Although De Jong, Grefenstette, and Schaffer et al found that a particular setting of parameters worked best for on-line performance on their test suites, it seems unlikely that any general principles about parameter settings can be formulated \textit{a priori}, in view of the variety of problem types, encodings, and performance criteria that are possible in different applications. Moreover, the optimal population size, crossover probability, and mutation probability likely change over the course of a single run. Several researchers (Booker 1987; Davis 1989; Fogarty 1989; Davis 1991) have expressed the belief that the most promising approach is to have the parameter values adapt in real time to the ongoing search.

2.6.3 Overview of Classifier Systems

Classifier systems have been in existence for more than twenty years. While there was considerable research in the 1980s, the field began to wane as the decade closed. In the early to mid 1990s, classifier systems seemed too complicated to be studied, with few successful applications reported. But, during the last several years, new models have been developed and new applications have been presented which have caused a resurgence of this area.

The first classifier system was Cognitive System One (CS-1) by Holland and Reitman (1978). CS-1 attempted to traverse a simulated linear maze with external payoff only at the maze ends, so that the correct step-direction had to be learned at each interior point. The system’s success was the first example of the generation of rules appropriate to a task under the genetic algorithm, and the effective allocation of credit under conditions of infrequent payoff. Although CS-1 learned under infrequent payoff, it did not do so using the bucket brigade of today’s classifier systems. Instead CS-1 apportioned credit to activated classifiers using an \textit{epochal} algorithm. The epochal algorithm kept track of fairly extensive activation statistics and paid classifiers active since the last payoff event when a reward was next received.

Smith (1980) stripped the classifier system of its apportionment of credit system in his study of a system called LS-1 in Waterman’s (1970) poker playing task. By so doing, Smith sidestepped many of the knotty questions of credit assignment by requiring genetic evaluation of entire rule sets. This approach contrasts starkly with that of CS-1 where a single rule is taken as the corpuscle of genetic manipulation.
Goldberg (1983) applied classifier systems to two control problems: the centering of a Newtonian object in a one-dimensional space, and the generation of a rule set that would cover both normal and exceptional operating conditions on a gas pipeline. Although Goldberg’s work was the first application of a classifier system to a physical system, his work did not consider infrequent reward.

Wilson’s (1985) Animat system was the first to demonstrate the bucket brigade, though a simplified one, under infrequent payoff conditions. The simplification was to omit the posting and matching of messages, which means that classifiers only match messages from the environment and actions generated from classifiers only modify the environment. Wilson introduced two other procedures that aided rapid learning in this food-finding task. First, when the system could not match an environmental input, a matching classifier was simply created using an action chosen randomly or by a form of look-ahead. Second, each classifier stored an estimate of the average number of steps between its activation and the finding of food. The estimate was combined with strength to form the classifier’s bid. This encouraged the formation of paths that were both remunerative and short. Later, Wilson developed other important models such as BOOLE (Wilson 1987), NEWBOOLE (Bonelli et al. 1990), and ZCS (Wilson 1994) which finally lead to XCS (Wilson 1995). XCS keeps all the main ideas of Holland’s model while it introduces some fundamental changes. First, in XCS the evaluation of classifiers is based on the accuracy of a classifier’s prediction but not on the classifier’s strength as in Holland’s model. Thus, in XCS, a classifier is considered “useful” if it conveys accurate information about the problem. Second, in XCS a niche genetic algorithm is used. Thus, the discovery component GA does not act on the whole population (as usually happened in Holland’s model) but on subsets of classifiers on each invocation. Since XCS is a single-layer classifier system without involving complex internal operations such as message posting and matching, it is more easily studied and analyzed compared with the classic classifier system.

Robertson and Riolo (1988) investigated letter sequence prediction and produced the first evidence of the generation, and the use by the bucket brigade, of internal messages. The prediction task required internal messages since performance depended partially on memory of letters seen earlier.
Grefenette (1988) made the first experimental comparison between the bucket brigade and a simplified version of Holland and Reitman’s epochal credit allocation scheme. In the latter, termed the profit sharing plan (PSP), Grefenette argued that classifier strengths under PSP more accurately predict final rewards than those under the bucket brigade, and his experimental results with a two-dimensional state space showed superior performance for PSP.

Holmes’s (1996) EpiCS was developed from NEWBOOLE to meet the demands of epidemiologic data. EpiCS’s distinctive features include: (i) techniques for controlling over- and under-generalization of data; (ii) the use of differential negative reinforcement of false positive and false negative errors in classification, and (iii) a methodology for determining risk as a measure of classification. All of these features have led to the successful use of EpiCS in knowledge discovery applications to actual clinical databases of various sizes and levels of complexity.

Stolzmann’s (2000) anticipatory classifier system (ACS) differs greatly from other classifier system models in that ACS learns not only how to perform a certain task, but also build an internal model of the dynamics of the task. In ACS, classifiers are not simple condition-action rules but are extended by an effect part. The effect part of a classifier is used to anticipate the environmental state which results from the execution of the classifier action. Besides the genetic algorithm, an Anticipatory Learning Process (ALP) is used for rule discovery which directly learns from changes in the environment. ACS forms explicit condition-action-effect classifiers with a generalization capability in the classifier conditions. This leads to an internal model of the environment.

From the system applications point of view, most of the reported results fall into three main areas:

- **Using Classifier Systems to Control Simulated Creatures.**

  This is the research area where classifier systems have been applied most. They have been reported in (Booker 1982; Wilson 1985; Wilson 1987; Dorigo & Sirtori 1991; Roberts 1993; Dorigo & Colombetti 1998).
Using Classifier Systems in Computational Economics.
Classifier systems have been used to model adaptive agents in artificial stock markets (Arthur et al. 1996; Mitlohnner 1996; Lebaron et al. 2000). Bull (1999) applied a classifier system to a simulated Continuous Double-Auction (CDA) market. Schulenburg and Ross (1999) employed classifier systems to model the behaviour of agents trading risk free bonds and risky assets in a stock market environment.

Using Classifier Systems for Knowledge Discovery.
In this area, classifier systems are used to extract information about interesting phenomena described by multidimensional data. Holmes (1996) applied his EpiCS to complete classification tasks from different medical domains. de Boer (1994) applied a classifier system to the Mushrooms Database (UCI 1998), showing the classifier system can have a good generalization ability. Saxon and Barry (2000) applied XCS to the Monks Problem (UCI 1998), demonstrating that XCS is able to produce a classification performance which exceeds the performance of most current machine learning techniques, such as neural networks and decision trees. Wilson (2000) applied XCS to the Wisconsin Breast Cancer Database (UCI 1998) and found that it performed as well as state-of-the-art machine learning algorithms. According to Lanzi et al. (2000), the recent results and the amount of ongoing work in utilizing classifier systems for knowledge discovery suggest that knowledge discovery applications may represent an important breakthrough for classifier systems, taking them well beyond the Evolutionary Computation community.

2.6.4 Limitations of Classifier Systems
Although quite a few researchers have invested effort in developing models and methods in the field of classifier systems, the current models and methods have shared some major limitations: interpretation difficulty, initial classifier chain generation difficulty, and initial system parameter setting difficulty.

A major drawback with classifier system concerns the interpretation difficulty. The string rule representation and the large number of rules necessary to solve any realistic problem in a classifier system impedes a straightforward and comprehensive interpretation. de Boer (1994) applied a classifier system to the Mushrooms Database (UCI 1998), showing that 200 classifiers
were necessary in order to obtain 90% predictive accuracy on test sets. No interpretation of the learned classifier system was undertaken, as the number of relevant classifiers was too large. Holmes (1996) applied EpiCS in diagnostic classification data sets. Although the experimental results demonstrated that EpiCS with 1000 classifiers achieved better classification accuracy than a traditional multivariate statistical analysis, the question “Can a EpiCS discover features in clinical data that point to possible associations with a specific outcome?” still remained. Wilson (2000) applied his XCS to the Wisconsin Breast Cancer Database (UCI 1998) showing that XCS performed as well as state-of-the art machine learning algorithms with the size of classifier population at 6400. Only a small fraction of the evolved population (the 27 most fit classifiers) were directly interpretable. Therefore, Wilson argued that “It would be desirable to find algorithms of extracting all the classifiers’ implications as rules of thumb and in other representations”.

The bucket brigade algorithm, as defined by Holland (1986), was developed to solve the credit assignment problem in classifier systems, and has experienced limited success to date. Credit assignment (Minsky 1963) deals with the problem of deciding, when many parts of a system are active over a period of time (or even at every time step), which of those parts that are active at some step $t$ should contribute to achieving some desired outcome at step $t+n$, for $n>0$. The bucket brigade has the virtue of distributing credit to large numbers of sequentially acting classifiers by means of strictly local transactions among them. Unfortunately, as Wilson and Goldberg (1989) have suggested, the technique has a primary weakness: initial bucket brigade chains are hard to generate. In the decades since the classifier system was invented, there have been few classifier systems in which chains of sequentially activated classifiers have actually been formed (Wilson 1985; Robertson & Riolo 1988; Grefenstette 1988). Of these three classifier systems, only Robertson and Riolo’s used internal messages. The three systems learned under conditions of infrequent payoff in which the generation of stage-setting classifiers was required. The discovery of such classifiers remains a difficult problem.

The determination of the best system parameters is an area that classifier system research has not addressed completely (Richards 1995). de Boer’s experiments (de Boer 1994) showed that these initial parameter settings can greatly influence the learning capabilities of the classifier system. Binary string representations have made the system initialisation more difficult. When learning
is occurring, some form of an initial population must be created. In such a case, if a classifier system starts with a randomly generated initial population, then many population parameters and message parameters must be set. These include the word length of the messages, the number of conditions in the antecedent, the word length for each condition and action, and the probability of selecting a wildcard (#) in the randomly generated population.

In summary, although much effort has been devoted to the development of working classifier systems over the past years, the current classifier systems have major weaknesses. In Chapter 6, a novel market-based rule learning (MBRL) system is proposed that is aimed at addressing these limitations.

2.7 Summary
This chapter has provided background material for the remainder of the thesis. The statistical methods used in the experiments of the thesis have been described. A brief introduction of feed-forward neural networks, fuzzy systems, and fuzzy neural networks have been given, and the current difficulty of constructing membership functions in fuzzy systems has been addressed. An overview of feature selection and rule extraction methods has been described, followed by demonstrations of some of the existing feature selection and rule extraction methods. In terms of feature selection via neural networks, the difficulty of finding a proper pruning threshold in pruning algorithms has been discussed. Finally, an introduction to classifier systems and an in-depth review of related work in classifier systems, including genetic algorithms have been provided. The major limitations of current classifier systems have been outlined.

The next chapter will provide a description of the problem domains used in the experiments of this research.
Chapter 3 The Problem Domains

3.1 Introduction

There are six data sets used in the experiments of this research. In this chapter, the descriptions of these six data sets are provided. Section 3.2 – 3.7 give the detailed descriptions and background information of the Iris Classification Data (Fisher 1936), the Pima Indians Diabetes Data (UCI 1998), the Wine Recognition Data (UCI 1998), the Golf Course Problem (Purvis et al. 1996), the Cook Islands Sea Cucumber Habitat Data (Drumm et al. 1999), and the New Zealand Asthma Incidence Data (Hales et al. 1998a) respectively. Section 3.8 describes how data sets were prepared for various techniques such as neural network training, rule extraction, and rule refinement.

All six selected problem domains are supervised learning tasks with output class information provided in the data sets. The Iris Classification Data, the Pima Indians Diabetes Data, and the Wine Recognition Data were selected because they have been widely analysed in the literature (too much to list here), so that comparisons can be easily made among different techniques. These three data sets can be obtained via ftp from the University of California - Irvine machine learning data repository (UCI 1998). The Golf Course Problem is a spatial problem. The experimental results conducted based on the Golf Course Problem can provide evidence of how the techniques under investigation perform for spatial data sets. The Cook Islands Sea Cucumber Habitat Data, and the New Zealand Asthma Incidence Data represent practical applications from the areas of ecology and medicine. In such practical applications, the data sets normally contain a significant level of noise, and are used to test if the relevant techniques can handle the data with noise. In addition, the experimental results based these on two latter data sets can not only provide evidence for the performance of relevant techniques, but also indicate possible useful solutions for human decision-makers.

3.2 Iris Classification Data

In the Iris data set, there are three classes of Iris flowers to be discriminated using four real-valued features that represent physical characteristics of the flowers. The Iris data set contains 150 samples, with 50 samples for each of the classes: *Iris setosa*, *Iris versicolor*, and *Iris*
virginica. Each sample is described using four numeric attributes: Sepal-length, Sepal-width, Petal-length, and Petal-width as shown in Table 3.1.

For the experiments presented in the thesis, all 150 samples were used. The class information is shown in Table 3.2.

**Table 3.1. Attribute descriptions for the Iris data Set**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Variable Type</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_1$</td>
<td>Sepal-length</td>
<td>Real</td>
<td>4.3 - 7.9</td>
</tr>
<tr>
<td>$I_2$</td>
<td>Sepal-width</td>
<td>Real</td>
<td>2.0 - 4.4</td>
</tr>
<tr>
<td>$I_3$</td>
<td>Petal-length</td>
<td>Real</td>
<td>1.0 - 6.9</td>
</tr>
<tr>
<td>$I_4$</td>
<td>Petal-width</td>
<td>Real</td>
<td>0.1 - 2.5</td>
</tr>
</tbody>
</table>

**Table 3.2. Class information for the Iris data set**

<table>
<thead>
<tr>
<th>Class</th>
<th>Number of samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iris Setosa</td>
<td>50</td>
</tr>
<tr>
<td>Iris Versicolor</td>
<td>50</td>
</tr>
<tr>
<td>Iris Virginica</td>
<td>50</td>
</tr>
</tbody>
</table>

### 3.3 Pima Indians Diabetes Data

The Pima Indians Diabetes database consists of 768 samples taken from patients who may show signs of diabetes. Each sample is described using 8 continuous-valued attributes shown in Table 3.3. The class index represents either a positive or negative test for diabetes.

For the experiments presented in later chapters, 768 samples – 268 samples “tested positive for diabetes” and the remaining 500 samples “tested negative for diabetes” – were used. The class information is shown in Table 3.4.
Table 3.3. Attribute descriptions for the Pima Indians Diabetes data set

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Variable Type</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_1$</td>
<td>Number of times pregnant</td>
<td>Integer</td>
<td>0 - 17</td>
</tr>
<tr>
<td>$I_2$</td>
<td>Plasma glucose concentration in a 2 hour oral glucose tolerance test</td>
<td>Real</td>
<td>0 - 199</td>
</tr>
<tr>
<td>$I_3$</td>
<td>Diastolic blood pressure</td>
<td>Real</td>
<td>0 - 122</td>
</tr>
<tr>
<td>$I_4$</td>
<td>Triceps skin fold thickness</td>
<td>Real</td>
<td>0 - 99</td>
</tr>
<tr>
<td>$I_5$</td>
<td>2 hour serum insulin</td>
<td>Real</td>
<td>0 - 846</td>
</tr>
<tr>
<td>$I_6$</td>
<td>Body mass index</td>
<td>Real</td>
<td>0 - 67.1</td>
</tr>
<tr>
<td>$I_7$</td>
<td>Diabetes pedigree function</td>
<td>Real</td>
<td>0.078 - 2.42</td>
</tr>
<tr>
<td>$I_8$</td>
<td>Age</td>
<td>Integer</td>
<td>21 - 81</td>
</tr>
</tbody>
</table>

Table 3.4. Class information for the Pima Indians Diabetes data set

<table>
<thead>
<tr>
<th>Class</th>
<th>Number of samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tested positive for diabetes</td>
<td>268</td>
</tr>
<tr>
<td>Tested negative for diabetes</td>
<td>500</td>
</tr>
</tbody>
</table>

3.4 Wine Recognition Data

The Wine Recognition Data consists of 178 samples, each of which is represented by 13 chemical, attributes shown in Table 3.5. All 13 attributes are real-valued. The 178 samples are classified into one of the three types of wines, which are represented as class 0, class 1 and class 2 in Table 3.6. We used all 178 samples for experiments.
Table 3.5. Attribute description for the Wine Recognition data set

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Variable Type</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_1$</td>
<td>Alcohol</td>
<td>Real</td>
<td>11.03 - 14.83</td>
</tr>
<tr>
<td>$I_2$</td>
<td>Malic acid</td>
<td>Real</td>
<td>0.74 - 5.8</td>
</tr>
<tr>
<td>$I_3$</td>
<td>Ash</td>
<td>Real</td>
<td>1.36 - 3.23</td>
</tr>
<tr>
<td>$I_4$</td>
<td>Alkalinity of ash</td>
<td>Real</td>
<td>10.6 - 30</td>
</tr>
<tr>
<td>$I_5$</td>
<td>Magnesium</td>
<td>Integer</td>
<td>70 - 162</td>
</tr>
<tr>
<td>$I_6$</td>
<td>Total phenols</td>
<td>Real</td>
<td>0.98 - 3.88</td>
</tr>
<tr>
<td>$I_7$</td>
<td>Flavanoids</td>
<td>Real</td>
<td>0.34 - 5.06</td>
</tr>
<tr>
<td>$I_8$</td>
<td>Nonflavanoid phenols</td>
<td>Real</td>
<td>0.13 - 0.66</td>
</tr>
<tr>
<td>$I_9$</td>
<td>Proanthocyanins</td>
<td>Real</td>
<td>0.41 - 3.56</td>
</tr>
<tr>
<td>$I_{10}$</td>
<td>Color intensity</td>
<td>Real</td>
<td>1.28 - 13</td>
</tr>
<tr>
<td>$I_{11}$</td>
<td>Hue</td>
<td>Real</td>
<td>0.48 - 1.71</td>
</tr>
<tr>
<td>$I_{12}$</td>
<td>OD280/OD315 of diluted wines</td>
<td>Real</td>
<td>1.27 - 4</td>
</tr>
<tr>
<td>$I_{13}$</td>
<td>Proline</td>
<td>Integer</td>
<td>278 - 1680</td>
</tr>
</tbody>
</table>

Table 3.6. Class information for the Wine Recognition data set

<table>
<thead>
<tr>
<th>Class</th>
<th>Number of samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class 0</td>
<td>59</td>
</tr>
<tr>
<td>Class 1</td>
<td>71</td>
</tr>
<tr>
<td>Class 2</td>
<td>48</td>
</tr>
</tbody>
</table>

3.5 Golf Course Problem

The Golf Course Problem is based on the problem of determining suitable sites for public golf courses in the South Island of New Zealand. For this problem it was assumed that a suitable location could be determined from the observed data of mean altitude, mean annual rainfall, mean summer temperature, and the distance to the nearest of four principle urban centres of the South Island as shown in Table 3.7.
Table 3.7. Attribute descriptions for the Golf Course Problem

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Variable Type</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>( I_1 )</td>
<td>mean altitude</td>
<td>Real</td>
<td>1 - 3203</td>
</tr>
<tr>
<td>( I_2 )</td>
<td>mean annual rainfall</td>
<td>Real</td>
<td>500 - 3000</td>
</tr>
<tr>
<td>( I_3 )</td>
<td>mean summer temperature</td>
<td>Real</td>
<td>10.5 - 16.1</td>
</tr>
<tr>
<td>( I_4 )</td>
<td>distance to the nearest of four</td>
<td>Real</td>
<td>0 - 261</td>
</tr>
<tr>
<td></td>
<td>principle urban centers</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Each of the 153,036 1 km² blocks of the South Island was taken to be a candidate location for a golf course, and for each block, values for the four attributes were determined and placed in a data set. In order to provide an evaluation mechanism, an artificially “correct” classification was determined based on a set of plausible rules. A description of this rule set can be found in Purvis et al. (1996). The output class of golf course suitability was taken to have one of five possible values, ranging from 0 (very unsuitable) to 4 (very suitable), and a map of the “correct” solution set is shown in Figure 3.1. Note that because of the rigid boundaries of the rules, the solution set is only piece-wise differentiable, and therefore neural network analysis, which is based on continuous differentiable functions, may show inaccuracies near these boundaries.

Figure 3.1. Solution set for golf course suitability (produced by human expert)
We randomly selected 1525 samples (~1% of the total) for experiments. The class information is shown in Table 3.8.

**Table 3.8.** Class information for the Golf Course Problem

<table>
<thead>
<tr>
<th>Class</th>
<th>Number of samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class 0 (very unsuitable)</td>
<td>152</td>
</tr>
<tr>
<td>Class 1 (unsuitable)</td>
<td>357</td>
</tr>
<tr>
<td>Class 2 (neutral)</td>
<td>404</td>
</tr>
<tr>
<td>Class 3 (suitable)</td>
<td>429</td>
</tr>
<tr>
<td>Class 4 (very suitable)</td>
<td>183</td>
</tr>
</tbody>
</table>

**3.6 Cook Islands Sea Cucumber Habitat Data**

This data set was used for predicting the habitat preferences of one species of tropical sea cucumber, *Holothuria leucospilota*, in the reef-top ecosystem of Rarotonga, Cook Islands (Drumm et al. 1999). A total of 128 sites were sampled for environmental and biological variables, using 2m × 50m (100m²) strip transects. This size sample unit was selected to account for the patchy distribution of the animals, and the number of *H. leucospilota* encountered along each transect was recorded. In addition to the species’ abundance, ten environmental variables that were expected to have an influence on the habitat preference of the sea cucumber were recorded at each of the 128 locations (Table 3.9). These included the exposure of the site (windward or leeward side of the island), and the following microhabitat variables: %sand, %rubble, %consolidated rubble, %boulder, %rock/pavement, %live coral, %dead coral, %mud/silt, and %gravel. These microhabitat variables were estimated as a percentage of the total 100m² area sampled.
Table 3.9. The related attributes of the habitat preference of the Sea Cucumber

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Variable Type</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_1$</td>
<td>Exposure (Windward or Leeward)</td>
<td>Discrete(nominal)</td>
<td>0, 1</td>
</tr>
<tr>
<td>$I_2$</td>
<td>%sand</td>
<td>Real</td>
<td>0 - 100%</td>
</tr>
<tr>
<td>$I_3$</td>
<td>%rubble</td>
<td>Real</td>
<td>0 - 100%</td>
</tr>
<tr>
<td>$I_4$</td>
<td>%consolidated rubble</td>
<td>Real</td>
<td>0 - 100%</td>
</tr>
<tr>
<td>$I_5$</td>
<td>%boulder</td>
<td>Real</td>
<td>0 - 100%</td>
</tr>
<tr>
<td>$I_6$</td>
<td>%rock/pavement</td>
<td>Real</td>
<td>0 - 100%</td>
</tr>
<tr>
<td>$I_7$</td>
<td>%live coral</td>
<td>Real</td>
<td>0 - 100%</td>
</tr>
<tr>
<td>$I_8$</td>
<td>%dead coral</td>
<td>Real</td>
<td>0 - 100%</td>
</tr>
<tr>
<td>$I_9$</td>
<td>%mud/silt</td>
<td>Real</td>
<td>0 - 100%</td>
</tr>
<tr>
<td>$I_{10}$</td>
<td>%gravel</td>
<td>Real</td>
<td>0 - 100%</td>
</tr>
</tbody>
</table>

All observed sites were divided into two distinct classes according to the frequency of animals at each location within the 100m² sample unit. The habitat condition class considered to be “average” was characterized by having $< 1$ animal/m², and the habitat condition class considered to be “good” was attributed to sites with $> 1$ animal/m². ‘Good’ condition sites were given a target value of 1, while ‘average’ ones had a target value of 0. The details are shown in Table 3.10.

Table 3.10. Site condition class ranges for Cook Islands Sea cucumber Habitat data

<table>
<thead>
<tr>
<th>Class Unnamed</th>
<th>Number of sites</th>
<th>Animal frequency range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class 0 (average condition class)</td>
<td>104</td>
<td>0 - 93</td>
</tr>
<tr>
<td>Class 1 (good condition class)</td>
<td>24</td>
<td>114 - 918</td>
</tr>
</tbody>
</table>

3.7 New Zealand Asthma Incidence Data

New Zealand Asthma Incidence Data was based on an investigation of the patterns of self-reported asthma symptoms in relation to demographic and environmental factors in New Zealand (Hales et al. 1998a). The data consists of 25,000 adults aged 20 – 44 who responded to a postal questionnaire. Each record was classified to be either “asthma positive”, or “asthma negative”. “asthma positive” was defined according to the proportion of subjects who reported one or more of the following symptoms: woken with shortness of breath in the past
In this research, we focus on the 5103 records associated with people aged 20 to 25. From these 5103 records, a set of 1000 records was randomly selected for experiments: 500 records reporting “asthma positive” and 500 records reporting “asthma negative”. For each record, 9 environmental and demographic attributes are given as shown in Table 3.11; 4 attributes (ethnicity, land use, social deprivation index, and sex) have discrete values, and the rest have continuous values. The class information is shown in Table 2.12.

**Table 3.11.** The related attributes of the New Zealand Asthma Incidence data set

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Variable Type</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_1$</td>
<td>Ethnicity</td>
<td>Discrete (nominal)</td>
<td>(0: European; 1: Maori; 2: Pacific Islander; 3: Chinese; 4: Others)</td>
</tr>
<tr>
<td>$I_2$</td>
<td>Humidity</td>
<td>Real</td>
<td>65.4% - 98.1%</td>
</tr>
<tr>
<td>$I_3$</td>
<td>Land use</td>
<td>Discrete(nominal)</td>
<td>52 classes</td>
</tr>
<tr>
<td>$I_4$</td>
<td>Mean temperature</td>
<td>Real</td>
<td>5 - 16 degree</td>
</tr>
<tr>
<td>$I_5$</td>
<td>Social deprivation index</td>
<td>Discrete(ordinal)</td>
<td>1 - 10</td>
</tr>
<tr>
<td>$I_6$</td>
<td>Rainfall</td>
<td>Real</td>
<td>30.9mm/month - 678.8 mm/month</td>
</tr>
<tr>
<td>$I_7$</td>
<td>Sex</td>
<td>Discrete(nominal)</td>
<td>(0: Male; 1: Female)</td>
</tr>
<tr>
<td>$I_8$</td>
<td>Sunshine</td>
<td>Real</td>
<td>12 hours/day - 15.4 hours/day</td>
</tr>
<tr>
<td>$I_9$</td>
<td>Wind speed</td>
<td>Real</td>
<td>5.5 m/s - 24.9 m/s</td>
</tr>
</tbody>
</table>

**Table 3.12.** Class information for the New Zealand Asthma Incidence data set

<table>
<thead>
<tr>
<th>Class</th>
<th>Number of samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asthma positive</td>
<td>500</td>
</tr>
<tr>
<td>Asthma negative</td>
<td>500</td>
</tr>
</tbody>
</table>
3.8 Data Preparation

As described earlier in Chapter 2, predictive accuracy, comprehensibility, and fidelity in the experiments of the thesis are evaluated by cross-validation, and experimental results are given on the mean calculated based on the 10-fold cross validation for all data sets. Therefore, for each selected data sets, ten cross-validation trials were created. For each cross-validation trial the relative proportions of the training set and testing set were 9/10 and 1/10, respectively. For example, for the Iris data set, each cross-validation trial consists of two subsets: 135 samples for training, and 15 samples for testing. It is important to acknowledge that, for the Golf Course Problem, a cross validation procedure may not produce truly independent sets of training and test samples since the 1525 total samples were randomly selected without considering the spatial correlation of the sample blocks. As the affect of this correlation, the prediction of the various learning techniques on the test sets might be overestimated.

As also described in Chapter 2, the training phase of neural network models is stopped when the measured error on a training validation has passed through a minimum and has begun to increase. Therefore, a training validation set was needed for this purpose. A training validation set can be just a subset of the training set. Recall that for each cross-validation trial, the relative proportions of the training set and testing set were 9/10 and 1/10, respectively. For those cases when a training validation set was used, each cross-validation trial employed a partitioning of the data according to the following relative proportions: 6/10 for the training set, 3/10 for the training validation set, and 1/10 for the test set. Note that training validation sets were only used for monitoring neural network and fuzzy neural network training in experiments conducted in Chapter 4 and Chapter 5. For other experiments associated with rule extraction and rule refinement, training validation sets were not employed.

Various techniques for feature selection, rule extraction, and rule refinement were employed for experimenting in this research. The data preparation for these different techniques are described as follows:


- **Data Preparation for the Techniques Associated with Feed-forward Neural Network Training.**

The techniques associated with feed-forward neural network training in this research include: three network pruning mechanisms proposed in Chapter 4, the NeuroLinear rule extraction approach discussed in Chapter 2 and experimented with in Chapter 8, and the MBRL system proposed in Chapter 6 and experimented with as a post-processing tool to refine NeuroLinear-generated rules in Chapter 7 and Chapter 8.

Before the feed-forward network’s learning process started, the training and test data were prepared as follows:

- All discrete input attributes and output classes were coded as binary representations. For example, the *sex* input attribute in the New Zealand Asthma Incidence data set was coded as male \{1, 0\} and female \{0, 1\}; and the three output attributes 0, 1, and 2 for the Wine Recognition data set were coded as \{1, 0, 0\}, \{0, 1, 0\}, and \{0, 0, 1\}.

Among six selected data sets, two data sets (Cook Islands Sea Cucumber Habitat Data and New Zealand Asthma Incidence Data) involve discrete input attributes. Their input attributes for neural network training are presented in Table 3.13 and Table 3.14, respectively.
### Table 3.13. Input Attributes of the Cook Islands Sea Cucumber Habitat data for neural network training

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_1 - X_2$</td>
<td>Exposure (Windward or Leeward)</td>
</tr>
<tr>
<td>$X_3$</td>
<td>%sand</td>
</tr>
<tr>
<td>$X_4$</td>
<td>%rubble</td>
</tr>
<tr>
<td>$X_5$</td>
<td>%consolidated rubble</td>
</tr>
<tr>
<td>$X_6$</td>
<td>%boulder</td>
</tr>
<tr>
<td>$X_7$</td>
<td>%rock/pavement</td>
</tr>
<tr>
<td>$X_8$</td>
<td>%live coral</td>
</tr>
<tr>
<td>$X_9$</td>
<td>%dead coral</td>
</tr>
<tr>
<td>$X_{10}$</td>
<td>%mud/silt</td>
</tr>
<tr>
<td>$X_{11}$</td>
<td>%gravel</td>
</tr>
</tbody>
</table>

### Table 3.14. Input attributes of the New Zealand Asthma Incidence data for neural network training

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_1 - X_5$</td>
<td>Ethnicity</td>
</tr>
<tr>
<td>$X_6$</td>
<td>Humidity</td>
</tr>
<tr>
<td>$X_7 - X_{38}$</td>
<td>Land use</td>
</tr>
<tr>
<td>$X_9$</td>
<td>Mean temperature</td>
</tr>
<tr>
<td>$X_{39} - X_{58}$</td>
<td>Social deprivation index</td>
</tr>
<tr>
<td>$X_{59}$</td>
<td>Rainfall</td>
</tr>
<tr>
<td>$X_{71} - X_{72}$</td>
<td>Sex</td>
</tr>
<tr>
<td>$X_{73}$</td>
<td>Sunshine</td>
</tr>
<tr>
<td>$X_{74}$</td>
<td>Wind speed</td>
</tr>
</tbody>
</table>

- All numeric input attributes were linearly normalized between 0 to 1. If the minimum and maximum values associated with input attribute $D$ are $D_{min}$ and $D_{max}$, the formula for transforming each data value $d$ to normalized $d'$ is:
\[ d' = \frac{d - D_{\min}}{D_{\max} - D_{\min}} \]  

(3.1)

The normalization was performed after the training and testing sets were split to ensure some independence between training and test sets. Therefore, \( D_{\min} \) and \( D_{\max} \) was taken independently between the training and test sets.

For the experiments associated with the MBRL system to refine NeuroLinear-generated rules, the training and test data sets used had normalized inputs and discrete outputs.

- **Data Preparation for the Techniques Associated with Fuzzy Neural Network FuNN.**

The techniques associated with the fuzzy neural network FuNN in this research includes: the Chi2-based membership selection approach presented in Chapter 4, the ReFuNN fuzzy rule extraction approach discussed in Chapter 2 and used empirically in Chapter 8, and the MBRL system presented in Chapter 6 and employed experimentally as a post-processing tool to refine ReFuNN-generated rules in Chapter 7 and Chapter 8.

Before the fuzzy neural network FuNN started training, the training and test data were prepared as follows:

- The output attribute was fuzzified, based on the membership functions generated by the fixed centre-based membership function selection approach.
- All input attributes were fuzzified, based on the membership functions generated by either the Chi2-based membership function selection approach or the fixed centre-based membership function selection approach.

For the experiments associated with the MBRL system to refine ReFuNN-generated rules, the training and test data set formats were of fuzzified inputs and discrete outputs. The fuzzied inputs were real-valued in the experiments.
Data Preparation for the Chi2 Algorithm, C4.5 Decision Trees, and the Rule Generator X2R.

For the experiments associated with the Chi2 algorithm and the C4.5 decision tree software, training and test data sets with original real-valued inputs and discrete outputs were used. For the experiments in connection with the rule generator X2R, those input data sets having real-valued attributes were first discretized so that the discrete-valued inputs could then be used in connection with the Chi2 algorithm.

3.9 Summary

This chapter has been devoted to describing six selected data sets that have been used in the experiments of the thesis. The discussion addressed issues such as why each data set was selected and how data sets were prepared for the various techniques.

The next chapter will focus on feature selection techniques, which are critical to the implementation of data pre-processing for rule extraction.
Part II
Feature Selection

Part II is devoted to feature selection issues and concerns both statistical and connectionist methods. Chapters 4 and 5 belong to this part. In Chapter 4, how to achieve feature selection via neural networks is first presented. Three neural network pruning schemes are then proposed. A detailed analysis and comparison of experimental results from six data sets are given. Chapter 5 presents Chi2-based spatial data filtering and a Chi2-based membership functions selection method for fuzzy systems. Both of these applications demonstrate how techniques from separate approaches can be combined to yield improved results.
Chapter 4
Feature Selection and Neural Network Analysis

4.1 Introduction
In this chapter we first discuss how feature selection can be achieved by means of neural networks. Three neural network pruning schemes are then presented. In order to examine the performance of the neural network feature selection schemes and compare them with a \( \chi^2 \) statistic-based feature selection algorithm, Chi2, six data sets are used for experiments. The analysis of these experimental results, particularly in connection with issues concerning the predictive accuracy, the complexity of representations, feature reduction, computing time, and the generality of selected features, are also presented in this chapter.

4.2 Feature Selection via Neural Networks
In this section, we examine how feed-forward neural networks select input attributes that are useful for discriminating output classes in a given set of input patterns. The use of neural networks as a feature selector was reported by Setiono and Liu (1997a). They used a simple criterion to remove an attribute based on the accuracy rate of the network, after which the network is retrained, and the selection process is repeated until no attribute meets the criterion for removal. The approach proposed in this section is more straightforward. First a three-layer feed-forward neural network is trained and then pruned while maintaining its performance. At the end, only those input attributes which have connections to hidden nodes are chosen as selected features. The network-pruning scheme is the key to the proposed algorithm.

4.2.1 Neural Network Training
The basic structure of the neural network in this work is a standard three-layer feed-forward neural network, which consists of an input layer, a hidden layer and an output layer. The number of input nodes corresponds to the dimensionality of the examples of a learning problem, and the number of output nodes is determined by the number of output classes. The number of hidden nodes depends on the problem in hand. The hyperbolic tangent function \( \delta(x) \) (equation 2.6 in Chapter 2) and the logistic function \( \sigma(x) \) (equation 2.5 in Chapter 2) are applied in the hidden
layer and the output layer, respectively, as the activation functions. The typical error function associated with the neural network is the mean squared error (4.1).

\[ E(w,v) = \frac{1}{2} \sum_{i=1}^{k} \sum_{p=1}^{c} (S_{pi} - t_{pi})^2 \]  

(4.1)

where \( k \) is the number of training examples, \( c \) is the number of output nodes, \( S_{pi} \) is the output of the network at output node \( p \) for training sample \( X_i \), and \( t_{pi} \) is the target value at the output node \( p \) for training sample \( X_i \). Given an \( n \)-dimensional input example \( X_i, i \in \{1,2,\ldots,k\} \), \( S_{pi} \) is given according to the equation

\[ S_{pi} = \sigma(\sum_{m=1}^{h} \delta(\sum_{l=1}^{n} X_i^l w_{ml}) v_{pm}) \]  

(4.2)

where \( n \) is the number of input nodes, \( h \) is the number of hidden nodes, \( w_{ml} \) is the weight of the connection from the \( l \)-th input node to the \( m \)-th hidden node, and \( v_{pm} \) is the weight of the connection from the \( m \)-th hidden node to the \( p \)-th output node.

When the task of pruning (i.e. reducing the number of connections in) a neural network is undertaken, it is a common practice to add a penalty term to the error function during training (Ji et al. 1990; Weigend et al. 1991; Reed 1993). This is used to discourage the weights from taking large values and encourage small weights to decay rapidly to zero. To achieve this goal, we make use of the following penalty function (Hinton 1989; Weigend et al. 1988):

\[ P(w,v) = \epsilon_1 \left( \sum_{m=1}^{h} \sum_{l=1}^{n} \frac{\beta w_{ml}^2}{1 + \beta w_{ml}^2} + \sum_{m=1}^{h} \sum_{p=1}^{c} \frac{\beta v_{pm}^2}{1 + \beta v_{pm}^2} \right) + \epsilon_2 \left( \sum_{m=1}^{h} \sum_{l=1}^{n} w_{ml}^2 + \sum_{m=1}^{h} \sum_{p=1}^{c} v_{pm}^2 \right) \]  

(4.3)

Where \( \epsilon_1 \) and \( \epsilon_2 \) are small positive weight decay constants and \( \beta \) is a parameter with a positive value. Following the discussions of this penalty function and the experimental results on a number of well known classification problems given in Hassibi & Stork (1993), for all the experiments reported in this thesis, we used the same values for the parameters involved in the function \( P(w,v) \). These were \( \epsilon_1 = 0.1, \epsilon_2 = 0.0001 \) and \( \beta = 1 \).
We applied the backpropagation algorithm or BFGS algorithm to update network weights and minimize the following error function:

$$\theta(w, v) = E(w, v) + P(w, v)$$

(4.4)

For the descriptions of these two algorithms, the reader should refer to Section 2.3.1.

In a problem with a large number of attributes, it is frequently the case that many of the input attributes are not relevant to the classification of the patterns. Adding the penalty term to the error function and minimizing the resulting augmented function means that connection weights from the irrelevant input nodes to the hidden nodes will have smaller magnitudes, and these connections can be eliminated. The benefits that can be gained by removing these connections are twofold. First, the feature selection can be achieved by identifying the relevant input attributes. Second, it is easier to interpret the resulting network for the purposes of rule extraction when there are fewer connections from the input nodes to the hidden nodes. For example, if the NeuroLinear technique is applied, a smaller number of rules can be extracted when a neural network has fewer connections between input layer and hidden layer.

### 4.2.2 Neural Network Pruning

The pruning approach begins with a fully connected network and removes connections that are deemed to be “redundant” in the network according to the magnitudes of their weights. In the process, input nodes that are not connected to any hidden nodes and hidden nodes that are not connected to any input nodes and/or output nodes can be removed from the network. After pruning, the removed inputs are identified as the features that are irrelevant to the learning problems. The steps of a pruning algorithm are outlined below.

**Neural network pruning algorithm**

1. Pick a fully connected network. Train this network until a predetermined accuracy rate $\eta_1$ is achieved.
2. Make a choice of $\eta_2$, a pruning-threshold that is a positive scalar to determine if a weight can be removed.
3. For each weight $w$ of the network, if
   $$|w| \leq \eta_2,$$
then remove $w$.

4. Retrain the network. If the network meets the predetermined accuracy level $\eta_1$, then go to step 2. Otherwise, stop and use the previous setting of network weights.

The fully connected network is trained first. Then the accuracy of the network is checked. If the classification rate of the network meets the prespecified required accuracy $\eta_1$, then the pruning process is initiated. Before the actual pruning and retraining is done, the current network is saved. Should pruning additional weight(s) and retraining the network fail to give a new set of weights that meet the prespecified accuracy requirement, the saved network would be considered as the smallest network for this particular run.

The value of the pruning threshold $\eta_2$ determines the range of weights to be removed in the network. As discussed in Chapter 2, most penalty-based pruning algorithms require $\eta_2$ to be set manually, which can cause over- or under-removal of the weights of the network. In order to address this limitation, here we propose three different approaches for the choice of a pruning-threshold. The suitability of each approach depends on the network complexity, the nature of the application, the required predictive accuracy of the pruned network and some other factors that will be explained later.

- Pruning threshold $\eta_2$ Selection Approach I: VBS

The first approach is a simple selection technique. It is named Value Based Selection, because it selects $\eta_2$, based on the values of weights in the network. It first searches the minimum and maximum weights according to the absolute values of all weights in the network, then applies equation (4.5) to obtain $\eta_2$.

$$\eta_2 = w_{\text{min}} + (w_{\text{max}} - w_{\text{min}}) \times p$$

(4.5)

where $w_{\text{min}}$ is the minimum absolute value of the weights, $w_{\text{max}}$ is the maximum absolute value of the weights, and $p$ is a small positive value between 0 and 1. The value of parameter $p$ reflects how large the value range of removed weights can be. For example, suppose we have the following set of weights:

-1.3, 2.32, 1.46, -1.94, 2.81, 2.52
The minimum absolute value of the weights \( w_{\text{min}} \) equals 1.3. The maximum absolute value of the weights \( w_{\text{max}} \) equals 2.81. If \( p \) is set to 0.2, then
\[
\eta^2 = 1.3 + (2.81-1.3) \times 0.2 = 1.602
\]
Thus any weights which have their absolute values below or equal to 1.602 are removed, so in this case, -1.3 and 1.46 are removed.

Although the VBS approach guarantees that at least one weight (\( |w| = w_{\text{min}} \)) can be removed for each run, it works more effectively when all values of the weights in the network are close to each other. When the penalty term (equation 4.3) is added to the error function (equation 4.1) during network training, the weights of the connections are prevented from getting larger. At the same time, small weights are made to decay towards zero. The effectiveness of the VBS approach works will be seen from the experimental results presented in later sections.

- **Pruning threshold \( \eta^2 \) Selection Approach II: NBS**

The second approach is named Number Based Selection, which means it selects \( \eta^2 \) based on the existing number of weights in the network.

Assume as given a trained network with a set of all weights \( W \),
\[
W = \{ w_0, w_1, \ldots, w_{n-1} \}
\]
where \( n \) is the total number of weights.

The NBS approach starts by sorting all weights in ascending order of absolute magnitude to form another set of weights \( W' \):
\[
W' \in \{ w_0', w_1', \ldots, w_k', \ldots, w_{n-1}' \}
\]
where \( w_0' \) is the minimum absolute value of the weights, and \( w_{n-1}' \) is the maximum absolute value of the weights. For the ordered set of weights \( W' \), \( \eta^2 \) is obtained by computing:
\[
\eta^2 = w_{\text{Location}}'
\]
\[
\text{Location} = n \times p
\]
where $p$ is a small positive value between 0 and 1. Unlike the parameter $p$ in the VBS approach, $p$ here determines what percentage of total weights will be removed each time. For example, if the current total number of weights $n = 100$ and the value of $p = 0.1$, then 11% of the total number of weights $\{w_0, ..., w_{10}\}$ will be removed for this particular run of the pruning process.

The NBS approach overcomes a drawback of the VBS approach: it guarantees a certain proportion of total weights will be taken out regardless of the value distribution of the weight set. However, if parameter $p$ is set too large, the VBS approach may not be able to arrive at a satisfactorily small network, since it always eliminates a significant quantity of weights for each run, which can make the required training accuracy level difficult to maintain. This is particularly likely to happen when the number of weights $n$ becomes small after running the retraining and pruning processes several times. On the other hand, the pruning process can be time-consuming if the value of $p$ is too small (e.g., if $p = 0$, only one weight is removed for each run). In short, it is not straightforward to find a proper $p$ for the VBS approach. This leads to the next approach, which makes the setting of parameter $p$ automatic instead of manual.

- **Pruning-threshold $\eta$2 Selection Approach III: PBS**

The VBS approach requires the user to specify a value of $p$ that is to be used for deciding the proportion of total weights to be pruned at each run without providing rules to choose this $p$. The third approach, named *Parameter Based Selection*, selects $p$ automatically according to the existing number of weights in the network. As shown in Figure 4.1, a linear function is used to establish an appropriate relationship between the parameter $p$ and the existing weights in the network. It begins with a high value $P$, e.g. 0.5, for pruning a fully connected network with a total number of weights $N$. Then parameter $p$ is updated when the total number of existing weights $n$ changes.
This relationship can be described by the following equation:

\[ p(n) = \frac{P}{N} \times n \]  

Recalling equations 4.6 and 4.7 for the NBS approach, we apply function 4.8 to produce

\[ \eta_2 = w_{Location} \]  

\[ Location = n \times \left( \frac{P}{N} \times n \right) \]  

Figure 4.2 illustrates the difference between the NBS approach and the PBS approach. If initially, the total number of weights is 100 and the initial \( P \) is 0.5, then 50 connections are removed by both NBS and PBS approach for the first run. But if the number of existing weights goes down to 50, then 25 connections are removed by NBS, while only 12 connections are removed by PBS.
In summary, the pruning process attempts to eliminate as many connections as possible from the network, while maintaining the prespecified accuracy rate. During the pruning process relevant attributes of the data are kept, while others are automatically discarded. An example demonstrating the proposed algorithm is shown in the next section.

4.2.3 Method Illustration: Iris Classification Data

In this section, the Iris data set is used to illustrate how features can be selected by a neural network. Ten repetitions of 10-fold cross validation were performed by the three proposed pruning schemes. For a description of the Iris data set and how the data was prepared for each cross-validation trail, the reader should refer to Sections 3.2 and 3.8 of Chapter 3.

- **Experiment 1**

10 different neural networks were trained and pruned by using different training sets and testing sets. The neural networks (with bias nodes) had 4 inputs, 5 hidden nodes, and 3 output nodes. The required training accuracy rate $\eta I$ was set at 98.5%. The networks were trained by the BFGS algorithm, starting from initial weights that had been randomly generated in the interval [-1,1]. By using the three different pruning approaches with an initial parameter $p = 0.2$, the
experimental results are tabulated in Table 4.1. N.N.F.S means “neural network feature selection”; the figures in parentheses are the standard deviations. The P-value was computed to check if there is any significant increase or decrease in the accuracy of the networks with selected input features from the three different pruning schemes compared to the networks with the whole set of attributes as input without pruning. The P-value was also computed for testing if there is any significant difference on the number of neural network connections before and after neural network pruning.

The P-value for the accuracy rates show that there is not a significant difference in the mean accuracy rates before and after neural network feature selections, based on our assumption of a p-value of 0.05 or lower for significance. On the other hand, the large difference in the number of neural network connections before and after three pruning schemes demonstrates the effect of the pruning schemes for reducing the complexity of the neural network architecture. After neural network feature selection, the VBS pruning approach yield a network with an average of 22 connections. In contrast, the average number of connections left in the network after NBS and PBS approach was 24.2, and 25.2, respectively. Among the 10 neural networks, attributes sepal-width, petal-length and petal-width were the most frequently selected features (each of these 3 features was selected by at least 6 neural networks).
Table 4.1. The experimental results for the Iris data set using the neural network feature selection methods (standard deviations are in parentheses)

<table>
<thead>
<tr>
<th></th>
<th>Before N.N.F.S</th>
<th>VBS</th>
<th>NBS</th>
<th>PBS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of attributes</td>
<td>4(0.000)</td>
<td>3.30(0.675)</td>
<td>2.90(0.994)</td>
<td>3.00(0.817)</td>
</tr>
<tr>
<td>Number of connections</td>
<td>40(0.000)</td>
<td>25.20(6.596)</td>
<td>24.20(7.927)</td>
<td>22.00(9.274)</td>
</tr>
<tr>
<td>Accuracy on training set (%)</td>
<td>98.89(0.6)</td>
<td>98.52(0.0)</td>
<td>98.52(0.0)</td>
<td>98.52(0.0)</td>
</tr>
<tr>
<td>Accuracy on testing set (%)</td>
<td>96.67(4.7)</td>
<td>96.00(4.7)</td>
<td>95.34(5.5)</td>
<td>94.67(6.1)</td>
</tr>
<tr>
<td>List of most frequent selected attributes</td>
<td>sepal-length</td>
<td>sepal-width</td>
<td>sepal-width</td>
<td>sepal-width</td>
</tr>
<tr>
<td></td>
<td>sepal-width</td>
<td>petal-length</td>
<td>petal-length</td>
<td>petal-length</td>
</tr>
<tr>
<td></td>
<td>petal-length</td>
<td>petal-width</td>
<td>petal-width</td>
<td>petal-width</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>P-value (Before /After N.N.F.S) Number of connections</th>
<th>-</th>
<th>0.000</th>
<th>0.000</th>
<th>0.000</th>
</tr>
</thead>
</table>

| P-value (Before /After N.N.F.S) Testing set accuracy | - | 0.343 | 0.168 | 0.081 |

One representative network resulting from the PBS approach is depicted in Figure 4.3. It has only 3 hidden nodes, 3 input nodes with connections to hidden nodes and a total 13 connections. The attribute *sepal-length* was distinguished as an irrelevant attribute, since it was not connected to any of the hidden nodes.
Figure 4.3. A network with 13 weights for Iris classification. The number next to a connection shows the weights for that connection. The accuracy rates for the training and testing set are 98.52% and 93.33%, respectively.

For comparison purposes, the experimental results of applying the Chi2 algorithm and the C4.5 learning technique are presented in Table 4.2, where “Before” and “After” means the C4.5 result was obtained before or after using Chi2. For the Chi2 algorithm, the inconsistency rate $\delta$ was set to 2% – that means only 3 (135×0.02) inconsistent results were acceptable after discretization. Each of the 10 different cross-validation trials was performed using the Chi2 algorithm, then both the original data and the dimensionally reduced ones were analyzed using C4.5 with its default settings (Quinlan 1993). Note that here, the dimensionally reduced data refers to the original data with a reduced feature space, not the size-reduced discrete data.

The most frequent selected features shown in Table 4.2 were those that were selected at least 6 times by the Chi2 algorithm based on the 10 cross-validation trials.
Table 4.2. The predictive accuracies and tree size of C4.5 before and after Chi2 for the Iris data set (standard deviations are in parentheses)

<table>
<thead>
<tr>
<th></th>
<th>Before</th>
<th>After</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of attributes</td>
<td>4(0.000)</td>
<td>3.3(0.675)</td>
<td>0.010</td>
</tr>
<tr>
<td>Number of data items</td>
<td>135(0.000)</td>
<td>25.1(10.100)</td>
<td>0.000</td>
</tr>
<tr>
<td>Tree Size</td>
<td>8.4(0.966)</td>
<td>8.2(1.033)</td>
<td>0.343</td>
</tr>
<tr>
<td>Accuracy on training set</td>
<td>98.09(0.5)</td>
<td>98.09(0.5)</td>
<td>-</td>
</tr>
<tr>
<td>Accuracy on testing set</td>
<td>94.66(5.3)</td>
<td>94.66(5.3)</td>
<td>-</td>
</tr>
<tr>
<td>List of most frequent selected attributes</td>
<td>sepal-length, sepal-width, petal-length, petal-width</td>
<td>sepal-width, petal-length, petal-width</td>
<td>-</td>
</tr>
</tbody>
</table>

### Experiment 2

In order to test if noise attributes can be removed by the proposed feature selection algorithm, a modified version of the Iris data set was prepared. The modified version of the Iris data set includes the 4 original attributes and an extra 5 noise attributes. The four original attributes sepal-length, sepal-width, petal-length and petal-width were denoted by A0, A1, A2 and A3, and the five noise attributes were labelled A4, A5, A6, A7 and A8. The attributes A4, A5, A6 and A7 were associated with attributes A0, A1, A2, and A3, respectively. The values of attributes A4 – A7 were determined by a standard normal distribution between the ave and (max - min)/6, where ave, max and min are the mean, maximum and minimum values of the corresponding original attributes A0 – A3. Attribute A8 was generated from a uniform distribution between 0 and 10.

In the experiment, a fully connected neural network was used for training by 10-fold cross validation. The neural network consisted of 9 input nodes with a bias, 5 hidden nodes and 3 output nodes. The trained network was pruned until its accuracy on the training data dropped below 98.5%. The parameter p was initially set to 0.2 for the three pruning methods. The results of the experiments are summarized in Table 4.3. In this table, the number of attributes before and after pruning by the three different pruning approaches and their standard deviations are given, and the associated P-values are shown. All three pruning schemes successfully eliminated the 5
noise attributes and identified the attribute *sepal-length* as an irrelevant feature. A pruned network using VBS is depicted in Figure 4.4.

**Table 4.3.** The summarized neural network results on the modified Iris data set for three pruning methods (standard deviations are in parentheses)

<table>
<thead>
<tr>
<th></th>
<th>VBS</th>
<th>NBS</th>
<th>PBS</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of attributes before pruning</td>
<td>9(0.000)</td>
<td>9(0.000)</td>
<td>9(0.000)</td>
</tr>
<tr>
<td>No. of attributes after pruning</td>
<td>2.80(1.033)</td>
<td>2.90(0.568)</td>
<td>3.00(0.667)</td>
</tr>
<tr>
<td>P-value</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
</tbody>
</table>

**Figure 4.4.** Pruned network with 9 connections.

**4.3 Experiments**

In the last section, we demonstrated the proposed neural network feature selection methods applied to the Iris data set. In order to show the effectiveness of the algorithms, this section presents the results of their applications to five practical problems tested through 10-fold cross validation. For comparison purposes, the experimental results of applying the Chi2 algorithm and the C4.5 learning method on the same data sets are also presented. All the experimental results are analyzed and assessed in Section 4.4.
As previously indicated, the initial $p$ for the three neural network pruning schemes must be specified. In order to avoid over-removing the weights of the network, the parameter $p$ for the VBS and NBS approaches is normally set to be very small, e.g., in the range of $0.01 \leq p \leq 0.15$. However, due to the PBS approach’s automatic parameter selection ability, the initial $P$ of the PBS approach can be set to a larger value, e.g., in the range of $0.15 \leq P \leq 1$. In the experiments associated with these three neural network pruning schemes, different values of parameter $p$ were tested. Only those values providing the best overall performance when considering both the simplicity and accuracy of the network are presented in this section.

### 4.3.1 Pima Indians Diabetes Data

For a description of the Pima Indians Diabetes data set and how the data was prepared for each cross-validation trial, the reader is referred to Sections 3.3 and 3.8 of Chapter 3.

Table 4.4 presents the experimental results using the Chi2 algorithm with an inconsistency rate $\delta = 5\%$, where $I_1 - I_8$ were explained in Table 3.3 of Chapter 3. The C4.5 analysis technique was applied to both the original data sets and the data sets with a reduced feature space. The predictive accuracies and tree sizes of C4.5 before and after the Chi2 processing, along with their associated P-values, are also presented in Table 4.4.

<table>
<thead>
<tr>
<th></th>
<th>Before Chi2</th>
<th>After Chi2</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of attributes</td>
<td>8(0.000)</td>
<td>6.9(0.738)</td>
<td>0.001</td>
</tr>
<tr>
<td>Number of data items</td>
<td>691.2(0.400)</td>
<td>472.1(66.200)</td>
<td>0.000</td>
</tr>
<tr>
<td>Tree Size</td>
<td>46.4(13.790)</td>
<td>41.2(15.530)</td>
<td>0.112</td>
</tr>
<tr>
<td>Accuracy on training set (%)</td>
<td>84.68(2.6)</td>
<td>83.51(2.6)</td>
<td>0.022</td>
</tr>
<tr>
<td>Accuracy on testing set (%)</td>
<td>73.94(4.6)</td>
<td>73.41(4.0)</td>
<td>0.501</td>
</tr>
<tr>
<td>List of most frequent selected attributes</td>
<td>$I_1 - I_8$</td>
<td>$I_1, I_2, I_4 - I_8$</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 4.4. The experimental results for the Pima Indians Diabetes data set using Chi2 feature selection and C4.5 classification (standard deviations are in parentheses)
For neural network feature selection methods, neural networks with 8 input nodes with a bias, 5 hidden nodes and 2 output nodes were trained by the BFGS algorithm. The required training accuracy \( \eta \) was set to 79% as this accuracy can be achieved by the original un-pruned neural networks. Three different pruning schemes with an initial \( p = 0.15 \) were applied to the networks. The experimental results are summarized in Table 4.5.

**Table 4.5.** The experimental results for the Pima Indians Diabetes data set using the neural network feature selection methods (standard deviations are in parentheses)

<table>
<thead>
<tr>
<th></th>
<th>Before N.N.F.S</th>
<th>After N.N.F.S</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>VBS</td>
<td>NBS</td>
</tr>
<tr>
<td>Number of attributes</td>
<td>8(0.000)</td>
<td>7.7(0.483)</td>
</tr>
<tr>
<td>Number of connections</td>
<td>55(0.000)</td>
<td>32.90(9.360)</td>
</tr>
<tr>
<td>Accuracy on training set (%)</td>
<td>79.36(0.8)</td>
<td>79.59(0.6)</td>
</tr>
<tr>
<td>Accuracy on testing set (%)</td>
<td>77.06(3.8)</td>
<td>76.03(4.4)</td>
</tr>
<tr>
<td>List of most frequent selected attributes</td>
<td>I_1 - I_8</td>
<td>I_1 - I_8</td>
</tr>
<tr>
<td>P-value (Before /After N.N.F.S)</td>
<td>-</td>
<td>0.000</td>
</tr>
<tr>
<td>Number of connections</td>
<td>-</td>
<td>0.284</td>
</tr>
</tbody>
</table>

**4.3.2 Wine Recognition Data**

For a description of the Wine Recognition data set and how the data was prepared for each cross-validation trial, the reader is referred to Sections 3.4 and 3.8 of Chapter 3.

The results of applying the Chi2 algorithm to the Wine Recognition data are shown in Table 4.6, where \( I_1 - I_{13} \) were explained in Table 3.5 of Chapter 3. The inconsistency rate \( \delta = 1\% \) was
allowed in the experiment. C4.5 with its default setting was used to check if the Chi2 algorithm had seriously degraded the performance of the learning method.

**Table 4.6.** The experimental results for the Wine Recognition data set using Chi2 feature selection and C4.5 classification (standard deviations are in parentheses)

<table>
<thead>
<tr>
<th></th>
<th>Before Chi2</th>
<th>After Chi2</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of attributes</td>
<td>13(0.000)</td>
<td>5.6(0.843)</td>
<td>0.000</td>
</tr>
<tr>
<td>Number of data items</td>
<td>160.2(0.420)</td>
<td>39.4(11.080)</td>
<td>0.000</td>
</tr>
<tr>
<td>Tree Size</td>
<td>9.2(0.632)</td>
<td>11.2(0.632)</td>
<td>-</td>
</tr>
<tr>
<td>Accuracy on training set (%)</td>
<td>98.90(0.58)</td>
<td>98.85(0.4)</td>
<td>0.673</td>
</tr>
<tr>
<td>Accuracy on testing set (%)</td>
<td>92.73(7.4)</td>
<td>94.92(6.2)</td>
<td>0.354</td>
</tr>
<tr>
<td>List of most frequent selected attributes</td>
<td>I₁ - I₁₃</td>
<td>I₁, I₃, I₆, I₇, I₁₀, I₁₂</td>
<td>-</td>
</tr>
</tbody>
</table>

For neural network feature selection methods, networks with 13 input nodes with a bias, 5 hidden nodes, and 3 output nodes were trained and pruned. The required training accuracy $\eta$ was set to be 100%. The parameter $p$ for the VBS and NBS pruning approaches was set to 0.01 and 0.05, respectively. For the PBS approach, the initial $P$ was 0.2. The experimental results are summarized in Table 4.7.
Table 4.7. The experimental results for the Wine Recognition data set using the neural network feature selection methods (standard deviations are in parentheses)

<table>
<thead>
<tr>
<th></th>
<th>Before N.N.F.S</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Number of attributes</td>
<td>13(0.000)</td>
<td>6.6(0.699)</td>
<td>6.6(0.699)</td>
<td>6.5(1.080)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Number of connections</td>
<td>85(0.000)</td>
<td>13.8(1.814)</td>
<td>13.8(1.814)</td>
<td>13.6(2.503)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Accuracy on training set (%)</td>
<td>100(0.0)</td>
<td>100(0.0)</td>
<td>100(0.0)</td>
<td>100(0.0)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Accuracy on testing set (%)</td>
<td>96.59(3.9)</td>
<td>96.56(4.8)</td>
<td>96.56(4.8)</td>
<td>95.40(6.6)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>List of most frequent selected attributes</td>
<td>$I_1 - I_{13}$</td>
<td>$I_1$ $I_3$ $I_7$ $I_{10}$ $I_{11}$ $I_{13}$</td>
<td>$I_1$ $I_3$ $I_7$ $I_{10}$ $I_{11}$ $I_{13}$</td>
<td>$I_1$ $I_3$ $I_7$ $I_{10}$ $I_{11}$ $I_{13}$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>P-value (Before /After N.N.F.S) Number of connections</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>P-value (Before /After N.N.F.S) Testing set accuracy</td>
<td>0.986</td>
<td>0.986</td>
<td>0.537</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4.3.3 Golf Course Problem

For a description of the Golf Course Problem and how the data was prepared for each cross-validation trial, the reader is referred to Sections 3.5 and 3.8 of Chapter 3.

The experimental results before and after using Chi2 algorithm and C4.5 are summarized in Table 4.8, where $I_1 - I_4$ were explained in Table 3.7 of Chapter 3. The inconsistency rate $\delta$ was set to 5%.
Table 4.8. The experimental results for the Golf Course Problem using Chi2 feature selection and C4.5 classification (standard deviations are in parentheses)

<table>
<thead>
<tr>
<th></th>
<th>Before Chi2</th>
<th>After Chi2</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of attributes</td>
<td>4(0.000)</td>
<td>4(0.000)</td>
<td>-</td>
</tr>
<tr>
<td>Number of data items</td>
<td>1372.5(0.530)</td>
<td>365.6(30.360)</td>
<td>0.000</td>
</tr>
<tr>
<td>Tree Size</td>
<td>34.8(1.476)</td>
<td>34.8(1.476)</td>
<td>-</td>
</tr>
<tr>
<td>Accuracy on training set (%)</td>
<td>74.51(0.6)</td>
<td>74.51(0.6)</td>
<td>-</td>
</tr>
<tr>
<td>Accuracy on testing set (%)</td>
<td>70.67(3.3)</td>
<td>70.67(3.3)</td>
<td>-</td>
</tr>
<tr>
<td>List of most frequent selected attributes</td>
<td>I₁ - I₄</td>
<td>I₁ - I₄</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 4.9 presents the experimental results after the training and pruning of neural networks with 4 inputs plus a bias, 10 hidden nodes and 5 output nodes. The parameter $p$ for the VBS and NBS approach was set to 0.05. The initial $P$ for the PBS approach was 0.15. The required training accuracy $\eta I$ was set at 73% as this accuracy can be achieved by the original un-pruned neural networks.
Table 4.9. The experimental results for the Golf Course data using the neural network feature selection methods (standard deviations are in parentheses)

<table>
<thead>
<tr>
<th></th>
<th>Before N.N.F.S</th>
<th>After N.N.F.S</th>
<th>VBS</th>
<th>NBS</th>
<th>PBS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of attributes</td>
<td>4(0.000)</td>
<td>4(0.000)</td>
<td>4(0.000)</td>
<td>4(0.000)</td>
<td></td>
</tr>
<tr>
<td>Number of connections</td>
<td>100(0.000)</td>
<td>63.7(3.65)</td>
<td>63.1(3.540)</td>
<td>64.4(4.600)</td>
<td></td>
</tr>
<tr>
<td>Accuracy on training set (%)</td>
<td>73.58(0.5)</td>
<td>75.20(0.2)</td>
<td>75.28(0.2)</td>
<td>75.19(0.1)</td>
<td></td>
</tr>
<tr>
<td>Accuracy on testing set (%)</td>
<td>73.40(3.3)</td>
<td>73.64(2.8)</td>
<td>73.80(3.6)</td>
<td>73.39(2.9)</td>
<td></td>
</tr>
<tr>
<td>List of most frequent selected attributes</td>
<td>$I_1 - I_4$</td>
<td>$I_1 - I_4$</td>
<td>$I_1 - I_4$</td>
<td>$I_1 - I_4$</td>
<td></td>
</tr>
<tr>
<td>P-value (Before /After N.N.F.S) Number of connections</td>
<td>-</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td></td>
</tr>
<tr>
<td>P-value (Before /After N.N.F.S) Testing set accuracy</td>
<td>-</td>
<td>0.763</td>
<td>0.525</td>
<td>0.986</td>
<td></td>
</tr>
</tbody>
</table>

4.3.4 Cook Islands Sea Cucumber Habitat Data

For a description of the Cook Islands Sea Cucumber Habitat data set and how the data was prepared for each cross-validation trial, the reader is referred to Sections 3.6 and 3.8 of Chapter 3.

The experimental results before and after using the Chi2 algorithm and C4.5 are summarized in Table 4.10, where $I_1 - I_{10}$ were explained in Table 3.9 of Chapter 3. The inconsistency rate $\delta$ was set to 1%.
Table 4.10. The experimental results for the Cook Islands Sea Cucumber data using Chi2 feature selection and C4.5 classification (standard deviations are in parentheses)

<table>
<thead>
<tr>
<th></th>
<th>Before Chi2</th>
<th>After Chi2</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of attributes</td>
<td>10(0.000)</td>
<td>6.7(0.823)</td>
<td>0.000</td>
</tr>
<tr>
<td>Number of data items</td>
<td>115.2(0.420)</td>
<td>62.6(7.760)</td>
<td>0.000</td>
</tr>
<tr>
<td>Tree Size</td>
<td>11.8(5.010)</td>
<td>12.4(3.270)</td>
<td>0.541</td>
</tr>
<tr>
<td>Accuracy on training set (%)</td>
<td>93.38(2.3)</td>
<td>92.32(4.2)</td>
<td>0.299</td>
</tr>
<tr>
<td>Accuracy on testing set (%)</td>
<td>81.91(9.3)</td>
<td>82.68(9.2)</td>
<td>0.343</td>
</tr>
<tr>
<td>List of most frequent selected attributes</td>
<td>I₁ - I₁₀</td>
<td>I₁ - I₇</td>
<td>-</td>
</tr>
</tbody>
</table>

For neural network feature selections, neural networks with 11 input nodes plus a bias, 3 hidden nodes, and 2 output nodes were trained and pruned. Table 4.11 presents the experimental results, where X₁ - X₁₁ were explained in Table 3.13 of Chapter 3. The parameter $p$ for the VBS and NBS approach was set to 0.15. The initial $P$ for the PBS approach was 0.2. The required training accuracy $\eta I$ was 90% as this accuracy can be achieved by the original un-pruned neural networks.
**Table 4.11.** The experimental results for the Cook Islands Sea Cucumber data using the neural network feature selection methods (standard deviations are in parentheses)

<table>
<thead>
<tr>
<th></th>
<th>Before N.N.F.S</th>
<th>After N.N.F.S</th>
<th>VBS</th>
<th>NBS</th>
<th>PBS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of attributes</td>
<td>11(0.000)</td>
<td>6.8(2.486)</td>
<td>6.9(1.792)</td>
<td>7.0(1.491)</td>
<td></td>
</tr>
<tr>
<td>Number of connections</td>
<td>42(0.000)</td>
<td>18.2(9.680)</td>
<td>15.0(3.560)</td>
<td>15.7(4.300)</td>
<td></td>
</tr>
<tr>
<td>Accuracy on training set (%)</td>
<td>91.48(1.3)</td>
<td>90.78(0.6)</td>
<td>90.68(0.8)</td>
<td>91.04(0.8)</td>
<td></td>
</tr>
<tr>
<td>Accuracy on testing set (%)</td>
<td>85.95(8.7)</td>
<td>83.45(10.2)</td>
<td>84.34(7.3)</td>
<td>84.34(7.3)</td>
<td></td>
</tr>
<tr>
<td>List of most frequent selected attributes</td>
<td>$X_1 - X_{11}$</td>
<td>$X_2 - X_5, X_7 - X_9$</td>
<td>$X_2 - X_9$</td>
<td>$X_2 - X_9$</td>
<td></td>
</tr>
<tr>
<td>P-value (Before /After N.N.F.S)</td>
<td>-</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td></td>
</tr>
<tr>
<td>Number of connections</td>
<td>-</td>
<td>0.373</td>
<td>0.333</td>
<td>0.333</td>
<td></td>
</tr>
<tr>
<td>Testing set accuracy</td>
<td>-</td>
<td>0.000</td>
<td>0.333</td>
<td>0.333</td>
<td></td>
</tr>
</tbody>
</table>

**4.3.5 New Zealand Asthma Incidence Data**

For a description of the New Zealand Asthma Incidence data set and how the data was prepared for each cross-validation trial, the reader is referred to Sections 3.7 and 3.8 of Chapter 3.

Table 4.12 summarizes all related parameters for the experiments using Chi2 and the neural network methods. Table 4.13 presents the results of experiments conducted by Chi2 and the C4.5 methods, where $I_1 - I_9$ were explained in Table 3.11 of Chapter 3.
Table 4.12. Experimental parameter settings for the New Zealand Asthma Incidence data

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inconsistency rate $\delta$</td>
<td>2%</td>
</tr>
<tr>
<td>NN required training accuracy $\eta I$</td>
<td>68%</td>
</tr>
<tr>
<td>$p$ for VBS</td>
<td>0.05</td>
</tr>
<tr>
<td>$p$ for NBS</td>
<td>0.15</td>
</tr>
<tr>
<td>Initial $P$ for PBS</td>
<td>0.2</td>
</tr>
</tbody>
</table>

Table 4.13. The experimental results for the New Zealand Asthma Incidence data using Chi2 feature selection and C4.5 classification (standard deviations are in parentheses)

<table>
<thead>
<tr>
<th></th>
<th>Before Chi2</th>
<th>After Chi2</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of attributes</td>
<td>9(0.000)</td>
<td>7.3(0.675)</td>
<td>0.000</td>
</tr>
<tr>
<td>Number of data items</td>
<td>900(0.000)</td>
<td>873.9(3.280)</td>
<td>0.000</td>
</tr>
<tr>
<td>Tree Size</td>
<td>219.6(13.100)</td>
<td>183.8(46.900)</td>
<td>0.041</td>
</tr>
<tr>
<td>Accuracy on training set (%)</td>
<td>73.19(0.7)</td>
<td>70.32(2.4)</td>
<td>0.005</td>
</tr>
<tr>
<td>Accuracy on testing set (%)</td>
<td>49.00(3.4)</td>
<td>48.80(3.4)</td>
<td>0.758</td>
</tr>
<tr>
<td>List of most frequent selected</td>
<td>$I_1 - I_9$</td>
<td>$I_1 - I_3$, $I_5 - I_7$, $I_9$</td>
<td>-</td>
</tr>
<tr>
<td>attributes</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For the neural network feature selection, including the input for bias, a total of 75 input nodes, 20 hidden nodes and 2 output nodes were present in the original neural networks. The experimental results are shown in Table 4.14, where $X_1 - X_{74}$ were explained in Table 3.14 of Chapter 3.
Table 4.14. The experimental results for the New Zealand Asthma Incidence data using the neural network feature selection methods (standard deviations are in parentheses)

<table>
<thead>
<tr>
<th></th>
<th>Before N.N.F.S</th>
<th>After N.N.F.S</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>VBS</td>
<td>NBS</td>
</tr>
<tr>
<td>Number of attributes</td>
<td>74(0.000)</td>
<td>66.7(3.455)</td>
</tr>
<tr>
<td>Number of connections</td>
<td>1560(0.000)</td>
<td>949.8(44.800)</td>
</tr>
<tr>
<td>Accuracy on training set (%)</td>
<td>68.61(2.9)</td>
<td>68.42(0.3)</td>
</tr>
<tr>
<td>Accuracy on testing set (%)</td>
<td>51.70(3.0)</td>
<td>51.50(4.6)</td>
</tr>
<tr>
<td>List of most frequent selected attributes</td>
<td>$X_1 - X_{74}$</td>
<td>$X_1, X_2, X_4 - X_{44}$, $X_{47}, X_{49} - X_{74}$</td>
</tr>
<tr>
<td>P-value (Before /After N.N.F.S)</td>
<td>-</td>
<td>0.000</td>
</tr>
<tr>
<td>Number of connections</td>
<td>-</td>
<td>0.900</td>
</tr>
</tbody>
</table>

4.4 Performance Assessment

Chapter 1 introduced the dimensions along which feature selection methods should be evaluated: predictive accuracy, complexity of representations, feature reduction, computing time, and generality of selected features. This section presents an analysis of the experiments that were described in the previous section in terms of those five evaluative criteria.

4.4.1 Predictive Accuracy

With all six problem domains, the P-values for the accuracy rates on the testing sets in Tables 4.2, 4.4, 4.6, 4.8, 4.10, and 4.13 show that there is no significant difference in the mean accuracy rates of the C4.5 decision tree procedure when the Chi2 procedure is used as a feature selection method. For the New Zealand Asthma Incidence data set, the C4.5 procedure failed regardless of whether Chi2 was applied or not. For the remaining five data sets, although it is not statistically significant, in only one case (Pima Indians Diabetes data set) did
the C4.5 procedure have slightly poorer performance after feature selection (73.94\% compared with 73.41\%).

For the Golf Course Problem, all the original input features remained after Chi2 feature selection, hence feature selection had no impact in terms of the number of data items used by the C4.5 procedure. However, if the size-reduced discrete data items generated by Chi2 were input to the C4.5 procedure, a mean accuracy rate on the test set was 65.63±7.95\%, which is significantly worse at the 99\% level than with the initial data set without Chi2 feature selection. The decrease in accuracy might be due to the C4.5 procedure’s lesser capability in connection with discrete data than with continuous data.

With respect to the three neural network feature selection schemes, all three have demonstrated their strong generalization abilities. The P-values for the accuracy rates in Tables 4.1, 4.5, 4.7, 4.9, 4.11, and 4.14 show that there is no significant difference in the mean accuracy rates before and after neural network feature selection on the empirical six problem sets.

When comparing the accuracy performance of neural network feature selection methods with the performance of C4.5 with features selected by the Chi2 method, the neural network feature selection methods perform better on all six data sets if statistical significance is disregarded. When statistical significance is considered, the P-values are summarized in Table 4.15. In three cases – the Pima Indians Diabetes data set, the Golf Course Problem, and the New Zealand Asthma Incidence data set, the neural network feature selection methods perform significantly better than the C4.5 with features selected by the Chi2 method.
Table 4.15. The P-values associated with predictive accuracy when comparing the three neural network feature selection schemes with C4.5 procedure based on the Chi2-selected features

<table>
<thead>
<tr>
<th></th>
<th>VBS/C4.5 with Chi2</th>
<th>NBS/C4.5 with Chi2</th>
<th>PBS/C4.5 with Chi2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iris</td>
<td>0.997</td>
<td>0.673</td>
<td>0.340</td>
</tr>
<tr>
<td>Pima Indians Diabetes</td>
<td>0.050</td>
<td>0.014</td>
<td>0.029</td>
</tr>
<tr>
<td>Wine Recognition</td>
<td>0.554</td>
<td>0.554</td>
<td>0.869</td>
</tr>
<tr>
<td>Golf Course Problem</td>
<td>0.016</td>
<td>0.014</td>
<td>0.016</td>
</tr>
<tr>
<td>Cook Islands Sea Cucumber Habitat</td>
<td>0.800</td>
<td>0.432</td>
<td>0.698</td>
</tr>
<tr>
<td>New Zealand Asthma Incidence</td>
<td>0.148</td>
<td>0.012</td>
<td>0.011</td>
</tr>
</tbody>
</table>

It should be mentioned that the difference in predictive accuracy between C4.5 based on the selected features and neural network learning is likely due to the difference of learning ability between the C4.5 decision tree model and the neural network learning model. Some detailed descriptions of these two learning models were provided in Sections 2.3.1 and 2.5.2 of Chapter 2. For discussions and experiments concerning the difference in their learning abilities, the reader should consult (German et al. 1999).

4.4.2 Complexity of Representations

After Chi2 feature selection, the tree size of C4.5 does not always decrease. In fact, a decrease is only observed in connection with two applications (the Pima Indians Diabetes data set and the New Zealand Asthma Incidence data set). This shows that although the feature spaces of data sets can be reduced by the Chi2 algorithm, the dimensionally reduced data representation might require more complex tree structures to be interpreted. However, after neural network feature selection, the complexity of the neural network architecture is always reduced. A summary of the number of neural network connections before and after neural network pruning schemes for all six data sets is provided in Table 4.16. It can be seen that the reduction of neural network connections can be as much as 84% of the original size (see the number of connections for the Wine Recognition data set in Table 4.16). Note that the
simplified neural network can not only identify the relevant input attributes but also provides a good foundation for the use of rule extraction techniques to extract more comprehensible rules.

Table 4.16. A summary of number of neural network connections before and after neural network selection methods on six data sets (standard deviations are in parentheses)

<table>
<thead>
<tr>
<th></th>
<th>Before N.N.F.S</th>
<th>After N.N.F.S</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>VBS</td>
</tr>
<tr>
<td>Iris</td>
<td>40(0.000)</td>
<td>25.20(6.596)</td>
</tr>
<tr>
<td>Pima Indians Diabetes</td>
<td>55(0.000)</td>
<td>32.90(9.360)</td>
</tr>
<tr>
<td>Wine Recognition</td>
<td>85(0.000)</td>
<td>13.8(1.814)</td>
</tr>
<tr>
<td>Golf Course Problem</td>
<td>100(0.000)</td>
<td>63.7(3.65)</td>
</tr>
<tr>
<td>Cook Islands Sea Cucumber Habitat</td>
<td>42(0.000)</td>
<td>18.2(9.680)</td>
</tr>
<tr>
<td>New Zealand Asthma Incidence</td>
<td>1560(0.000)</td>
<td>949.8(44.800)</td>
</tr>
</tbody>
</table>

Comparing the three different neural network feature selection schemes, the PBS approach has produced the simplest networks with the fewest connections on three applications, and produced the second simplest neural networks on two other applications.

4.4.3 Number of Features Selected and Generality of the Selected Features

If we know *a priori* that some features are relevant, we can expect a good feature selection method to find these features, either in the minimum subset or at the front of a ranked list. Knowing that some features are irrelevant or redundant can also help. Just like the experiments on the modified version of the Iris data set described in Section 4.2.3, the neural network feature selection approaches have shown their ability to remove noise features. But this type of evaluation is not always possible in many real-world applications. Without knowing data characteristics, indirect evaluation of features must be employed (Liu & Motoda 1998).
Indirect evaluation means that, instead of checking whether features are relevant or not, a learning algorithm is evaluated to see whether using selected features can improve or at least maintain performance. Since one of the ultimate goals of feature selection is to improve or maintain predictive accuracy, as long as the accuracy improves or stays approximately the same, we can assume that the selected features are relevant. As already indicated, with Chi2-selected features, C4.5 performs equally well on all six applications compared with its performance with the original feature set, and neural networks have significantly better performance than C4.5 on Chi2-selected features in three applications. So both the Chi2 algorithm and the neural network feature selection approaches have demonstrated their capability of selecting relevant features for the given problem sets. A summary of the number of features before and after applying the Chi2 algorithm and the neural network feature selection approaches on six data sets is provided in Table 4.17.

Table 4.17. A summary of number of features before and after applying the Chi2 algorithm and neural network feature selection approaches on six data sets

<table>
<thead>
<tr>
<th></th>
<th>Before Chi2</th>
<th>After Chi2</th>
<th>Before N.N.F.S</th>
<th>After N.N.F.S</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>VBS</td>
<td>NBS</td>
<td>PBS</td>
<td></td>
</tr>
<tr>
<td>Iris</td>
<td>4(0.000)</td>
<td>3.3(0.675)</td>
<td>4(0.000)</td>
<td>3.3(0.675)</td>
</tr>
<tr>
<td>Pima Indians Diabetes</td>
<td>8(0.000)</td>
<td>6.9(0.738)</td>
<td>8(0.000)</td>
<td>7.7(0.483)</td>
</tr>
<tr>
<td>Wine Recognition</td>
<td>13(0.000)</td>
<td>5.6(0.843)</td>
<td>13(0.000)</td>
<td>6.6(0.699)</td>
</tr>
<tr>
<td>Golf Course Problem</td>
<td>4 (0.000)</td>
<td>4(0.000)</td>
<td>4(0.000)</td>
<td>4(0.000)</td>
</tr>
<tr>
<td>Cook Islands Sea Cucumber Habitat</td>
<td>10(0.000)</td>
<td>6.7(0.823)</td>
<td>11(0.000)</td>
<td>6.8(2.486)</td>
</tr>
<tr>
<td>New Zealand Asthma Incidence</td>
<td>9(0.000)</td>
<td>7.3(0.675)</td>
<td>74 (0.000)</td>
<td>66.7(3.455)</td>
</tr>
</tbody>
</table>

When features are selected, it is also useful to know whether the same set of features are selected by different techniques. If possible, we often hope that the relevant features are selected independent of any feature selection method. In this context, we take a closer look at
the selected features derived by the different feature selection methods on the six learning applications:

- For the Iris data set, the attributes *sepal-width*, *petal-length* and *petal-width* were selected by both approaches.
- For the Pima Indians Diabetes data set, the neural network approaches selected all 8 input attributes as relevant features, while the Chi2 method selected 7 relevant features.
- For the Wine Recognition data set, both the Chi2 method and the neural network selection schemes selected 6 relevant features. Although two different sets of selected features were generated by the Chi2 method and the neural network approaches, 4 of the selected features were common to the two approaches.
- For the Golf Course Problem, all 4 input attributes were considered to be relevant by the Chi2 feature selection method and the neural network feature selection approaches.
- For the Cook Islands Sea Cucumber data set, the Chi2 method selected 7 relevant features, while 7 or 8 features were considered to be relevant by three neural network feature selection methods (PBS and NBS selected 7 relevant features, and NBS selected 8 relevant features). Attributes %mod/silt and %gravel were removed by the Chi2 method and the three neural network approaches.
- For the New Zealand Asthma Incidence data set, the attributes *Mean temperature*, *Humidity* and *Sunshine* were removed by the Chi2 method, while the neural network feature selection approaches retained them as relevant features. Since the neural network has a superior ability to handle discrete attributes, it has given us more insight into the problem. For example, all three neural network feature selection approaches removed the input attribute $X_3$ (Table 4.14), which indicated that the attribute $X_3$ that represents “ethnicity Pacific Islander” did not have any influence to on the output classification. This kind of feature reduction is not possible obtained from the Chi2 method due to its limitation of working on the numeric attributes only.

In summary, the selected features from the neural network methods overlap those selected by the Chi2 algorithm on five applications (the Iris data set, the Pima Indians Diabetes data set, the Wine Recognition data set, the Golf Course Problem, and the Cook Islands Sea Cucumber
In only one case – the New Zealand Asthma Incidence data set, the neural networks-selected features do not match those from the Chi2 algorithm.

4.4.4 Computing Time

Having training and pruning processes as their essential procedures, neural network feature selection approaches are computationally more expensive than the $\chi^2$ statistic-based algorithm. However the differences can be reduced if computationally efficient training procedures are employed in connection with neural networks. For our experiments, we implemented the BFGS search method which resulted in a speedup of up to two orders of magnitude over the traditional backpropagation method. For example, with the neural network feature selection approach PBS, the neural network employing back-propagation on the Iris data set took about 43 seconds on a Pentium III platform (Intel 686 processor Model 7 Stepping 3) with 128 Mbytes of memory, and the neural network employing BFGS took 4 seconds to achieve a similar result on the same computer. However, the Chi2 method only spent 2 seconds to complete the procedure.

4.5 Summary

In this chapter, three feature selection approaches that employ neural networks and associated pruning schemes have been described. The neural network algorithms have been evaluated and compared with the Chi2 algorithm by experimenting with six practical applications. The experimental analysis has shown that the proposed neural network feature selection algorithms have a potentially useful capability to select relevant features by removing a considerable number of network connections while maintaining good predictive accuracy. The proposed algorithms offer a different approach from those of statistical-based methods and have been demonstrated to be effective for selecting relevant input attributes for classification purposes. The experimental results have also shown that the $\chi^2$ statistic-based feature selection method, Chi2, is a simple and efficient tool for feature selection and data reduction.

In the next chapter, more applications of Chi2 algorithm will be presented.
Chapter 5
Applications of the $\chi^2$ Statistic-based Discretization Algorithm

5.1 Introduction
As described in Chapter 2, a $\chi^2$ statistic-based discretization algorithm, Chi2, can select relevant features according to the characteristics of the data. In addition, it performs automatic discretization, which can reduce the number of data items. This chapter first describes how to apply the Chi2 algorithm to large spatial data sets in order to perform spatial data filtering, and then presents a Chi2 algorithm-based membership function selection method for fuzzy systems, in connection with applications of fuzzy neural networks. In order to demonstrate the effectiveness of the Chi2-based membership function selection method, experiments with three data sets are then described.

5.2 Spatial Data Filtering
The increasing availability of large stores of spatial data has led to demands for improved methods for analyzing these data sets. Neuro-fuzzy techniques have some advantages in connection with managing and analyzing these geographical data sets (Purvis et al. 1999). In the case of artificial neural networks, their capability of approximating any continuous function to any desired degree of precision, without the need for specifying the type of function (Cybenko 1989), makes neural networks good candidates to analyse and process spatial data. In addition, neural networks can be applied to incomplete or imprecise data and still yield acceptable results, which makes them particular suitable for the analysis of digital terrain information commonly found in the land-use modelling domain. Fuzzy system modelling provides a mathematical environment in which vague conceptual phenomena can be rigorously studied. It exploits the tolerance for uncertainty, imprecision and partial truth of various types (in fuzzy logic) to achieve tractability, low solution costs, and a better rapport with reality, which is particularly useful for handling uncertainty in spatial data and GIS-based analysis (Shyllon 2000).

In the context of large spatial information data sets, the hope is that by using Neuro-fuzzy techniques one can perform data trawling without having to make presuppositions concerning the data distribution function, and, also extract knowledge from the data that can be used in
subsequent analysis or operations. Given the typically large sizes of these data sets, it may be advantageous to perform data filtering to reduce the size of the data set prior to carrying out computationally expensive connectionist analysis, and thus the goal of data filtering is to reduce the size of a large data space, both in terms of number of data items and number of features (so that neuro-fuzzy engineering computation can be more efficiently performed) without sacrificing the discriminating power associated with the original data. In addition, being able to identify which features are irrelevant can help to reduce the effort of data collection.

There are two areas where data reductions can be made:

- If the data elements are highly correlated, it should be possible to reduce the number of data elements under consideration. This is likely to be true with spatial data sets, where neighbouring locations are likely to contain correlated information (Toblers 1969). With neural computations this should result in a reduction of the size of the training set.
- If the individual features associated with each data element are correlated, it should be possible to consider a subset of the features and still retain sufficient discriminating power. This should lead to smaller neural network architectures and hence speedier computation.

The Chi2 statistic-based discretization algorithm offers promise in both of the above areas for spatial data preparation. It can result in a smaller number of data items, and a smaller number of features among the data items that remain.

Consider, for illustrative purposes, a version of the golf course problem altered from that described in Section 3.5 of Chapter 3. The previous solution rule set involved only a consideration of the altitude right at the given site: a high (local) altitude resulted in a contribution of low “altitude suitability” and thus a negative impact on overall site suitability. On this occasion the site suitability of a golf course will be determined solely by how ‘level’ the terrain is in the region surrounding the site, and the attributes of temperature, rainfall, and distance-to-urban-centre will be ignored. Since our data set lists the mean altitude for each 1 km² block in the New Zealand South Island, an evaluation will be made of the difference between the altitude of the block in question and that of its eight neighbouring blocks. Let $\text{alt}_{\text{diff}}$ be the
maximum difference in altitude between the given block and its eight nearest neighbours. Then
the “altitude suitability”, \textit{alt\_suitability}, can be set according to the following conditions:

\begin{align*}
\text{If } & \text{ alt\_diff } \geq 377.8 & \text{ then } & \text{ alt\_suitability } = 0 \\
\text{If } & \text{ alt\_diff } \geq 276.3 \text{ and } \text{ alt\_diff } < 377.8 & \text{ then } & \text{ alt\_suitability } = 1 \\
\text{If } & \text{ alt\_diff } \geq 188.4 \text{ and } \text{ alt\_diff } < 276.3 & \text{ then } & \text{ alt\_suitability } = 2 \\
\text{If } & \text{ alt\_diff } \geq 100.5 \text{ and } \text{ alt\_diff } < 188.4 & \text{ then } & \text{ alt\_suitability } = 3 \\
\text{If } & \text{ alt\_diff } \geq 0 \text{ and } \text{ alt\_diff } < 100.5 & \text{ then } & \text{ alt\_suitability } = 4
\end{align*}

In other words, a level region is considered likely to be good for golf course siting if it located
on level ground as specified by the eight nearest neighbour 1 km² blocks, irrespective of whether
the region is a plateau or low lying land.

Now to make this a more realistic data trawling experiment, we suppose that the data mining
analysts for this problem do not know the size of the region around a site that contributes to good
golf course selectibility and thus should be considered around each block for analysis. So they
have set the size of the region to be 25 blocks as shown in Figure 5.1.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{figure5_1.png}
\caption{Region of blocks under analysis}
\end{figure}

This 25-block region then will represent a window on the spatial data set that is located
appropriately whenever a given block (the black square in Figure 5.1) is to be considered for site
suitability. Only the grey-shaded blocks come into play for the determination of the solution set,
but the hypothetical analysts do not “know” this. Thus for each site, there are 25 inputs.
According to experiments conducted by Benwell et al. (1995) and Purvis et al. (1999), a sample size of 1000 was adequate for neural network training. Therefore, 1000 blocks were randomly chosen from a central region of the South Island. A Chi2 statistic-based discretization algorithm was applied. The inconsistency rate was set to be 0.3%, so that up to 3 (that is 1000*0.3%) inconsistent elements would be allowed before attribute merging would be halted. The results are shown in Figure 5.2.

Spatial data filtering merged 17 inputs (the shaded circles), leaving only 8 relevant inputs, which exactly identifies the correct features. With just these remaining attributes, a 3-layer artificial neural network with 8 input nodes, 10 hidden nodes and 5 output nodes was trained over the 1,000 blocks using the backpropagation algorithm. The trained network was then tested by applying it over an additional 100,000 randomly chosen blocks and comparing its classification with that of the ‘experts’. Its evaluation is shown in Table 5.1.

**Table 5.1. Test results of the neural network after the data filtering procedure**

<table>
<thead>
<tr>
<th>No difference</th>
<th>95,856 (95.86%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>One class difference</td>
<td>3,721  (3.721%)</td>
</tr>
<tr>
<td>Two class difference</td>
<td>403   (0.403%)</td>
</tr>
<tr>
<td>Three class difference</td>
<td>19    (0.019%)</td>
</tr>
<tr>
<td>Four class difference</td>
<td>1     (0.001%)</td>
</tr>
</tbody>
</table>
As can be seen from Table 5.1, 95.86% of the blocks were classified correctly. Moreover, if a misclassification by one class difference is considered tolerable, then more than 99% of the results of the neural network’s classifications are acceptable.

Now consider an expanded version of the same problem. Instead of considering only how level a site is, suppose we are also consider rainfall and mean temperature for a given block. A new “overall golf course suitability” rule set was then used, that for simplicity did not involve the distance to a major urban centre of the South Island; it was only a function of the local rainfall, the local mean temperature, and the altitude difference as described above. Again, assume that our analysts do not know the expert’s criteria and so select 25-block regions for each of altitude, rainfall, and mean temperature. With the 25-block region for each site under consideration, there are now 75 attributes to be evaluated. This would necessitate a large neural network with significant computational implications. In order to lessen the computational burden that this implies, we applied the data filtering technique of Chi2 method again.

1,500 blocks were randomly chosen from a central portion of the South Island, and associated with each block was a region like that shown in Figure 5.1. The inconsistency rate was set at 0.5%, so that up to 8 (1500*0.5%) inconsistent elements would be allowed before attribute merging would be halted. The results are shown in Figure 5.3.

![Figure 5.3](image)

*Figure 5.3.* Attributes of altitude difference, rainfall and mean temperature merged by the data filtering procedure. Merged attribute indices are shown in shaded circles
The attributes are labelled with indices from 0 to 74 and are distributed among the 25 blocks as shown in the figure. For each individual block, the altitude attribute is associated with the first (lowest numbered) index, followed by the rainfall and temperature indices. After discretization, 104 of the 1,500 original data elements were found to be redundant, so the original data set could be reduced to 1,396. 52 of the original 75 attributes were discarded, leaving 23 features necessary for classification of the 1,396 data elements to 99.5% accuracy.

The data filtering operation correctly identified the central block’s temperature attribute as essential (index 38), but also retained two outer temperature attributes. This can happen due to the correlations among the data values and the set of remaining attributes can be reduced if more data elements are considered for the training set. This problem is more evident with the rainfall attribute, which played a relatively small importance in the overall site suitability rule set. Also the variation in rainfall over the portion of the South Island considered for this experiment was relatively uniform.

For the altitude attribute, the central block’s attribute was correctly identified as irrelevant, while 5 of the 8 neighbouring blocks’ altitude attributes were retained. However 3 presumably essential blocks were discarded by the filtering procedure, and some altitude attributes for blocks outside the neighbouring region were retained. Again, the correlations in the smoothly varying data attributes enabled the data filtering procedure to classify the discretized data set to 99.5% accuracy with the 23 attributes it selected, even though not all of the attributes were the ones that we expected. They are still sufficient to achieve satisfactory classification and offer a greatly reduced data set for subsequent neuro-fuzzy analysis.

The above two case studies have demonstrated that the Chi2-based spatial data filtering can successfully reduce the number of spatial data items and spatial features. The illustrated examples might also be well-suited for studying feature construction techniques, since feature construction techniques might be able to create a set of new features that only capture the expression regarding the slope of the land, such as altitude difference between the given block and its eight nearest neighbours in the first example.
5.3 A Chi2-based Membership function selection Method for Fuzzy Systems

The first step necessary to implement a basic fuzzy system if one is using continuous real-valued data is to convert the real-valued data into a fuzzified representation using fuzzy membership functions (MFs). As described in Section 2.3.3 of Chapter 2, the membership functions are typically in practice determined subjectively by the designer, which can often be difficult and error-prone. It would be preferable to let the data determine how many and what kind of membership functions to use. This is what inspires the Chi2-based membership function selection approach.

5.3.1 Fuzzy Membership function selection based on the Chi2 Approach

The goal of the Chi2-based membership function selection approach is to choose the optimal membership functions via the Chi2 algorithm to make neuro-fuzzy computation more efficient. The Chi2-based membership function selection approach performs automatic discretization of the data, which can lead to an appropriate selection of the number and widths of trapezoidal membership functions.

The merged intervals for a given attribute (input) from the Chi2 algorithm (phase 1 or phase 2) determine the number and the widths of the membership functions. Four-point trapezoidal membership functions (see Figure 5.8) are used such that each input value belongs to a maximum of two membership functions, the membership degrees for which will always sum to one. In order to calculate the boundaries of each membership function, first the smaller interval is chosen from each pair of adjacent intervals. Then the half size (or any size smaller than the half size, e.g. quarter size) spaces of these smaller intervals are calculated. The fuzzy boundaries are obtained by setting those spaces on both sides of each interval boundary. For example, for the Iris data set described in Section 3.2 of Chapter 3, three intervals (discrete values) are needed for the attribute petal-width ($X$) at the end of phase 2:

\[
\begin{align*}
0.1 & \leq X < 1.0 \quad \rightarrow \quad 1 \\
1.0 & \leq X < 1.7 \quad \rightarrow \quad 2 \\
X & \geq 1.7 \quad \rightarrow \quad 3
\end{align*}
\]
Thus for attribute petal-width, three membership functions are needed as shown in Figure 5.4. The half size width 0.35 was calculated from the interval [1, 1.7) since it was a smaller interval compared with its neighboring interval [0.1, 1). Consequently 1.0 - 0.35 = 0.65 was set as the lower boundary for the second membership function, and 1.0 + 0.35 = 1.35 was set as the upper boundary for the first membership function.

![Figure 5.4](image)

**Figure 5.4.** Membership functions for attribute petal-width of the Iris data set created by the Chi2-based approach

### 5.3.2 Experiments

In this section, we describe the use of a fuzzy neural network model, called FuNN, to test the effectiveness of the Chi2-based membership function selection method. The detailed descriptions of the FuNN model were provided in Section 2.3.4 of Chapter 2. Three example applications were used for experiments. They are: the Pima Indians Diabetes data set, the Golf Course Problem, and the Cook Islands Sea Cucumber data set. For comparison and evaluation purposes, the fixed centre-based membership function selection method was also applied to these data sets. For the descriptions of these three data sets, and how the Pima Indians Diabetes data set and the Cook Islands Sea Cucumber data set were split for training and testing by 10-fold cross validation, the reader should refer to Chapter 3. In order to directly compare the experimental results with those demonstrated by Benwell *et al.* (1995) and Purvis *et al.* (1999), for the experiment conducted for the Golf Course Problem, 1000 randomly selected samples were used for training and the entire data set (153,036 samples) was used for testing.
Pima Indians Diabetes Data Set

The task of this problem domain is to analyse the patients who may show signs of diabetes. The detailed descriptions are provided in Section 3.3 of Chapter 3.

Fixed Centre-based Approach

Each of the 8 input variables was represented as five fuzzy values. The membership functions for the attribute *number of times pregnant* are illustrated in Figure 5.5(a). The condition elements layer of the fuzzy neural network was specified to have 40 nodes, and the middle (rule) layer was given 10 nodes. Thus the five layer fuzzy neural network had the configuration shown in Table 5.2. The BFGS algorithm was used to train the middle layers. Ten repetitions of 10-fold cross validation were used to assess the accuracy. The mean accuracy rates for training and test data, and their standard deviations are shown in Table 5.5.

### Table 5.2. Fuzzy neural network configuration based on the centre-based approach for the Pima Indians Diabetes data set

<table>
<thead>
<tr>
<th>Layer</th>
<th>Number of Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>40</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
</tr>
</tbody>
</table>

Chi2 Approach

The same 10 data partitions of cross validation as those used in the previous section were used with the Chi2 approach. For each data partition, the Chi2 algorithm was executed on the training data and generated a mapping table, based on which the training data and test data were fuzzified. A representative mapping table that includes the number of intervals and interval boundaries for the attributes at the end of Phase 2 of the Chi2 execution are presented in Table 5.3.
Table 5.3. The number of intervals and interval boundaries after Phase 2 of the Chi2 method for the Pima Indians Diabetes data set

<table>
<thead>
<tr>
<th>Attribute</th>
<th>No. of Intervals</th>
<th>Interval Boundaries</th>
</tr>
</thead>
<tbody>
<tr>
<td>A0</td>
<td>2</td>
<td>[0, 7), [7, +∞)</td>
</tr>
<tr>
<td>A1</td>
<td>5</td>
<td>[0, 44), [44, 100), [100, 128), [128, 155), [155, +∞)</td>
</tr>
<tr>
<td>A2</td>
<td>2</td>
<td>[0, 84), [84, +∞)</td>
</tr>
<tr>
<td>A3</td>
<td>3</td>
<td>[0, 8), [8, 24), [24, +∞)</td>
</tr>
<tr>
<td>A4</td>
<td>3</td>
<td>[0, 37), [37, 96), [96, +∞)</td>
</tr>
<tr>
<td>A5</td>
<td>9</td>
<td>[0, 22.9), [22.9, 28.9), [28.9, 32.9), [32.9, 33.1), [33.1, 33.3), [33.3, 45.4), [45.4, 46.1), [46.1, 48.3), [48.3, +∞)</td>
</tr>
<tr>
<td>A6</td>
<td>10</td>
<td>[0.078, 0.272), [0.272, 0.279), [0.279, 0.537), [0.537, 0.545), [0.545, 1.189), [1.189, 1.251), [1.251, 1.318), [1.318, 1.4), [1.4, 1.893), [1.893, +∞)</td>
</tr>
<tr>
<td>A7</td>
<td>4</td>
<td>[21, 23), [23, 28), [28, 63), [63, +∞)</td>
</tr>
</tbody>
</table>

According to Table 5.3, the number of membership functions for each input variable are variable; for example there are 2 membership functions for A0 (number of times pregnant) and 9 membership functions for A5 (body mass index). The membership functions for A0 (number of times pregnant) are shown in Figure 5.5(b). Based on Table 5.3, a fuzzy neural network architecture (shown in Table 5.4) with 38 nodes in the condition elements layer, 10 nodes in the rule layer and 2 nodes in the action elements layer was constructed and is shown in Table 5.4.

Table 5.4. Fuzzy neural network configuration based on the Chi2-based approach for the Pima Indians Diabetes data set

<table>
<thead>
<tr>
<th>Layer</th>
<th>Number of Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>38</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
</tr>
</tbody>
</table>
With the Chi2-based FuNN, the mean accuracy rates of ten-fold cross validation, and the standard deviations of these values are shown in Table 5.5.

**Figure 5.5.** Membership functions for the attribute *number of times pregnant* by (a) centre-based and (b) Chi2-based approach
Table 5.5. Accuracy rates (%) of the fuzzy neural networks for the Pima Indians Diabetes data set (standard deviations are in parentheses)

<table>
<thead>
<tr>
<th></th>
<th>Membership functions produced by the centre-based approach</th>
<th>Membership functions produced by the Chi2-based approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acc. on training set (%)</td>
<td>83.73(8.7)</td>
<td>88.88(5.7)</td>
</tr>
<tr>
<td>Acc. on test set (%)</td>
<td>71.99(3.3)</td>
<td>76.19(2.7)</td>
</tr>
<tr>
<td>P-value (Acc. on test set)</td>
<td>0.01</td>
<td></td>
</tr>
</tbody>
</table>

The P-value was computed for testing the null hypothesis that the means of two groups of observations on test accuracy are equal. The P-value 0.01 shows that the performance with the Chi2 approach was significantly better than that with the centre-based approach at the 99% level.

Cook Islands Sea Cucumber Habitat Data

The task of this problem domain is to identify the habitat preferences of sea cucumbers. The detailed descriptions are provided in Section 3.6 of Chapter 3.

Fixed centre-based approach

All 10 input variables were represented as three fuzzy values. The middle layers of the fuzzy neural network were constructed with 30 nodes in layer 2, 10 nodes in layer 3 and 2 nodes in layer 4. The mean accuracy rates of ten-fold cross validation, and the standard deviations of these values are shown in Table 5.7.

Chi2 approach

A representative mapping table is shown in Table 5.6, which lists the number of intervals for all attributes and the boundaries for each intervals after applying the Chi2 algorithm to the training data from one of the ten data partitions.
Table 5.6. The number of intervals and interval boundaries after Phase 2 of the Chi2 method for the Cook Islands Sea Cucumber data set

<table>
<thead>
<tr>
<th>Attribute</th>
<th>No. of Intervals</th>
<th>Interval Boundaries</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exposure</td>
<td>2</td>
<td>[0, 0], (0, 1]</td>
</tr>
<tr>
<td>%sand</td>
<td>2</td>
<td>[0, 10), [10, +∞)</td>
</tr>
<tr>
<td>%rubble</td>
<td>2</td>
<td>[0, 90), [90, +∞)</td>
</tr>
<tr>
<td>%consolidated rubble</td>
<td>2</td>
<td>[0, 2.5), (2.5, +∞)</td>
</tr>
<tr>
<td>%boulder</td>
<td>3</td>
<td>[0, 3.3), [3.3, 5), [5, +∞)</td>
</tr>
<tr>
<td>%rock/pavement</td>
<td>5</td>
<td>[0, 10), [10, 14), [14, 80), [80, 85), [85, +∞)</td>
</tr>
<tr>
<td>%live coral</td>
<td>5</td>
<td>[0, 6), [6, 10), [10, 16), [16, 20), [20, +∞)</td>
</tr>
<tr>
<td>%dead coral</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>%mud/silt</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>%gravel</td>
<td>0</td>
<td>-</td>
</tr>
</tbody>
</table>

Ten repetitions of 10-fold cross validation were performed. The mean accuracy rates for training and test set, and their standard deviations are shown in Table 5.7.

Table 5.7. Accuracy rates (%) of the fuzzy neural networks for the Cook Islands Sea Cucumber data set (standard deviations are in parentheses)

<table>
<thead>
<tr>
<th>Membership functions produced by the centre-based approach</th>
<th>Membership functions produced by the Chi2-based approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acc. on training set (%)</td>
<td>95.26(5.0)</td>
</tr>
<tr>
<td>Acc. on test set (%)</td>
<td>81.92(9.9)</td>
</tr>
<tr>
<td>P-value (Acc. on test set)</td>
<td>0.039</td>
</tr>
<tr>
<td></td>
<td>88.33(7.6)</td>
</tr>
</tbody>
</table>

It is seen that the Chi2-based FuNN performs significantly better than the centre-based FuNN. It is also observed that the Chi2-based approach can result in smaller, and therefore more efficient, neural networks, yet they still achieves superior results.

- **Golf Course Problem**
The task of this problem domain is to determine suitable sites of golf courses. The detailed descriptions are provided in Section 3.5 of Chapter 3.

**Fixed Centre-based Approach**

All input attributes: *altitude, rainfall, temperature,* and *distance* were represented as five fuzzy values each, as illustrated in Figure 5.6. Five fuzzy values: *very unsuitable, unsuitable, neutral, suitable,* and *very suitable,* were created for describing the output variable – the suitability level, as depicted in Figure 5.7. A fuzzy neural network, with 20 nodes in the condition elements layer, 20 nodes in the rule layer, and 5 nodes in the action elements layer was trained with 1000 samples of the 153,036 total data examples using the backpropagation algorithm. The fuzzy neural network was tested over the entire data set. Its evaluation is shown in Table 5.9 (a). The Fuzzy neural network was found to classify 85.6% correctly and another 14.2% of classifications were off by one membership class.
Figure 5.6. Membership functions for all the input variables of the Golf Course Problem produced by the centre-based approach
**Chi2 approach**

The number of intervals for all the attributes after applying Chi2 algorithm to 1000 training samples are shown in Table 5.8, which also lists the boundaries for each intervals.

**Table 5.8.** The number of intervals and interval boundaries after Phase 2 of the Chi2 method for the Golf Course Problem

<table>
<thead>
<tr>
<th>Attribute</th>
<th>No. of Intervals</th>
<th>Interval Boundaries</th>
</tr>
</thead>
<tbody>
<tr>
<td>Altitude</td>
<td>5</td>
<td>[1, 113), [113, 515), [515, 791), [791, 917), [917, +∞)</td>
</tr>
<tr>
<td>Rainfall</td>
<td>4</td>
<td>[500, 1125), [1125, 1958), [1958, 2792), [2792, +∞)</td>
</tr>
<tr>
<td>Temperature</td>
<td>5</td>
<td>[10.5, 11.8), [11.8, 12.2), [12.2, 12.7), [12.7, 13.9), [12.7, +∞)</td>
</tr>
<tr>
<td>Distance</td>
<td>4</td>
<td>[0, 76.6), [76.6, 101.2), [101.2, 150.5), [150.5, +∞)</td>
</tr>
</tbody>
</table>

The membership functions for all input attributes are shown in Figure 5.8. The middle layers of the fuzzy neural network were created with 18 nodes in the condition elements layer, 20 nodes in the rule layer, and 5 nodes in the action elements layer to calculate the output membership degrees.
Figure 5.8. Membership functions for all the input variables of the Golf Course Problem produced by the Chi2-based approach
The fuzzy neural network was trained with the 1,000 fuzzified samples. After the FuNN was tested again over the entire data set, 92.0% of the values were correct over the full test set and 7.9% were off by one membership class as shown in Table 5.9 (b). It can be seen that the Chi2-based FuNN had fewer total fuzzy values (18 versus 20) than the centre-based FuNN, but still had the superior performance on testing (92% correct versus 86% correct for the centre-based FuNN).

Figure 5.9 (a,b) shows the fuzzy neural networks solution for the whole data set based on the centre-based FuNN and the Chi2-based FuNN.

Table 5.9. The test result for the fuzzy neural network for the Golf Course Problem (a) membership functions produced by the centre-based approach (b) membership functions produced by the Chi2-based approach

<table>
<thead>
<tr>
<th></th>
<th>Error After Training</th>
<th></th>
<th>Error After Training</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.114</td>
<td></td>
<td>0.0798</td>
<td></td>
</tr>
<tr>
<td>No difference</td>
<td>131,060 (85.64%)</td>
<td>No difference</td>
<td>140,818 (92.02%)</td>
<td></td>
</tr>
<tr>
<td>One class difference</td>
<td>21,729 (14.20%)</td>
<td>One class difference</td>
<td>12,131 (7.927%)</td>
<td></td>
</tr>
<tr>
<td>Two classes difference</td>
<td>247 (0.16%)</td>
<td>Two classes difference</td>
<td>80 (0.052%)</td>
<td></td>
</tr>
</tbody>
</table>

(a) 1,000 samples model (85.6%) Centre-based FuNN

(b) 1,000 samples model (92.0%) Chi2-based FuNN

**Figure 5.9 (a).** Centre-based FuNN solution on the whole data set

**Figure 5.9 (b).** Chi2-based FuNN solution on the whole data set
5.3.3 Discussion

Most fuzzy set theorists have been interested in what can be done with already existing membership functions, such as using genetic algorithms for adapting membership functions, and have not been concerned with how one would construct appropriate membership functions in the first place. By far the most common method for assigning membership functions is based on direct, subjective judgements by one or more experts. In this method, an expert rates objects on a membership scale, and assigns membership values directly. However, this method has shortcomings. First, it requires an expert (experts) to give answers that are precise (or close enough) to capture subjective judgements. This is often infeasible for complex concepts. Second, investigator bias (even unconscious bias) or inconsistency can creep in when the ratings need to be made for conceptually complicated data sets. It is much harder to defend a membership rating that comes solely from expert judgement when there is little to back up the procedure besides the expert’s status as an expert. It is better to have a clear procedure in place that makes fuzzy membership function selections as transparent as possible. The Chi2-based membership function selection approach is one such approach.

For a given attribute, the Chi2 algorithm divides the range of data values into intervals (clusters). These intervals can be considered as fuzzy variables of the attribute. For example, in the experiment with the Golf Course Problem, 5 intervals were found by the Chi2 algorithm for the attribute *Altitude* as shown in Table 5.8. Then the attribute *Altitude* could be represented by the 5 fuzzy variables labelled as “very low”, “low”, “medium”, “high”, and “very high”. For a given attribute, the Chi2 algorithm also provides the boundary values of the intervals, which determines the widths of the membership functions associated with the fuzzy variables. Compared with using a genetic algorithm for adjustment of membership functions, the Chi2-based membership function selection provides an automatic selection of the number and widths of the membership functions, which appears to be more natural, more efficient and less computationally expensive.

Note that the Chi2-based membership function selection approach is only applicable for continuous attributes. If there are mixed (continuous and discrete) attributes, the Chi2-based membership function selection can be specified to operate only on the continuous attributes for membership function construction.

It is also important to emphasize that the Chi2-based membership function selection approach is
not limited to applications of fuzzy neural networks. It can be generally applied to any fuzzy systems in connection with the membership function selection.

5.4 Summary
In this chapter, we have described how the $\chi^2$ statistic-based discretization algorithm Chi2 can be combined with neuro-fuzzy engineering to perform spatial data filtering and fuzzy membership function selection. Case studies have shown that spatial data filtering can successfully reduce spatial data vertically (reducing the number of data items) and horizontally (reducing the number of features), so that neural computation can be more efficiently performed. The Chi2-based membership function selection method is a method to produce an appropriate selection of the number and widths of the membership functions automatically via discretization. By means of three experimental examinations, it has been shown that the Chi2-based membership function selection method can not only provide an automatic selection of the number and widths of the membership functions but also improve the generalization ability of the FuNN fuzzy neural network.

In the next chapter, issues concerning rule extraction and refinement techniques will be discussed.
Part III
Rule Extraction and Refinement

Part III is about issues concerning rule extraction and refinement, and it encompasses Chapters 6 – 8. In Chapter 6, an architecture and computational process for a market-based rule learning (MBRL) system are proposed, with the description focusing on differences between the proposed system and Holland’s classic classifier system. The steady-state behaviour of the proposed system is analyzed mathematically and its time complexity is evaluated. Chapter 7 focuses on how the proposed MBRL system is used as a rule-refinement tool to improve the quality of extracted rules from fuzzy neural networks and feed-forward neural networks. Chapter 8 presents the experiments and an empirical evaluation of the proposed MBRL system with six selected data sets. For comparison and evaluation purposes, experiments using four existing rule extraction techniques, namely C4.5 decision trees, the X2R rule generator, the NeuroLinear approach, and the ReFuNN approach, are also described.
Chapter 6
A Market-based Rule Learning System

6.1 Introduction
People have used markets for thousands of years to get things done. In human markets, shopping centres get built, and new products are designed, all without a global controller or overseer. Any systems that use the concept or certain features found in a market can be called “market-based systems” (Clearwater 1996). In contrast to the use of a centralized controllers, a market-based system does not need any of the agents in the system to know all the parameters of the system in order for the overall system to function smoothly, instead through the simple interactions of trading among individual agents, a global optimization can be achieved. This feature of market-based systems offers promise (Zhou & Purvis 1999) for application to rule extraction and refinement systems. By adopting the concept of economic trading behaviour among individual commercial agents, a rule discovery system can be thought of as an artificial market where individual rule agents are interacting and competing in order to achieve satisfactory behaviour of the system.

In this chapter, a ‘market trading’ technique is integrated with the techniques of rule discovery and refinement for data mining. A classifier system-inspired model, the market-based rule learning (MBRL) system is proposed. As described in Chapter 2, current classifier systems share major weaknesses: difficulties in interpretation, initial classifier chain generation, and initial system parameter setting. This chapter specifically addresses how a proposed MBRL system solves or lessens these difficulties.

The three-layer structure of the MBRL system is described in Section 6.2, with emphasis on differences in each of the layers compared with the classic classifier system. In Section 6.3, the properties of the MBRL system that distinguish it from classifier systems are outlined. A mathematical analysis of the behaviour of an MBRL system is presented in Section 6.4. The results of this analysis are applied in a practical way, being used to determine the initial system parameters for the experiments conducted in Chapter 7 and Chapter 8.
6.2 The Layers of the System

A market-based rule learning (MBRL) system is an adaptive learning system based on the market principle and which is used to modify rule sets that have been previously generated by other learning systems to improve the performance the quality of these rule sets in terms of predictive accuracy and comprehensibility.

The MBRL system takes the basic structure of a classic classifier system but introduces some changes in each of the layers. Like a classifier system, an MBRL system consists of three layers (see Figure 6.1). The first layer sees to it that the system is able to provide answers to the problems it is confronted with. This is the rule and message system. The second layer evaluates the performance of the first layer. It can also adjust that layer’s performance by using the payoff provided by the environment. This payoff is high if the behaviour of the system is good and low if the behaviour is not. It cannot, however, change the behaviour in a creative way, as it can only adjust things that are already present in the system. The algorithm used for this aspect is called apportionment of credit. The task of the third layer is to try to find new ways in which the learning system can perform its function. This is done by the genetic algorithm.

**Figure 6.1.** Model of a market-based rule learning system
6.2.1 Rule and Message System

In general, the rules derived by means of rule extraction strategies are of the following form:

\[
\text{IF } \langle \text{condition1} \rangle \& \langle \text{condition2} \rangle \& \ldots \& \langle \text{conditionN} \rangle \\
\text{Then } \langle \text{action} \rangle 
\]

Classifier systems have traditionally used binary strings to represent rules and inputs. In contrast, real-valued representations are adopted by the MBRL system.

As described in Chapter 2, the binary string representations are partially responsible for the difficulty of setting initial system parameters. Many binary string related parameters, such as the word length of the messages, the word length for each condition and action, and the probability of selecting a wildcard (\#) in a randomly generated population, must be set. If real-valued representations are used, this parameter setting can be omitted. In addition some other parameters, such as the number of conditions in the antecedent, can be set in a straightforward fashion.

The binary-string rule representations also provide a barrier for inspecting rules transparently. In order to exhibit the learning results on the application of the Wisconsin Breast Cancer Database (UCI 1989), Wilson (2000) had to convert the rules with binary string representations into real-valued representations. The MBRL system’s real-valued representation has an advantage in terms of the interpretation of learned rules and avoids the additional overhead of converting among different rule representations that are often involved with classifier systems.

Furthermore, continuous real-valued variables, such as temperature or age, are typical in real world problems. It is more natural to use real-valued representations for these applications.

6.2.2 Apportionment of Credit Algorithm

A classic classifier system employs the bucket brigade algorithm to modify strength of classifiers. Strength modifications occur via three interrelated mechanisms:
Auction
Reinforcement & punishment
Taxation

In the MBRL system, an algorithm called the *market-based trading algorithm*, is used for modifying the strengths of rules. Strength modification occurs via two mechanisms:

Reinforcement & punishment
Taxation

In the MBRL system when rules are matched against environmental messages or actions of other rules, they do not participate in an activation auction. Instead, all matched rules are allowed to perform their actions. The motivation for elimination of the auction mechanism comes from the MBRL system’s principal role as a rule refinement tool. It is expected that the MBRL system only deals with existing rule sets extracted using other learning techniques, such as ReFuNN and NeuroLinear techniques, and thus the number of initial rules in the MBRL is not large (say, less than 100). Under this assumption, the elimination of the auction mechanism gives each rule an equal opportunity to perform and affect the outside world.

Imagine a message from the external environment flowing to the system as the action of a business startup that is offering shares on the stock market. The individual rules in the rule base can be thought of as stock trader agents that may make an investment (buy shares) in the business in order to gain a profit. If the number of trader agents that are interested in purchasing some shares of the business is not large, these trader agents should be considered to have an equal opportunity to make an investment without unnecessary competition in an auction. So when a rule is activated, it can be imagined that a rule agent has made an investment to purchase some shares of the business: its investment serves as a payment to the business shares provider, which is the incoming environmental message in this case. In the meantime, the activated rule posts its action on the message list. If its action triggers other rules, we can imagine that the shares are sold to other agents, and the receipt is collected and transformed from these buyer agents. However, if its action does not trigger any other rules, we can consider this as due to no trader agents being interested in making investments on it, and no payment is received. If a rule
sequence produces a ‘good’ (i.e. that which is desired) final output, it can be imagined that the successful chains of trader agents led to the ultimate buyers – a reward is received from the environment. However, if a rule sequence produces a ‘bad’ final output, we can think that the unsuccessful chains of trader agents have not led to the ultimate buyers – no reward is given by the environment. Apart from Wilson’s XCS (1995) model, which is a single-layer classifier system that does not need to use the auction and the bucket brigade algorithm, the author has not seen any other systems in the field that deliberately omit the auction mechanism.

As in classifier systems, each rule in the MBRL system maintains a record of its net worth, called strength (wealth). When a rule matches an environmental message or an action of an other rule, it becomes active. It then invests some of its wealth (proportional to its strength) called payment. Every matched rule has to pay, so every matched rules gets its strength decreased by the amount of its payment. The payment of rule $i$ at iteration $t$, $P_i(t)$, is calculated as:

$$P_i(t) = C_{\text{payment}} \cdot S_i(t)$$

(6.1)

where,

- $C_{\text{payment}}$ is a rule payment coefficient that determines what proportion of a rule’s strength will be lost on a single step.
- $S_i(t)$ is the strength of rule $i$ at step $t$.

By comparing the above equation with equation (2.18) in Chapter 2, it can be seen that the setting of two system parameters ($bid_1$ and $bid_2$), and the calculation of specificity are omitted in the MBRL system.

In the market-based trading algorithm, the activated rules get their strengths decreased by the amount of payment and the divided amounts paid to the contributing rules get added to their strengths. Figure 6.2 shows an example of how the trading algorithm works. Rule C distributes its payment to Rules A and B, which are the parties responsible for its activation. Usually the payment is distributed equally among the contributing rules. In a subsequent time step, activated Rule D makes its payment to the previously active Rule C. Finally, a reward comes into the system and is paid to the last active rule, Rule D, if it produces a good (rewarded) final output.
As in classifier systems, there are also two types of taxes in the MBRL system: *life tax* and *activation tax*. The life tax is applied to every rule on every iteration, just as in the classifier system. Unlike the bid tax in the classifier system, though, the activation tax in the MBRL system is designed to penalize those rules that are active frequently but are not able to activate other rules.

In contrast to the complete strength equation (2.20) used by the bucket brigade algorithm, the complete strength equation used in connection with the market-based trading algorithm is

\[
S_i(t + 1) = S_i(t) - t_{\text{life}} \cdot S_i(t) - P_i(t) + R_i(t) - t_{\text{activation}} \cdot S_i(t)
\]

(6.2)

where,

- \(S_i(t)\) is the strength of rule \(i\) at the beginning of iteration \(t\).
- \(S_i(t+1)\) is the strength of rule \(i\) at the end of iteration \(t\).
- \(P_i(t)\) is the \(i\)-th rule’s payment during iteration \(t\) (as defined by equation 6.1). It is only paid if rule \(i\) matches with an environment message or an action of another rule.
- \(R_i(t)\) is the reward given by the activated classifiers or that from the environment. It is only be non-zero if rule \(i\) was active and able to activate other rules or affect the environment on iteration \(t-1\).
- \(t_{\text{activation}} \cdot S_i(t)\) is the rule activation tax during iteration \(t\). It is only paid if rule \(i\) matches with an environment message or an action of other rules.
- \(t_{\text{life}}\) is the rule life tax coefficient.
$t_{activation}$ is the rule activation tax coefficient

The rewarding (paying) of rules for being activated is designed to reinforce chains of good rules. Chains of rules will be necessary if the function that is computed by the learning system is complex (that is, when a good result cannot be achieved by a single master rule). So if the answers given by the chain of rules are usually good, the last rule (the one producing the output) will get rewarded, and its strength will increase. The rules that help it to become activated will then get payments, so their strengths will increase as well. This way the good rewards flow down the chain. Bad chains of rules will get no reward (they may even get a negative reward, which can be called a punishment). Rules that activate the rule that produces the bad output will also get very small payments, so over time their strength will decrease, and the strength of all the rules in the chain will, too. Through such a trading system, the rule base can be refined by the genetic algorithm which means that rules with high strength will survive, while rules with low strength will die off.

The system may not be able to respond properly to those inputs for which no chain of rules can be established. In order for the system to be able to respond to such input, new rules must be created, and that is where the genetic algorithm comes in.

6.2.3 The Genetic Algorithm

In Holland’s classifier system, rules are represented by bit strings consisting of zeroes, ones, and wildcards, so the GA operates on bit strings. However, in order to represent rules more naturally, each gene in the MBRL system is an integer or floating-point number rather than a single bit. Each chromosome is thus a list of real values. Note that the real-valued encoding methods for specific problems will be described in Section 7.2.1 and Section 7.3.1 of Chapter 7.

The basic genetic algorithm actions involved in the MBRL system are:

- Selection
- Crossover
- Mutation
- Scanning
- Crowding
The placement of these actions in the overall genetic algorithm is shown in Figure 6.3.

In Figure 6.3 there is a box that reads ‘Determine strengths for all population members’; however in an MBRL system this determination does not occur in a single iteration. Instead, an MBRL system determines the strengths and thus the ranking among the population members over multiple iterations with the environment, during which strength changes occur by means of the market-based trading algorithm. Only after multiple interactions with the environment will the rule strengths represent a measure of how well the rule performs in the environment.
After the system parameters have been set, the initial population has been generated, and the strengths of all population members have been determined, the following genetic algorithm actions are performed:

- **Selection**
  Unlike roulette-wheel selection as used in Holland’s classifier system, rank-based selection is employed in the MBRL system. In the rank-based selection, the individuals in the population are ranked according to fitness, and the probability of each individual being selected to reproduce depends on its rank rather than on its absolute fitness. There is no need to scale fitnesses in this case, since absolute differences in fitness are ignored.

- **Crossover**
  After a pair of parent chromosomes are selected from the current population, crossover is performed with crossover probability $P_c$.

For chromosomes formed by real-valued numbers, a variety of real-number crossover operators can be employed (Beasley et al. 1993). Montana and Davis’s (1989) crossover method is used here. The offspring chromosomes are created as follows: each gene in the offspring is created by copying the corresponding gene from one or other parent at random. This is illustrated in Figure 6.4.

![Figure 6.4. Illustration of Montana and Davis’s crossover method](image-url)
Mutation

The two offspring are mutated with mutation probability \( P_m \).

For floating-point chromosomes used here, Davis’s (1991) creep mutation method is employed. The mutation operator randomly selects \( n \) genes, and for each selected gene, adds a randomly generated value to it. This randomly generated value is based on the distribution of the gene values where the mutation is being applied. Suppose \( C = (c_1, ..., c_n) \) is a chromosome and \( c_i \in [a_i, b_i] \) is a gene to be mutated, where \( a_i \) is a minimum value of variable \( i \), and \( b_i \) is a maximum value of variable \( i \). The gene, \( c_i' \), is produced by applying the equation:

\[
C_i' = C_i + \frac{\delta}{1 + e^{-|G|}}
\]

where,

\( \delta \) is the maximum possible change in \( c_i \). It can be calculated by taking the higher absolute value of \( |(a_i - c_i)| \) and \( |(b_i - c_i)| \). \( \delta \) is positive if \( |(b_i - c_i)| \geq |(a_i - c_i)| \), and \( \delta \) is negative if \( |(b_i - c_i)| < |(a_i - c_i)| \).

\( G \) is a randomly generated standard Gaussian variable. The random Gaussian variable \( G \) is generated independently for each variable \( i \).

An example of a random value added to selected genes is shown in Figure 6.5.

<table>
<thead>
<tr>
<th>Before mutation</th>
<th>0.3</th>
<th>-0.4</th>
<th>0.2</th>
<th>0.8</th>
<th>-0.3</th>
<th>-0.1</th>
<th>0.7</th>
<th>-0.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>After mutation</td>
<td>0.3</td>
<td>-0.4</td>
<td>0.2</td>
<td>0.6</td>
<td>-0.3</td>
<td>-0.9</td>
<td>0.7</td>
<td>-0.1</td>
</tr>
</tbody>
</table>

**Figure 6.5.** Illustration of Davis’s mutation method.

For integer chromosomes, the mutation operator is performed by randomly selecting \( n \) genes, and for each selected gene, replacing it by a random integer selected from a given range, e.g. from 1 to 9.
Whenever a new rule is generated by the GA, the population is scanned to see if there already exists a rule with the same condition and action. If so, the new rule is discarded. Otherwise, the new rule is ready to insert into the population by using the replacement method. The scanning operation ensures that the resulting population consists entirely of structurally unique rules.

- Crowding

Following Holland’s classifier system, De Jong’s (1975) crowding technique is used as a replacement method in the MBRL system.

The Euclidean distance is used here as a similarity measure for comparing real-valued representation rules. For the N-dimensional data points $x$ and $y$, the Euclidean distance $D_E$ is defined as:

$$D_E = \sqrt{\sum_{i=1}^{N} (x_i - y_i)^2}$$  \hspace{1cm} (6.4)

Although the Euclidean distance is the default distance measure in most computer packages, it is only one of many possible ways of measuring the distance or similarity between two individual vectors: Legendre and Legendre (1983) have discussed 35 different similarity measures. The reason for selecting the Euclidean distance here is because it is an appropriate measure for the type of data (rules) being compared in the MBRL system. Sneath and Sokal (1973) have shown that the Euclidean distance has a number of characteristics that can make it dangerous to the unwary:

- It is scale dependent: any change in the unit of one of the variables could completely change the pattern of distances.
- Because the differences are squared, it is very sensitive to outliers, or to variables where the size of the differences depends on their average value.

The rules (ReFuNN rules and NeuroLinear rules) handled by the MBRL system do not contain variables with units. For example, the elements of NeuroLinear rules are the coefficients associated with the input variables, the discrete intermediate output values, and final output
values. Therefore the Euclidean distance’s scale dependency problem does not apply to this particular case.

For ReFuNN rules and NeuroLinear rules, the coefficients associated with input and output variables are drawn from the weights of connections in the associated neural networks. When the penalty term (specified in Chapter 4) is added to the error function during network training, the weights of the connections are prevented from getting too large. As a result, the value difference between a single weight and all other weights is relatively small. Therefore, the Euclidean distance’s sensitivity to outliers is not so critical in MBRL systems.

In summary, the GA in the MBRL system works as follows. The system is cycled many times, with the trading algorithm determining the fitness of the rules. When the strengths of the rules have stabilized, the genetic algorithm is invoked. Individual rules will be selected, crossed over and/or mutated, and scanning and crowding are employed to arrive at a new population. The new population is then used by the system. The system should not forget completely what it has learnt between two steps of the genetic algorithm. Therefore the probability of crossover and mutation must be chosen to be relatively low, so only a small part of the rule set should be supplanted every generation. The set of rules will then change only slowly under the action of the genetic algorithm.

By adding the genetic algorithm as a third layer on top of the basic rule and message system and market-based trading algorithm, the population of rules evolves over time, continually exploring new regions of the space of possibilities. That causes the system not only to learn from experience but also able to create new rules.

### 6.3 The Properties of the System

As a classifier system-inspired model, an MBRL system has the following properties when compared with classifier systems:

- An MBRL system is a learning system that aims at evolving and refining existing rule sets in order to lead to improved performance by increasing the accuracy of the rule inference and/or improving the comprehensibility of the rule set. It is designed to refine the existing rule sets,
hence gain a better understanding of the data and problem domains. The experiments presented in Chapter 7 and Chapter 8 of applying the MBRL system to the rule sets extracted by the ReFuNN and NeuroLinear techniques demonstrate that the MBRL system has the capability to improve the quality of rule sets in terms of their predictive accuracy and comprehensibility. In contrast, a classifier system is a machine learning system that learns syntactically simple string rules (called classifiers) to guide its performance in an arbitrary environment. In some respects, a classifier system is similar to a control system. In as much as a control system (Dorf 1983) uses feedback to “control” or “adapt” its output for an environment, a classifier system uses feedback to “teach” or “adapt” its classifiers in order to lead to a satisfactory output. It is often the case that a classifier system is applied in a given domain for the purpose of constructing a system that can perform a control or planning task. Thus in the past years most of the successful classifier system applications have been in the areas of direct control and planning. Only Holmes (1996), Saxon and Barry (2000), and Wilson (2000) have actually investigated the interpretation of learned classifiers (rules) in classifier systems.

- Although an MBRL system keeps the main structure of Holland’s classifier system model, it introduces the fundamental change of importing existing rule sets generated by other rule development techniques to the system. This not only makes the MBRL system begin with pre-established rule sets (rather than a random set) but also enhances the likelihood of being able to interpret the evolved rules.

In the following chapters, the MBRL system is also investigated as a independent learning model to accomplish computational tasks when no prior knowledge is given (working on randomly selected rule sets). The experimental results in Chapter 7 and Chapter 8 demonstrate that by employing appropriate rule presentations and formats, e.g. floating-point rule representations and the format of NeuroLinear rules, the MBRL system is capable of independently generating accurate and comprehensible rules that allow domain experts to examine what has been learned.

- In the field of classifier systems, only a few applications (de Boer 1994; Holmes 1996; Saxon & Barry 2000; Wilson 2000) have been reported as employing these systems in the areas of
machine learning and knowledge discovery. All of these applications employed single-layer classifier systems (which means that classifiers only match messages from the environment and immediately generate actions that modify the environment). In other words, except for allocating payoffs directly to the classifiers that produce results, the bucket brigade algorithm as defined by Holland (1986) did not play a role in these reported systems. From the system applications point of view, the MBRL system’s evolution of two levels of rule sets initially extracted from feed-forward neural networks is the first successful application that demonstrates how the Apportionment of Credit Sub-system (in our case, the market-based trading algorithm) can be applied to multiple levels of rule sets for the purpose of knowledge discovery for real world problems. Compared with single-layer classifier systems, the MBRL system evolving two levels of rule sets involves more complex interactions and operations comprising message posting and matching, rule activation, payment clearing, reward gathering, and rule strength updating.

- With respect to the chain generation difficulty that exists in the bucket brigade algorithm, the MBRL system counters the problem by adopting multiple-level rule bases generated by other learning algorithms. In this thesis, the MBRL system has been tested on the basis of two-level NeuroLinear-type rules. But applying the MBRL system to the multiple level rule bases that can be extracted by the rule extraction technique proposed by (Setiono 1996) may be a fruitful area for additional experimental investigation.

- As described earlier, the real-valued rule representations in the MBRL system lessen the problem of setting initial system parameters found in the classifier systems. In the MBRL system, the setting of binary string related parameters is omitted, and some parameters such as the number of conditions in the antecedent can be easily set. However, the appropriate values of other parameters such as the payment coefficient \( C_{\text{payment}} \), the life tax coefficient \( t_{\text{life}} \), and activation coefficient \( t_{\text{activation}} \) still need to be further investigated. According to Goldberg (1989a), “selection of classifier system parameters remains something of an art form; however, useful design guides may be obtained by calculating expected steady-state performance.” Therefore, a mathematical analysis of the MBRL system in the steady state is given in the next section.
Scalability is not an issue with the MBRL. The MBRL system is primarily designed as a post-processing tool that refines rule sets that have been previously generated by other rule development techniques. Thus the scalability of such a post-processing tool does not play a crucial role for knowledge discovery and data mining applications. Scalability is more of a challenge for pre-processing and discovery techniques that may be directly applied to very large data sets.

6.4 Analysis of the Behaviour of the System

In order to gain a deeper understanding of the operation of an MBRL system, it is necessary to analyse it mathematically. This analysis can be conducted from different points of view. We can investigate what the best values of the system parameters are. We can also try to find out what the behaviour of the system in the long run is, and how long will it take for the system to run?

Goldberg (1989a) analysed the steady-state behaviour of a classifier system with the bucket brigade algorithm in. However no conclusive results were given on what are the best system parameters, and no detailed analysis was provided in terms of the relationships among these parameters. Since the market-based trading algorithm differs from the bucket brigade, we here to investigate the steady-state behaviour of an MBRL system and the relationships among different system parameters.

6.4.1 Steady-state Behaviour

The MBRL system is a complex system in which many interactions take place. These interactions consist of the activations of rules, the clearing of payments, receiving rewards, and updating strengths. Although this process is difficult to analyse completely and mathematically, it is possible to investigate the long-term behaviour of the system and especially to look carefully at its steady-state behaviour. The steady state is an abstract state in which all the rules have a fixed strength and a fixed payment. This is obviously something which will not happen quickly in any practical learning system, since there will always be conflicting rules and random variations in the input that cause perturbations. However, analysing the system’s steady-state behaviour will help to estimate certain things about the expected behaviour of rules and determine the values of the parameters of the system.
We therefore obtain the steady-state strength $S_{ss}$ by setting $S(t+1) = S(t) = S_{ss}$, and the steady-state payment $P_{ss}$ by setting $P(t+1) = P(t) = P_{ss}$. It is assumed that in the steady state of the system there are no conflicting rules, and a rule always gets the constant receipt $R_{ss}$ for every action it generates. These assumptions are not unreasonable if we consider a learning system after it has been trained fully.

- **Market-based Rule Learning System without Life tax**

First we analyse the system without the life tax. This means that the strength of non-active rules is never reduced. A possibility that individuals that do nothing will dominate the population in genetic search arises in this case. The steady-state strength $S_{ss}$ is given by:

\[
S_{ss} = S_{ss} - P_{ss} - t_{activation} \cdot S_{ss} + R_{ss}
\]  

(6.5)

where $S_{ss}$, $P_{ss}$ and $R_{ss}$ are described as above, and $t_{activation}$ is a tax coefficient for the activation tax which must be paid by activated rules. The steady-state payment $P_{ss}$ is given by

\[
P_{ss} = c_{payment} \cdot S_{ss}
\]  

(6.6)

where $c_{payment}$ is the payment coefficient. A rule pays in proportion to its strength.

Combining equations (6.5) and (6.6), $S_{ss}$ is calculated by

\[
S_{ss} = \frac{R_{ss}}{c_{payment} + t_{activation}}
\]  

(6.7)

Here the strength is simply the receipt amplified by the gain coefficient $1/(c_{payment} + t_{activation})$.

Combining equations (6.6) and (6.7), the steady payment is given by the following equation:

\[
P_{ss} = \frac{c_{payment} \cdot R_{ss}}{c_{payment} + t_{activation}}
\]  

(6.8)

Since $t_{activation}$ is usually small with respect to the payment coefficient $c_{payment}$, the steady payment value usually approaches the steady receipt value, i.e. $P_{ss} = R_{ss}$. It can also be seen that for bad rules that do not get any reward at all, both the strength and the payment will go to zero. When the fixed receipt $R_{ss}$ equals 1, Figure 6.6 shows the steady-state payment $P_{ss}$ for various values of $c_{payment}$ and $t_{activation}$. From this figure it can be seen that the steady-state payment $P_{ss}$ is sensitive
to the activation-tax coefficient $t_{activation}$. $P_{ss}$ decreases sharply when $t_{activation}$ increases a small amount within the range of 0 to 0.2. Figure 6.7 shows how the strength $S_{ss}$ changes for different values of $c_{payment}$ and $t_{activation}$ in the steady-state. When the value of $c_{payment}$ lies between 0 and 0.2, while $t_{activation}$ is in the range of 0 to 0.06, the steady-state strength makes significant changes. But when both constants are greater, the strength tends to be stable. Although Figures 6.6 and 6.7 only show what the values of steady-state payment $P_{ss}$ and steady-state strength $S_{ss}$ would be for specific values of $c_{payment}$ and $t_{activation}$, they reveal possible selection ranges for these two system parameters.

**Figure 6.6.** Values of the payments for different payment coefficients and activation taxes in the steady state ($R_{ss}=1$)

**Figure 6.7.** Values of the strength for different payment coefficients and activation taxes in the steady state ($R_{ss}=1$)
In the case where a life tax exists, the equations describing the steady-state of the system will be different from above. This is due to the fact that when the rule is not activated, its strength will still decrease. In the steady state without life tax, we could ignore the cycles in which the rule does not become active, since nothing happens. In the case where we have a life tax, we have to take into account the number of cycles a rule is inactive between activations.

We assume that the number of inactive cycles between two activations is $E$, so the steady-state strength $S_{ss}$ with the existence of a life tax is given by:

$$S_{ss} = (1 - t_{life})^E \cdot (S_{ss} - P_{ss} - t_{activation} \cdot S_{ss}) + R_{ss}$$  \hspace{1cm} (6.9)

where $t_{life}$ is a tax coefficient for life tax, and $S_{ss}, P_{ss}, R_{ss}$ and $t_{activation}$ are the same as in previous descriptions.

By using equations (6.9) and (6.6), the steady-state strength $S_{ss}$ with life tax is given by:

$$S_{ss} = \frac{R_{ss}}{1 - ((1 - t_{life})^E \cdot (1 - c_{payment} - t_{activation}))}$$  \hspace{1cm} (6.10)

By using equation (6.10) and equation (6.6), the steady-state payment $P_{ss}$ with life tax is written as follows:

$$P_{ss} = \frac{c_{payment} \cdot R_{ss}}{1 - ((1 - t_{life})^E \cdot (1 - c_{payment} - t_{activation}))}$$  \hspace{1cm} (6.11)

It is clear that these two equations are exactly same as equations (6.7) and (6.8) for the case when $t_{life}$ is zero. In order to see the influence of the life tax coefficient $t_{life}$ on the strength in the steady state, Figure 6.8 shows the steady-state strength $S_{ss}$ for various values of $t_{life}$ and various values of inactive cycles $E$. The strengths were calculated by setting $c_{payment}$ as 0.1, $t_{activation}$ as 0.01 and the fixed receipt $R_{ss}$ as 1. From this figure we can see that we must be very careful in choosing $t_{life}$. This is because if this constant is too high, then in a short period of time the strength will make very little change while the number of inactive steps $E$ increases (which is not the desired situation as more active rules should have higher strength than less active rules over longer periods of time). This can be seen more clearly from Figure 6.9. When the life-tax coefficient $t_{life}$ has the value 0.002, the strength continuously decreases when the number of inactive cycles is
increased to 50. But if the life tax coefficient $t_{life}$ was set to 0.04, the strength $S_{ss}$ changes very little after the number of cycles of inactivity is greater than 30. In the experiments described in Chapter 8, the system in some cases is cycled up to 2000 times in order to stabilize the strength, so the parameter $t_{life}$ must be set to a very small value, such as 0.00001.

Figure 6.8. Strengths in the steady state for different values of life tax and number of inactive steps ($R_{ss}=1$)

Figure 6.9. Strengths in the steady state when the life tax equals 0.002 and 0.04

According to Goldberg’s analysis (1989a), a classifier system will remain stable (even with the switching nonlinearity introduced by activating and deactivating rules) when $0 \leq K \leq 2$, where $K = c_{bid} + t_{bid} + t_{life}$, $c_{bid}$ is the bid coefficient, $t_{bid}$ is the bid tax coefficient, and $t_{life}$ is the life tax coefficient. Although the MBRL system deliberately omits the auction mechanism from the
bucket brigade algorithm, a situation in which all matched rules become active in the MBRL system can be considered to be an extreme case of an auction in the bucket brigade, which is when all matched rules win in the auction. So Goldberg’s assumption still applies to the MBRL system in the following way: \( K = c_{payment} + t_{activation} + t_{life} \). However, in practice we hold \( K \leq 1 \) to enforce nonnegativity of the rule strength in the MBRL system.

### 6.4.2 Starting Parameters

As we have seen in the previous section, the parameters of the system can have a significant influence on performance. One can use the analysis just presented as a guide for estimating the starting values of some system parameters such as the life tax coefficient \( t_{life} \). Table 6.1 shows the values of the most important parameters of the system that were used in the experiments which will be described in Chapter 7 and Chapter 8.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>initial strength</td>
<td>1</td>
<td>Initial strength of a rule after initialization</td>
</tr>
<tr>
<td>payment coefficient</td>
<td>0.1</td>
<td>Percentage of the strength of a rule that is used as a payment to the rules that helped in activating it.</td>
</tr>
<tr>
<td>activation tax coefficient</td>
<td>0.01</td>
<td>Percentage of the strength that is used as a tax imposed on an active rule</td>
</tr>
<tr>
<td>life tax coefficient</td>
<td>0.00001</td>
<td>Percentage of the strength that is used as a tax imposed on every rule at every cycle of the system</td>
</tr>
<tr>
<td>good reward</td>
<td>1</td>
<td>Reward for a good answer</td>
</tr>
</tbody>
</table>

For the setting of various parameters associated with the genetic algorithm, such as population size, crossover probability, and mutation probability, as discussed earlier in Chapter 2, there are no definitive specifications concerning what is best. The MBRL system follows the parameter setting guidelines given by De Jong (1975), Grefenstette (1986), and Schaffer et al (1989), which are: crossover probability 0.7-0.95, mutation probability 0.001-0.01, and a small population size 20-50.
6.4.3 Time Complexity
The time it takes for an MBRL system to learn a given problem is another unknown factor. The number of cycles needed to reach a state in which no major changes in performance take place in the system depends on the problem domain. In the experiments of Chapter 8, we will see that approximately 50 - 2000 cycles are needed.

The time complexity of a single cycle of an MBRL system (without genetic algorithm) in a computer program is a straightforward calculation. In a typical implementation every message in the message list is compared to every condition of every rule. After that the activated rules are paid and the actions of the activated rules get placed in the new message list. Suppose $M$ is the number of messages in the message list, $N$ is the number of rules in the rule base, $k$ is the number of conditions per rule, and $l$ is the number of elements of a message. The comparison between messages in the message list and rules in the rule base costs $O(M \cdot N \cdot k \cdot l)$. The time taken for the paying step is $O(N)$ and the time for placing of messages is $O(M \cdot l)$. It is evident that a system with a larger size of the rule base will run more slowly than one with a smaller size.

6.5 Summary
This chapter has proposed a three-layer market-based rule learning (MBRL) system which consists of the rule and message system, the market-based trading algorithm, and a genetic algorithm. As a classifier system-inspired model, although it takes the main structure of a classifier system, it introduces the fundamental change of importing existing rule sets generated by other rule extraction techniques to the system. Moreover, it presents various modifications in each of the layers of the structure. With the change and modifications introduced by the MBRL system, the problems existing in current classifier systems can be solved or lessened.

The guidelines for choosing the starting values of some important system parameters have been derived by mathematically analysing the MBRL system’s steady-state behaviour. The performance of the proposed MBRL system has been demonstrated by experiments that are reported in Chapter 7 and Chapter 8.
In the next chapter, issues concerning how the MBRL system is used to evolve and refine the extracted rules from fuzzy neural networks and feed-forward neural networks will be given in detail.
Chapter 7
Rule Evolution and Refinement Using the Market-based Rule Learning System

7.1 Introduction
In this chapter, the market-based rule learning (MBRL) system is combined with rule discovery techniques to generate rules with higher accuracy and less complexity. Section 7.2 describes how the MBRL system evolves and refines fuzzy rules extracted from fuzzy neural networks. Section 7.3 provides a detailed description of combining the MBRL system with extracted rules from feed-forward neural networks. Section 7.4 summarizes the whole chapter.

7.2 Market-based Rule Evolution and Refinement Based on Extracted Rules from Fuzzy Neural Networks
The general model of market-based rule evolution and refinement based on fuzzy rules is shown in Figure 7.1. There are two basic steps:
1) discovery process: A fuzzy neural network (FuNN) is trained and the ReFuNN approach is chosen to extract fuzzy rules.
2) post processing: fuzzy rules are sifted and refined by the MBRL system.

![Figure 7.1. The general model of market-based rule evolution based on fuzzy rules](image)

For the general descriptions of the FuNN network and the ReFuNN approach, see Sections 2.3.4 and 2.5.5 of Chapter 2. Here, we focus on the post-processing operation: how the MBRL system selects and adjusts existing fuzzy rules, and creates new fuzzy rules for a given problem.

It should be pointed out that the accuracy of the FuNN fuzzy rules extracted by the ReFuNN
technique is significantly worse than that of the original fuzzy neural network (FuNN) and the performance of the original feed-forward neural network. For example, for the Iris classification data with 10-fold cross validation, the fuzzy neural network FuNN using Chi2-based membership functions achieved a mean 96.17% ±0.036 success rate on the testing samples, and the three-layer feed-forward neural network using the PBS pruning scheme made correct predictions for 94.67%±0.061 of the testing samples (see details in Section 4.2.3). But the fuzzy rules extracted from the trained FuNN by the ReFuNN technique had only a mean success rate of 76.14%±0.121 on the testing samples, which was 20.03% and 18.53% worse than the performance of the FuNN and the feed-forward neural network at the 99% level, respectively. Thus it can be valuable to use a rule-refinement tool such as the proposed MBRL system to improve the quality of the ReFuNN-generated rules extracted from FuNN fuzzy neural networks.

7.2.1 Fuzzy Rule and Input Information Encoding

First we consider how to convert a particular format of fuzzy rules and environmental information into appropriate rule representations and internal messages to make them applicable to the MBRL system. In general, many search and learning applications use fixed-length, fixed-order bit strings to encode candidate solutions. Typically, they are strings consisting of zeroes, ones and wildcards. Here, however, a real-value encoding scheme is adopted to represent weighted fuzzy rules and fuzzy membership values.

For a given FuNN network with \( x_1, \ldots, x_n \) representing inputs for a given problem, then the ReFuNN technique can extract rules with the following form:

\[
R_k: \text{If } x_1 \text{ is } A_1^k \text{ with } d_1 \text{ and } \ldots \text{ and } x_n \text{ is } A_n^k \text{ with } d_n \text{ then } Y \text{ is } C_1 \text{ with } r_1^k \text{ and } \ldots \text{ Y is } C_j \text{ with } r_j^k
\]

where \( A_1^k, \ldots, A_n^k \) are fuzzy labels (values) associated with each of the input variables, \( d_1, \ldots, d_n \) are numerical coefficients of importance attached to each of the fuzzy input variables, which are the weights of the connections between the rule node \( k \) and the condition elements nodes in the FuNN, \( C_1, \ldots, C_j \) are fuzzy labels (values) associated with output variable \( Y \), and \( r_1^k, \ldots, r_j^k \) are the certainty degrees attached to each fuzzy output variable, which are the weights of the connections between the rule node \( k \) and the action element nodes in the FuNN. If “is not” appears in the rule,
then it indicates that the numerical coefficient of importance $d$ or the certainty degree $C$ is a negative value. In a FuNN network, each node in the rule layer represents a single fuzzy rule like $R_k$. That is why the middle layer of a FuNN is named as a “rule layer”, which was described in Section 2.3.3 of Chapter 2.

The encoding of a fuzzy rule into the MRBL system is as follows. Each fuzzy rule is a list of real-valued coefficients of importance associated with input fuzzy variables and certainty degrees associated with output fuzzy variables. The values are read off the fuzzy rule and placed in a list. For those fuzzy variables which do not appear in the fuzzy rule, a value of 0 is given in the corresponding locations.

Suppose for the Golf Course Problem, five fuzzy values (from very unsuitable to very suitable), are created for describing the output (decision) variable. All the input variables: altitude, rainfall, temperature and distance, are represented as five fuzzy values each. We use A, B, C, D, and E to represent both input and output fuzzy values, with A being the lowest value and E being the highest value. A fuzzy rule is shown below:

\[
\text{If} \quad \text{<ALTITUDE is not A 16.5>} \quad \text{and} \quad \text{<RAINFALL is not C 2.2>}
\text{and} \quad \text{<RAINFALL is E 2.42>} \quad \text{and} \quad \text{<TEMPERATURE is A 12.6>}
\text{and} \quad \text{<TEMPERATURE is B 5.1>} \quad \text{and} \quad \text{<DISTANCE is not C 8.6>}
\text{then} \quad \text{<SUITABILITY is A 13.5>} \quad \text{and} \quad \text{<SUITABILITY is not B 20>}
\]

Then, it is converted to a list shown in Figure 7.2:

![Figure 7.2](image)

Figur 7.2. Illustration of converting a fuzzy rule into a list that serves as a rule representation in the market-based rule learning system

Here, since the linguistic labels B, C, D, E of the attribute “Altitude” are not used in the antecedent part of the rule, zeroes are placed in their corresponding locations in the list.
In order to feed input information into the MBRL system, the input information must be decoded into message form and then delivered to the system. Again a real-valued encoding scheme is employed. Fuzzification is done before the encoding process begins. The task of fuzzification is to take crisp input values and determine the degree to which they belong to each of the appropriate fuzzy sets via membership functions. The encoding process takes the membership function degrees of the input variables and places them on a list. The expected (targeted) solution for the given input data is the last element in the list. This solution value will be used to compare the system output to evaluate the performance of the rules.

This is illustrated in the following example. Suppose for the Golf Course Problem, a particular land block is considered to be an excellent site for a golf course; thus 4 is the expected output class. The membership function values of the input variables to which these data belong is found to be:

\[
\begin{align*}
\mu_A: & \text{very low (Altitude)} = 0.3 \\
\mu_B: & \text{low (Altitude)} = 0.7 \\
\mu_C: & \text{medium (Altitude)} = \mu_D: \text{high (Altitude)} = \mu_E: \text{very high (Altitude)} = 0 \\
\mu_A: & \text{very light (Rainfall)} = \mu_B: \text{light (Rainfall)} = \mu_C: \text{medium (Rainfall)} = 0 \\
\mu_D: & \text{heavy (Rainfall)} = 0.9 \\
\mu_E: & \text{very heavy (Rainfall)} = 0.1 \\
\mu_A: & \text{very low (Temperature)} = 1.0 \\
\mu_B: & \text{low (Temperature)} = \mu_C: \text{medium (Temperature)} = \mu_D: \text{high (Temperature)} = \mu_E: \text{very high (Temperature)} = 0 \\
\mu_A: & \text{very near (Distance)} = \mu_B: \text{near (Distance)} = \mu_C: \text{somewhat distant (Distance)} = \mu_D: \text{distant (Distance)} = 0 \\
\mu_E: & \text{very distant (Distance)} = 1.0
\end{align*}
\]

The encoded message is a list shown in Figure 7.3.

![Figure 7.3](image-url)

**Figure 7.3.** Illustration of converting input information into a list that serves as an environmental message that is provided as input to the market-based rule learning system.

7.2.2 Evolutionary Learning of Fuzzy Rules

Since the fuzzy rules calculate an output directly from the input, there is no message list in the system. Such a system is so-called a single-layer learning system. The definition of the market-
based trading algorithm obviously works for rules triggering each other in the system. But what happens when there is only input and output?

At the input-stage, we put only one environmental message into the system. For those rules that have become active, their strength is decreased. The decreased amount can be thought as a payment to the outside world. At the input-stage, the net result is that the total strength of the system decreases.

For the output, the situation is equally simple. Usually, the action that a fuzzy rule-based system produces does not depend on one rule only, but on all the fired (executed) rules. If the output equals the expected solution, each of the active fuzzy rules is rewarded by the outside world with an amount $R$. This amount is added to their strengths: $S_c' = S_c + R$, in which $S_c$ is the strength of the active fuzzy rule that contributes to generate good output and $S_c'$ is its updated strength. If the generated output does not match the expected solution, no reward is given. Notice that negative rewards are not used in this system. This prevents the strength of rules from becoming negative.

For the GA procedure in the MBRL system based on fuzzy rules, the fuzzy rules are taken as individuals in the population. Each individual is encoded as a chromosome consisting of genes, and each gene in the chromosome is a floating-point number. An example is shown in Figure 7.2. The strength of an individual rule is used as its fitness. The market-based trading algorithm updates the fitness of the individual rules between two iterations of the genetic algorithm. In the initial population $S^0 = \{S^0_1, \ldots, S^0_M, S^0_L\}$, $S^0_1, \ldots, S^0_M$ are $M$ existing fuzzy rules generated by the ReFuNN technique, $S^0_{M+1}, \ldots, S^0_L$ are individuals created by random uniform variations around $\{S^0_1, \ldots, S^0_M\}$, and $L$ is the initial population size. The initial fitness of $S^0_{M+1}, \ldots, S^0_L$ is set to 0.

Following the basic structure of the MBRL system, the system architecture of an MBRL system based on fuzzy rules is shown schematically in Figure 7.4.
Figure 7.4. Market-based rule learning system based on fuzzy rules

The market-based fuzzy rule evolution consists of the following steps:

1. Input an environmental message to the system. (This represents a training data sample.)
2. The “Match Maker” compares the environmental message to the antecedent parts of all fuzzy rules and records all matches (fired rules). Issues regarding how to judge which fuzzy rules are fired were discussed in Section 2.5.5 in Chapter 2.
3. Process the recorded active rules to produce the system’s behaviour. The procedures to produce a final action from active fuzzy rules were given in Section 2.5.5 of Chapter 2. If the final action is correct, which means the system output matches the expected solution, the active fuzzy rules will get rewarded. If the final action is incorrect, no reward will be given.
4. Apply the market-based trading algorithm to reallocate the strengths of all the individual rules in the fuzzy rule base.
5. If a predetermined number of training cycles $N$ has been executed, then go to step 6. Otherwise, go to step 1. (In all experiments to be described in Chapter 8, the MBRL system was trained by all available training examples for a given problem. So the number of training
cycles \( N \) equals the number of training examples in the training data set.)

6. Apply the genetic algorithm by using the strengths of each of the individual rules as fitness measurements to discover new rules, while replacing other, low-strength rules.

7. Replace the contents of the fuzzy rule base with a new generation of rules produced by the GA procedure.

8. Apply the inference engine to evaluate the performance of the rules. If the inference performance is satisfactory or the pre-specified maximum generation of GA has been reached, end the process. Otherwise, go to step 1. The purpose of the inference engine is to examine the system’s generalization ability, which is the ability to classify correctly samples from the problem space. If the system meets a predetermined training accuracy, then the process finishes.

### 7.2.3 Method Illustration: Iris Classification Data

The description of the Iris data set can be found in Section 3.2 of Chapter 3, and a description of the creation and training process of a FuNN module can be found in Section 5.3 in Chapter 5.

By means of the Chi2-based membership function selection approach, for one partition of 10-fold cross validation, attribute *Sepal-width* was represented by 5 fuzzy variables ranging from A to E, attribute *Petal-length* was represented by 4 fuzzy variables ranging from A to D, and attribute *Petal-width* was represented by 3 fuzzy variables ranging from A to C. By using the ReFuNN procedure, 4 fuzzy rules were extracted. After encoding, they are displayed in Table 7.1.

<table>
<thead>
<tr>
<th>No</th>
<th>Rules</th>
<th>Strength</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sepal-width</td>
<td>Petal-length</td>
</tr>
<tr>
<td></td>
<td>A B C D E</td>
<td>A B C D</td>
</tr>
<tr>
<td>0</td>
<td>6.3 0 0 -4.7 0</td>
<td>0 0 0 0 7.9</td>
</tr>
<tr>
<td>1</td>
<td>0 0 0 0 0</td>
<td>0 0 0 0 0</td>
</tr>
<tr>
<td>2</td>
<td>0 0 0 0 0</td>
<td>2.1 -7.0 0 0</td>
</tr>
<tr>
<td>3</td>
<td>0 0 0 0 0</td>
<td>0 9.9 0 0</td>
</tr>
</tbody>
</table>

The initial strengths for each single fuzzy rule were set to 1.0. The payment coefficient was 0.1,
the activation-tax coefficient was 0.01, the life-tax coefficient was 0.00001, and the reward from the environment was 1.0. If the environmental message listed below is introduced to the system:

0 0 0.5 0.5 0 0 0 0 1 0 0.07 0.93 2

then when compared to the condition parts of each of the rules, two rules are matched (i.e., the overall degree of matching for the condition side of each rule is positive). They are:

Matched rule numbers: {0, 3};

By using these two active rules, the system output was calculated to be the value 2, which was same as the expected solution. Therefore, a reward of 1.0 was given to the active rules. For the two active rules, their strengths were adjusted by the equation (6.2):

\[ S' = 1 - 0.00001 \cdot 1 - 0.1 \cdot 1 - 0.01 \cdot 1 + 1.0 = 1.8899911 \]

For the remaining two inactive rules, only the life tax was taken out from their initial strengths:

\[ S' = (1-0.00001) \cdot 1 = 0.99999 \]

The updated strengths of each rule are shown in Table 7.2.

**Table 7.2.** The updated strengths of fuzzy rules after training with one example (Iris Classification)

<table>
<thead>
<tr>
<th>No</th>
<th>Strength</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.889911</td>
</tr>
<tr>
<td>1</td>
<td>0.99999</td>
</tr>
<tr>
<td>2</td>
<td>0.99999</td>
</tr>
<tr>
<td>3</td>
<td>1.8899911</td>
</tr>
</tbody>
</table>

After the system was trained with 135 examples, the strengths obtained are shown in Table 7.3.
When the strengths of the rules had stabilized, the genetic algorithm procedure was executed. The population size was set to 40, the probability of crossover was 0.8, and the probability of mutation was set to 0.001. The list of all the parameters used is shown in Table 7.4.

Table 7.4. Values of the parameters used in the experiment of market-based rule evolution starting with the ReFuNN-generated fuzzy rules in connection with the Iris data set

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>initial strength</td>
<td>1</td>
<td>Initial strength of a rule after initialization</td>
</tr>
<tr>
<td>payment coefficient</td>
<td>0.1</td>
<td>Percentage of the strength of a rule that is used as a payment to the rules that helped in activating it.</td>
</tr>
<tr>
<td>activation tax coefficient</td>
<td>0.01</td>
<td>Percentage of the strength that is used as a tax imposed on an active rule</td>
</tr>
<tr>
<td>life tax coefficient</td>
<td>0.00001</td>
<td>Percentage of the strength that is used as a tax imposed on every rule at every cycle of the system</td>
</tr>
<tr>
<td>bad reward</td>
<td>0</td>
<td>Reward for a bad answer</td>
</tr>
<tr>
<td>good reward</td>
<td>1</td>
<td>Reward for a good answer</td>
</tr>
<tr>
<td>initial rule size</td>
<td>4</td>
<td>Number of existing rules generated by the ReFuNN technique</td>
</tr>
<tr>
<td>initial population size</td>
<td>40</td>
<td>Number of individuals in the population</td>
</tr>
<tr>
<td>crossover probability</td>
<td>0.8</td>
<td>Probability of crossover between two chromosomes</td>
</tr>
<tr>
<td>mutation probability</td>
<td>0.01</td>
<td>Probability of mutation of a gene</td>
</tr>
<tr>
<td>training times</td>
<td>135</td>
<td>Number of training cycles of the system between two steps of the genetic algorithm</td>
</tr>
<tr>
<td>maxgenerations</td>
<td>2000</td>
<td>Maximum number of generations (steps) of the genetic algorithm</td>
</tr>
<tr>
<td>crowding factor</td>
<td>3</td>
<td>Number of subpopulations tested to find most similar individual to new child</td>
</tr>
<tr>
<td>crowdingsubpop</td>
<td>10</td>
<td>Number of individuals in the subpopulation</td>
</tr>
</tbody>
</table>

Based on the four existing fuzzy rules, the average results over 50 experimental runs were obtained and shown in Table 7.5. It can see that, with the average number of 3.6 rules, the
system classified an average 85.3% of the test examples correctly, which represented a significant improvement of 18.73% at the 98.7% level of confidence.

Table 7.5. A comparison of accuracy performance, and number of rules for the ReFuNN-generated rules before and after using the market-based procedure for one cross-validation trial of the Iris data set (standard deviations are shown in parentheses)

<table>
<thead>
<tr>
<th></th>
<th>ReFuNN</th>
<th>Market-based evolution starting with the ReFuNN-generated rules</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acc. on the training set (%)</td>
<td>68.44(0.0)</td>
<td>87.08(10.9)</td>
<td>0.022</td>
</tr>
<tr>
<td>Acc. on the test set (%)</td>
<td>66.60(0.0)</td>
<td>85.33(9.8)</td>
<td>0.013</td>
</tr>
<tr>
<td>Number of rules</td>
<td>4(0.000)</td>
<td>3.6(0.894)</td>
<td>0.374</td>
</tr>
</tbody>
</table>

After the MBRL system learning, the initial fuzzy rule base was modified by the GA procedure. The final fuzzy rule sets developed by the MBRL system display an interesting characteristic: only a small proportion of rules (the rules with the highest fitnesses) had influence on the output classification. A large proportion of rules in the final rule set did not make any contributions to the classification tasks. However, this large proportion of rules provided a larger space for the GA to search well performing rules.

In the 50 experimental runs, 30 experiments produced sets of three fuzzy rules with the mean predictive accuracy 88±11.9%. One representative new rule created by the MBRL system is:

\[
\text{If} \ <\text{petal-length is A, 2.15}> \ \text{and} \ <\text{petal-length is not B, 6.97}> \ \text{and} \ <\text{petal-width is not C, 9.96}>, \text{then} \ <\text{output is not Iris Setosa, 19.56}>
\]

7.3 Market-based Rule Evolution and Refinement Based on Extracted Rules from Feed-forward Neural Networks

In the past few years, there have been several efforts made to find effective algorithms to extract rules from trained feed-forward neural networks. The NeuroLinear approach is one such effort. It offers some advantages compared to other methods since it works well for problem domains with continuous input attributes and thus does not require discretization of the input data. When
extracting rules from a feed-forward neural network, there are two desirable features of the
derived rule set:

• Accuracy of classification
• Simplicity (comprehensibility) of the rule set

Unfortunately the complexity of the extracted rule set is often relatively high, due to the
complexity of the neural network. In this section, we will see how the MBRL system works on
two-level NeuroLinear-generated rules in order to achieve a higher accuracy performance and
a smaller rule size.

The detailed description of the NeuroLinear approach is presented in Section 2.5.4 of Chapter
2.

7.3.1 NeuroLinear Rule Encoding

A distinguishing feature of the NeuroLinear method (Setiono & Liu 1997b) is the fact that it
generates two levels of rules: the set of rules which describe the relationship between discretized
hidden activation values and network inputs are referred to here as rule base 1, and the set of
rules which describe the network outputs in terms of the discretized activation values of hidden
nodes are here called, rule base 2.

The format of rules in rule base 1 is as follows:

\[ R: \text{If } a_1I_1 + a_2I_2 + \ldots + a_nI_n \in (b_1, b_2) \text{ then the discrete intermediate output } = c \]

where \( n \) is the number of inputs, \( I_1, I_2, \ldots I_n \) are input variables, \( a_1, a_2 \ldots a_n \) are coefficients
associated with each of the input variables, \( (b_1, b_2) \) is a constraint boundary in which \( b_1 \) is a lower
boundary and \( b_2 \) is a higher boundary, and \( c \) is the discrete intermediate output value associated
with rule \( R \). The encoding scheme for a rule is as follows: list the coefficients, constraint
boundaries and the discrete intermediate output value in order (from left to right). In other words,
each individual rule is a list which consists of \( n \) coefficients, two boundary values, and one
discrete output. This is illustrated by the following example:
Suppose, a rule in *rule base 1* is

\[
\text{If } -3.54 \text{ Altitude} - 4.36 \text{ Rainfall} - 1.37 \text{ Temperature} + 7.49 \text{ Distance} \in (-2.94, 0.09) \text{ then the discrete intermediate output} = 5
\]

Then, a list is created as shown in Figure 7.5.

<table>
<thead>
<tr>
<th>Coefficient</th>
<th>Boundary</th>
<th>Discrete Intermediate Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3.54 -4.36 -1.37 7.49</td>
<td>-2.94 0.09</td>
<td>5</td>
</tr>
</tbody>
</table>

**Figure 7.5.** Illustration of converting a NeuroLinear rule from *rule base 1* into a list that serves as a rule representation in the market-based rule learning system

The format of rules in *rule base 2* is as shown below:

\[
\text{If } \quad \text{the discrete intermediate output}_1 = c_1 \\
\quad \text{and} \quad \text{the discrete intermediate output}_2 = c_2 \\
\quad \quad \ldots \\
\quad \text{and} \quad \text{the discrete intermediate output}_k = c_k \text{ then the final output} = Y
\]

where \( k \) is the number of discrete values in the antecedent part of a rule, \( c_1, c_2, \) and \( c_k \) are the discrete intermediate outputs, and \( Y \) is the final output.

Encoding a rule from *rule base 2* for the MBRL system is similar: the discrete values and the output are listed in order. For example a rule from *rule base 2* is

\[
\text{If } \quad \text{the discrete intermediate output}_1 = 1 \\
\quad \text{and} \quad \text{the discrete intermediate output}_2 = 4 \\
\quad \quad \text{and} \quad \text{the discrete intermediate output}_3 = 7 \text{ then the final output} = 0
\]
Then, a corresponding list is shown in Figure 7.6.

<table>
<thead>
<tr>
<th>Discrete Intermediate Outputs</th>
<th>Final Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 4 7</td>
<td>0</td>
</tr>
</tbody>
</table>

**Figure 7.6.** Illustration of converting a NeuroLinear rule from *rule base 2* into a list that serves as a rule representation in the market-based rule learning system.

The normalized training data items can be directly imported into the learning system without making any changes. An example is shown in Figure 7.7.

<table>
<thead>
<tr>
<th>Altitude</th>
<th>Rainfall</th>
<th>Temperature</th>
<th>Distance</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0112</td>
<td>0.3808</td>
<td>0.5</td>
<td>0.8333</td>
<td>2</td>
</tr>
</tbody>
</table>

**Figure 7.7.** An example of an environmental message for the market-based rule learning system based on the NeuroLinear rules.

### 7.3.2 Evolutionary Learning of NeuroLinear Rules

Unlike fuzzy rules, Neurolinear rules do not calculate an output directly from the input. Instead, there are two levels of “inference”: an output is generated by evaluating an environmental message across two levels of rule bases. So a MBRL system based on NeuroLinear rules is not a single-layer system. As already indicated for multiple-layer learning systems, the rules and the message list are the two major components to form the computational backbone. Information enters from the environment after it has been encoded in the form of a *message*. This environmental message is placed on a *message list*, where the message may then activate the rules in the *rule base 1*. When activated, a rule posts its output action to the message list. These messages may then invoke rules in the *rule base 2*. Whether an active rule actually receives a reward is determined by whether its action ultimately produces the correct output.

In the MBRL system based on the NeuroLinear rules, there are two levels of matching in
accordance with the two levels of rules\textsuperscript{1}. First, all rules in rule base 1 are matched against the environmental message and a match list (message list) is constructed, which contains the matched rules. Second, all rules in rule base 2 are compared to the matched rules from rule base 1 and a second match list is generated, which contains the active rules from rule base 2. With the matching completed, we are ready to distribute payment among the rules by using the market-based trading algorithm.

According to our general description of the market-based trading algorithm in the previous chapter, the matched rules make their payments to the previously active rules that sent the messages which matched the currently active rules (and thus invoked them). For matched rules from rule base 1, the strength of active rules is decreased by a value (proportional to its strength) which can be treated as a payment to the outside world. For every active rule from rule base 2, the strength is also decreased by a certain amount which is paid to the active rules from rule base 1 that helped in activating it. The payment is distributed equally among the contributing rules from rule base 1. Thus, every active rule from rule base 1 gets its strength increased if it is able to help to activate rules from rule base 2. Active rules from rule base 2 that are able to produce correct output are rewarded by the environment. In order to discourage nonproductive rules, two different types of tax are collected from rules: a life tax and an activation tax. A life tax is collected from all rules in both rule base 1 and rule base 2 in every cycle of system learning. An activation tax is only collected from each active rule in both rule base 1 and rule base 2. The market-based trading algorithm distributes and collects payments and taxes among rules to help assure that good rules achieve high strength and bad rules achieve relatively low strength. Thereafter strength is used as a fitness measure to facilitate a genetic search for new, possibly better rules.

Since there is a representation difference between rule base 1 and rule base 2, the genetic algorithm procedure operates a little differently on the two rule sets. While the genetic algorithm is operating on rule base 1, all the rules in rule base 2 remain unmodified. After the GA procedure operates on rule base 1 for a pre-specified number (for example, 50) epochs, the GA procedure switches to operate on rule base 2, while rule base 1 is held fixed.

\textsuperscript{1} It is possible to restrict the GA procedure so that it only operates on one particular rule base, if necessary. For example, if rule base 1 is sufficiently compact and accurate, one could decide to import rule base 2 to the GA only. In this case, only rule base 2 is updated by the GA while rule base 1 is held fixed during the entire learning process.
Figure 7.8 shows how the basic rule-based system, the market-based trading algorithm and the genetic algorithm work together to evolve the NeuroLinear rules.

The learning procedure consists of the following steps:
1. Input an environmental message to the system.
2. Compare the environmental message to the condition parts of all rules in *rule base 1* and record all matches.
3. Compare the action parts of matched rules in step 2 to the condition parts of all rules in *rule base 2* and record all matches.
4. Process the recorded matches in step 3 through the output interface. If the active rules are able to produce a correct final action, the active rules are rewarded. If the active rules do not lead
to a correct final action, then no reward is given.

5. Apply the market-based trading algorithm to reallocate the strengths of all the individual rules in rule base 1 and rule base 2.

6. If a predetermined number of training cycles $N$ has been achieved, then go to step 7. Otherwise, go to step 1. The number of training cycles $N$ equals the number of training examples in the training data set.

7. Apply the genetic algorithm by using the strengths of each of the individual rules as a fitness measure to discover new rules, while replacing other, lower-strength rules. If both rule base 1 and rule base 2 are operated on by the GA procedure, the GA procedure runs initially on rule base 1 for a predetermined number of generations. Then, the GA operates on rule base 2 for a pre-specified number of epochs. After each generation, the contents of a rule base are replaced with a new generation produced by the GA.

8. Apply the inference engine to evaluate the performance of the rules. If the inference performance is satisfactory or the pre-specified maximum number of generations of the GA procedure has been reached, end the process. Otherwise, go to step 1. Since the MBRL system allows parallel activation of rules, it is possible that more than one active rule in rule base 2 can apply their actions to the outside world at the same time. Whether an active rule is actually allowed to perform a final action is determined by its strength. The stronger a rule, the greater the chance that it can cause a final action when it is activated.

7.3.3 Genetic Search within the System

As already indicated, the market-based trading algorithm provides a clear procedure for evaluating rules and deciding among competing alternatives. The genetic algorithm provides a way of injecting new, possibly better rules into the system. By using the GA procedure, new rules are created by the reproduction process (employing selection, crossover and mutation). These rules are then placed in the population and processed by clearing payment, adding rewards, and the reinforcement mechanism to evaluate properly their role in the system. Since the GA procedure operates on rule base 1 and rule base 2 separately and differently, we must take a closer look at how the GA procedure works.

In rule base 1, each rule is an individual of the GA population. Each individual consists of three chromosomes: the Coefficient chromosome, the Boundary chromosome and the Discrete
Intermediate Output chromosome. The Coefficient chromosome and the Boundary chromosome consist of floating-point numbers, and the Discrete Intermediate Output chromosome consists of integers. Montana and Davis’s crossover and mutation methods (described in Chapter 6), are applied to the Coefficient chromosomes and to the Boundary chromosomes. In order to avoid inconsistencies (different inputs yielding identical outputs) which may cause a matching confusion between rule base 1 and rule base 2, the GA procedure does not operate on the Discrete Intermediate Output chromosome. In other words, only the Coefficient chromosomes and the Boundary chromosomes are operated on by crossover and mutation operators, while the Discrete Intermediate Output chromosomes remain unchanged during the GA process.

In rule base 2, similarly, each rule is an individual of the GA population. Each individual has two chromosomes: a Discrete Intermediate Output chromosome and a Final Output chromosome. They are all integer chromosomes. Montana and Davis’s method is used for the crossover operation on the Discrete Intermediate Output chromosome. The mutation operator is performed by randomly selecting \( n \) genes, and for each selected gene, replacing it by a random integer number selected from a given range, e.g. from 1 to 9. For a Final Output chromosome, there is only one gene which is a single integer number. The crossover operator simply swaps the integer numbers from the two parents and copies them to the offspring. The mutation operator then chooses a random integer number from a given range (from 0 to 4 for the Golf Course problem), and replaces the old integer number by the randomly selected number.

For rule base 1 and rule base 2, the initial population \( S_0^0 = \{S_{i,0}^0, ..., S_{M,0}^0, S_{L,0}^0\} \), where \( S_{i,0}^0, ..., S_{M,0}^0 \) are \( M \) existing rules generated by the NeuroLinear technique, \( S_{M+1,0}^0, ..., S_{L,0}^0 \) are individuals created by random uniform variations around \( \{S_{i,0}^0, ..., S_{M,0}^0\} \), and \( L \) is the initial population size. The initial fitness for \( S_{M+1,0}^0, ..., S_{L,0}^0 \) is set to 0.

The genetic algorithm used for rule discovery in the system works as follows:

1. Select a pair of parent individuals from the current population, the probability of selection being an increasing function of fitness.
2. With crossover probability \( p_c \), mate the pair at a randomly chosen point in the chromosome (chosen with uniform probability) to form two offspring. An illustration is shown in Figure 7.9.
If no crossover takes place, the two offspring are exact copies of their respective parents.

3. With mutation probability \( p_m \), mutate the two offspring at randomly chosen points in the chromosome (chosen with uniform probability), and insert the resulting individuals into the new population by using the *scanning* operation and the *crowding* replacement method.

4. Replace the current population with the new population.

### 7.3.4 Method Illustration: Iris Classification Data

Recall from Section 2.5.4 in Chapter 2, two level of rules were extracted by the NeuroLinear approach based on the one cross-validation trial of the Iris data set. They are listed in Table 7.6 and Table 7.7, respectively. Table 7.6 lists the coefficient, constraint boundary and discrete intermediate output value associated with each of the rules. (9999 and -9999 are used to approximate positive infinity and negative infinity, respectively).

#### Table 7.6. NeuroLinear-generated *rule base 1*

<table>
<thead>
<tr>
<th>Coefficient</th>
<th>Boundary</th>
<th>Discrete Intermediate Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00 4.95</td>
<td>-8.93 -12.7</td>
<td>9999 -12.5 1</td>
</tr>
<tr>
<td>0.00 4.95</td>
<td>-8.93 -12.7</td>
<td>-12.5 9999 2</td>
</tr>
<tr>
<td>0.00 0.00</td>
<td>0.00 2.58</td>
<td>-9999 0.97 3</td>
</tr>
<tr>
<td>0.00 0.00</td>
<td>0.00 2.58</td>
<td>0.97 9999 4</td>
</tr>
</tbody>
</table>
Table 7.7. NeuroLinear-generated rule base 2

<table>
<thead>
<tr>
<th>Discrete Intermediate Output</th>
<th>Final Output Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Based on these two sets of rules, the inference performance on the training set and test set is shown in Table 7.8.

Table 7.8. Performance of the NeuroLinear-generated rules

<table>
<thead>
<tr>
<th>Data Set</th>
<th>#Correct Patterns</th>
<th>#Incorrect Patterns</th>
<th>%Correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training Set</td>
<td>129</td>
<td>6</td>
<td>95.6</td>
</tr>
<tr>
<td>Test Set</td>
<td>12</td>
<td>3</td>
<td>80</td>
</tr>
</tbody>
</table>

Three experiments were conducted. The first experiment was to test if the MBRL system could select well performing rules and eliminate the poor performing rules. In this experiment, rule base 1 values (shown in Table 7.6) were held fixed and the procedure focused on rule base 2 (shown in Table 7.7). The second experiment let the system run on both rule base 1 and rule base 2 to see if the system could improve the overall performance on both the simplicity and accuracy of the rule sets. The third experiment started from scratch. It was designed to test if the MBRL system has the ability to work on purely random-generated rule sets with no prior knowledge. For the first two experiments, one partition of the 10-fold cross validation containing 135 training samples and 15 test samples was used. For the last experiment, ten partitions of 10-fold cross validation were used.

Experiment 1
The initial strengths of all 4 rules in Table 7.7 were set to 1.0. The payment coefficient was 0.1, the activation-tax coefficient was 0.01, the life-tax coefficient was 0.00001, and the reward was 1.0. The GA procedure was not used in this experiment. After the system was trained by 135 training examples, the system identified only 3 rules having influence on the output classification. The rule set was sorted from high to low in terms of rule strength and is shown in Table 7.9.
Table 7.9. Updated rule base 2

<table>
<thead>
<tr>
<th>Discrete Intermediate Output</th>
<th>Final Output Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

By using these three rules, the rule performance on the training data and the test data were the same as those shown in Table 7.8. This result shows that the original four rules were reduced to three rules with no loss of performance accuracy. Three good rules were selected, while a poorly performing rule was thrown away. In the original rule base, this poorly performing rule was inconsistent (identical inputs yielding different outputs) with the rest of the rules. The system, in this case at least, made the correct decision to select automatically those rules activated more frequently by the environmental messages.

**Experiment 2**

In this experiment, both rule base 1 and rule base 2 were operated on by the GA procedure. The values of the parameters used in the experiment are shown in Table 7.10.
Table 7.10. Parameters used in the experiment with market-based rule evolution starting with *rule base 1* and *rule base 2* generated by the NeuroLinear technique for Iris Classification

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>initial strength</td>
<td>1</td>
<td>Initial strength of a rule after initialization</td>
</tr>
<tr>
<td>payment coefficient</td>
<td>0.1</td>
<td>Percentage of the strength of a rule that is used as a payment to the rules that helped in activating it.</td>
</tr>
<tr>
<td>activation tax coefficient</td>
<td>0.01</td>
<td>Percentage of the strength that is used as a tax imposed on an active rule</td>
</tr>
<tr>
<td>life tax coefficient</td>
<td>0.00001</td>
<td>Percentage of the strength that is used as a tax imposed on every rule at every cycle of the system</td>
</tr>
<tr>
<td>good reward</td>
<td>1</td>
<td>Reward for a good answer</td>
</tr>
<tr>
<td>training times</td>
<td>135</td>
<td>Number of training cycles of the system between two steps of the genetic algorithm</td>
</tr>
<tr>
<td>maxgenerations</td>
<td>2000</td>
<td>Maximum number of generations (steps) of the genetic algorithm</td>
</tr>
<tr>
<td>crowding factor</td>
<td>3</td>
<td>Number of subpopulations tested to find a individual most similar to a new child</td>
</tr>
<tr>
<td>crowdingsubpop</td>
<td>5</td>
<td>Number of individuals in the subpopulation</td>
</tr>
</tbody>
</table>

Parameters only associated with *rule base 1*

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>initial rule size</td>
<td>4</td>
<td>Number of existing rules in <em>rule base 1</em> generated by the NeuroLinear technique</td>
</tr>
<tr>
<td>population size</td>
<td>40</td>
<td>Number of individuals in the population</td>
</tr>
<tr>
<td>crossover probability</td>
<td>0.7</td>
<td>Probability of crossover between two chromosomes</td>
</tr>
<tr>
<td>mutation probability</td>
<td>0.01</td>
<td>Probability of mutation of a gene</td>
</tr>
<tr>
<td>maxgen-rulebase1</td>
<td>50</td>
<td>Maximum number of generations (running steps) of the genetic algorithm based on <em>rule base 1</em> before switching to <em>rule base 2</em></td>
</tr>
</tbody>
</table>

Parameters only associated with *rule base 2*

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>initial rule size</td>
<td>4</td>
<td>Number of existing rules in <em>rule base 2</em> generated by the NeuroLinear technique</td>
</tr>
<tr>
<td>population size</td>
<td>20</td>
<td>Number of individuals in the population</td>
</tr>
<tr>
<td>crossover probability</td>
<td>0.7</td>
<td>Probability of crossover between two chromosomes</td>
</tr>
<tr>
<td>mutation probability</td>
<td>0.001</td>
<td>Probability of mutation of a gene</td>
</tr>
<tr>
<td>maxgen-rulebase2</td>
<td>20</td>
<td>Maximum number of generations (running steps) of the genetic algorithm based on <em>rule base 2</em> before switching to <em>rule base 1</em></td>
</tr>
</tbody>
</table>

Table 7.11 provides the average results over 50 experimental runs and associated statistical analysis. The refined rule sets classified an average 93.3% of the test examples correctly, which represented a significant improvement of 13.3% when compared with the accuracy performance of the original NeuroLinear rules (80%).
Table 7.11. A comparison of accuracy performance, and number of rules for the NeuroLinear-generated rules before and after using the market-based procedure for one cross-validation trial of the Iris data set (standard deviations are shown in parentheses)

<table>
<thead>
<tr>
<th></th>
<th>NeuroLinear</th>
<th>Market-based evolution starting with the NeuroLinear-generated rules (GA operated on rule base 1 and rule base 2)</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Acc. on the training set (%)</td>
<td>95.60(0.0)</td>
<td>99.26(0.0)</td>
<td>0.000</td>
</tr>
<tr>
<td>Acc. on the test set (%)</td>
<td>80.00(0.0)</td>
<td>93.33(0.0)</td>
<td>0.000</td>
</tr>
<tr>
<td>Number of rules I</td>
<td>4(0.000)</td>
<td>4(0.000)</td>
<td></td>
</tr>
<tr>
<td>Number of rules II</td>
<td>4(0.000)</td>
<td>3(0.000)</td>
<td>0.000</td>
</tr>
</tbody>
</table>

After the MBRL system learning, the initial four rules in rule base 1 were modified by the GA procedure. A representative GA-created rule base 1 is shown in Table 7.12. It is observed that, among 50 experimental runs, the three well performing rules in rule base 2 shown in Table 7.9 were always selected by the system.

Table 7.12. Updated rule base 1 (the rules that were modified by the GA procedure are underlined)

<table>
<thead>
<tr>
<th>Coefficient</th>
<th>Boundary</th>
<th>Discrete Intermediate Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4.95</td>
<td>-8.93</td>
</tr>
<tr>
<td>0</td>
<td>4.95</td>
<td>-8.93</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>2.58</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>2.58</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0.1</td>
</tr>
</tbody>
</table>

As shown in Table 7.12, one coefficient value was changed in rule base 1. With updated rule base 1 and rule base 2 shown in Table 7.12 and Table 7.9, an accuracy of 99.26% on the training set and 93.33% on the testing set were achieved.

Experiment 3
The initial rule bases had the same format as the NeuroLinear-generated rules, which use the
coefficients, the boundaries, and the discrete intermediate outputs representing the first-level rules, and the combinations of the discrete intermediate outputs forming the second-level rules. All coefficients and boundaries in the first-level rule set were initialized to 0, and all combinations of the discrete intermediate outputs in the second-level were selected randomly. The values of the parameters used in this experiment are shown in Appendix J.

Based on the ten repetitions of 10-fold cross validation, the average rule performance, average number of rules, and statistical analysis with those of randomly generated rules are provided in Table 7.13. In the experiment, for each partition of 10-fold cross validation, ten experimental runs were conducted. So the experimental results shown in Table 7.13 are the mean average of system behaviour over 100 iterations. It can be seen that the rule performance after MBRL learning had significant improvement on both the training set and the testing set.

Table 7.13. A comparison of accuracy rule performance and number of rules before and after the market-based rule learning system was applied and started without prior knowledge for the Iris data set (standard deviations are shown in parentheses)

<table>
<thead>
<tr>
<th></th>
<th>Before</th>
<th>After</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acc. on the training set (%)</td>
<td>0.00(0.0)</td>
<td>96.80(3.1)</td>
<td>0.000</td>
</tr>
<tr>
<td>Acc. on the test set (%)</td>
<td>0.00(0.0)</td>
<td>93.32(4.4)</td>
<td>0.000</td>
</tr>
<tr>
<td>Number of rules</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>40(0.000)</td>
<td>3.7(0.483)</td>
<td>0.000</td>
</tr>
<tr>
<td>II</td>
<td>20(0.000)</td>
<td>3.0(0.000)</td>
<td>0.000</td>
</tr>
</tbody>
</table>

After MBRL system learning, rule base 1 and rule base 2 were updated completely and the number of rules in rule based 1 and rule base 2 was largely reduced. The representative rule sets are shown in Table 7.14 and Table 7.15. In Table 7.15, the rules in rule base 2 are sorted from high to low in terms of rule strength.
Table 7.14. The evolved rule base 1 generated by the MBRL system starting with no prior knowledge for the Iris data set (the rules that were modified by the GA procedure are underlined)

<table>
<thead>
<tr>
<th>Coefficient</th>
<th>Boundary</th>
<th>Discrete Intermediate Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.48</td>
<td>0.37</td>
<td>-0.78</td>
</tr>
<tr>
<td>-0.02</td>
<td>0.72</td>
<td>-1.68</td>
</tr>
<tr>
<td>0.22</td>
<td>1.37</td>
<td>-1.44</td>
</tr>
</tbody>
</table>

Table 7.15. The evolved rule base 2 generated by the MBRL system starting with no prior knowledge for the Iris data set (the rules that were modified by the GA procedure are shown underlined)

<table>
<thead>
<tr>
<th>Discrete Intermediate Output</th>
<th>Final Output Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

From Table 7.14 and Table 7.15, it is seen that the number of rules was cut from 40 to 3 in rule base 1 and from 20 to 3 in rule base 2 after market-based rule evolution. It is interesting to note that the newly created rules in rule base 2 had only one antecedent per rule which means that more instances were covered by such a rule. The evolved rule sets shown in Table 7.14 and Table 7.15 had an accuracy of 97.8% on the training set and an accuracy of 93.3% on the test set.

7.4 Summary

This chapter has described how the MBRL system can be used as a post-processing tool to evolve and improve the quality of extracted rules from a fuzzy neural network and a feed-forward neural network. In terms of fuzzy rules extracted from fuzzy neural networks and NeuroLinear rules extracted from feed-forward neural networks, we have illustrated how to encode and feed input information and rules to the system, and how training is carried out with the system. The illustrative examples of the Iris Classification have demonstrated that the MBRL system can lead to improved rule performance by reducing the number of rules while increasing the generalization ability.
Although the conducted experiment has demonstrated the MBRL system is capable of working on randomly selected rule sets for given problems, for practical applications, it is sometimes better to take into account pre-established knowledge instead of attempting to find a good solution from scratch.

In the next chapter, experiments with the MBRL system using six examples that span a range of applications are presented. For comparison and evaluation purposes, existing well-known knowledge extraction techniques, such as the C4.5rules and the rule generator X2R, are also applied to these data sets.
Chapter 8
Experimental Implementation and Evaluation

8.1 Introduction
In this chapter we first discuss some of issues concerning how the experiments were conducted. Six selected data sets described previously in Chapter 3 were used for experiments. One set of experiments conducted on these data sets with the existing four rule extraction techniques (the C4.5rules, the X2R approach, the NeuroLinear approach and the ReFuNN approach) are presented in Section 8.3. Another set of experiments associated with market-based rule evolution and refinement (based on the ReFuNN-generated fuzzy rules, the NeuroLinear-generated rules, and two-level randomly selected rules), are then presented in Section 8.4. In order to evaluate the performance of the rules, particularly those rules refined by the MBRL system, Section 8.5 presents a collective analysis of these experimental results, with attention given to the rule evaluation criteria presented in Section 1.3 of Chapter 1. Section 8.6 summarizes the whole chapter.

8.2 Methodology
Some of the significant procedures that were followed in the experimental work with the existing four rule extraction techniques are illustrated as follows:

C4.5rules
The C4.5rules method is an approach representative of direct rule-generation methods. C4.5rules have been chosen because it has been commonly used in the machine learning community. A description of the C4.5rules method can be found in Section 2.5.2 of Chapter 2. In all the experiments conducted in this chapter, the C4.5rules method was run using its default parameter values (Quinlan 1993).

X2R
The X2R rule generator is a straightforward approach for generating a set of rules from discrete-valued data sets. For continuous-valued data, the input variables must first be discretised, and we used the Chi2 discretization procedure for this step. In the experiments associated with the X2R
method, the discretized data sets for six problem domains were all obtained from the Chi2 experimental procedures described in Chapter 4. For descriptions of the X2R and Chi2 methods, the reader is referred to Sections 2.5.3 and 2.4.2 in Chapter 2.

**ReFuNN**

The ReFuNN approach is a fuzzy rule extraction technique based on a fuzzy neural network model (FuNN). All fuzzy rules in the experiments presented in this chapter were extracted from the FuNN model using the ReFuNN approach. Except for the Golf Course Problem, the membership function selections of the FuNN modules were based on the Chi2-based membership function selection approach. The extracted fuzzy rules for the Golf Course Problem presented in this chapter were based on a FuNN module with fixed centre-based membership function selections. Descriptions and illustrative examples of the ReFuNN approach, the Chi2-based membership function selection method, and the fixed centre-based membership function selection method, were presented in Section 2.5.5 in Chapter 2 and Section 5.3 in Chapter 5.

**NeuroLinear**

In the experiments, the two levels of neuron rules were extracted from 3-layer feed-forward neural networks using the NeuroLinear approach. The neural networks were first trained and pruned before rule extraction took place. Pruned neural networks for all six learning examples were obtained from the earlier conducted experiments in applying the PBS pruning scheme to simplify the neural network architecture. Descriptions of the PBS pruning scheme and the PBS-related experimental results were presented in Sections 4.2 and 4.3 of Chapter 4, respectively. An introduction and demonstration of the NeuroLinear approach was presented in Section 2.5.4 of Chapter 2.

For the experiments with the market-based rule learning (MBRL) system, three major experiments were conducted:

- Applying the MBRL system to refine ReFuNN-generated fuzzy rules.
- Applying the MBRL system to refine NeuroLinear-generated rules.
- Testing the MBRL system’s capability when it starts with no prior knowledge.
For all the experiments, the cross-validation resampling technique was employed. For all the experimental results reported in this chapter, the mean values and their standard deviations are given. In the experiments with the MBRL system, for each partition of 10-fold cross validation, 10 experimental runs were conducted. The reported experimental results are the average of system behaviour exhibited over 100 trials.

Appendices A – G show representative rules extracted and refined by different techniques for each problem domain. The parameters used in the MBRL system related experiments are presented in Appendices H – J.

8.3 Experiments with Existing Rule Extraction Methods

In this section, we apply the following four existing machine learning approaches to extract inference rules about the problem domains:

- C4.5 rules based on decision trees
- X2R rule generator
- Fuzzy rule extraction approach ReFuNN
- NeuroLinear rule extraction approach

The rule sets extracted by the ReFuNN approach and the NeuroLinear approach were then used as initial rule bases in the experiments with market-based rule learning. This is reported in Section 8.4.

8.3.1 Iris Classification

Table 8.1 provides a summary of the accuracy on both the training set and the test set for each of the four rule extraction techniques – the figures in parentheses are the standard deviations. Table 8.2 presents the P-values when comparing test accuracy between pairs of techniques (C4.5 rules versus X2R, C4.5 rules versus ReFuNN, NeuroLinear versus X2R, and NeuroLinear versus ReFuNN). It can be seen that the rule sets extracted by means of two existing techniques (the C4.5 rules and the NeuroLinear) performed well on the Iris data set. The C4.5-generated rules had performances that were significantly better than the ReFuNN-generated rules and the rules extracted by the X2R techniques. The NeuroLinear-generated rules performed significantly better.
than the rules extracted by the ReFuNN technique. There was no significant difference in the test accuracy of generated rules when comparing C4.5rules versus NeuroLinear, and ReFuNN versus X2R.

**Table 8.1.** Accuracy(%) of the rule sets extracted from the four existing rule extraction techniques in connection with the Iris data set (standard deviations are in parentheses)

<table>
<thead>
<tr>
<th>Rule Set</th>
<th>Acc. on training set (%)</th>
<th>Acc. on test set (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C4.5rules</td>
<td>97.47(0.5)</td>
<td>95.98(3.5)</td>
</tr>
<tr>
<td>X2R</td>
<td>94.52(4.6)</td>
<td>87.99(6.9)</td>
</tr>
<tr>
<td>ReFuNN</td>
<td>77.00(14.8)</td>
<td>76.14(12.1)</td>
</tr>
<tr>
<td>NeuroLinear</td>
<td>96.67(0.8)</td>
<td>92.00(6.9)</td>
</tr>
</tbody>
</table>

**Table 8.2.** The P-values associated with test accuracy (Iris data set)

<table>
<thead>
<tr>
<th>Rule Set</th>
<th>C4.5rules</th>
<th>NeuroLinear</th>
</tr>
</thead>
<tbody>
<tr>
<td>X2R</td>
<td>0.018</td>
<td>0.295</td>
</tr>
<tr>
<td>ReFuNN</td>
<td>0.007</td>
<td>0.022</td>
</tr>
</tbody>
</table>

Table 8.3 provides a summary of the rule complexity (number of rules and number of antecedents per rule) for each of the four rule extraction techniques – the figures in parentheses are the standard deviations, Row I means *rule base 1* associated with the NeuroLinear-generated rules, and Row II means *rule base 2* associated with the NeuroLinear-generated rules. Although the first-level rule set (*rule base 1*) associated with the NeuroLinear approach consisted of a set of inequalities that involved a number of coefficients and boundaries, we identify a single inequality of a rule (for example, $2.58 \times I3 < 0.97$) in *rule base 1* as one antecedent of a rule in Table 8.3. This is also the case for all the other following tables related to the NeuroLinear rules and the MBRL-refined rules.
Table 8.3. Complexity (number of rules and number of antecedents per rule) of the rule sets extracted from the four existing rule extraction techniques in connection with the Iris data set (standard deviations are in parentheses)

<table>
<thead>
<tr>
<th>Technique</th>
<th>Number of rules</th>
<th>Number of antecedents per rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>C4.5rules</td>
<td>5.00(0.000)</td>
<td>1.45(0.158)</td>
</tr>
<tr>
<td>X2R</td>
<td>5.60(1.350)</td>
<td>1.79(0.379)</td>
</tr>
<tr>
<td>ReFuNN</td>
<td>3.71(0.756)</td>
<td>3.40(0.801)</td>
</tr>
<tr>
<td>NeuroLinear</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>3.70(0.483)</td>
<td>1.00(0.000)</td>
</tr>
<tr>
<td>II</td>
<td>3.30(0.675)</td>
<td>1.70(0.483)</td>
</tr>
</tbody>
</table>

8.3.2 Pima Indians Diabetes Data

A summary of the accuracy on both the training set and the test set for each of the four rule extraction techniques applied to the Pima Indians Diabetes data set is provided in Table 8.4. Table 8.5 presents the P-values when comparing test accuracy between pairs of techniques (C4.5rules versus X2R, C4.5rules versus ReFuNN, NeuroLinear versus X2R, and NeuroLinear versus ReFuNN). It can be seen that the rules extracted by the C4.5rules technique and by the NeuroLinear technique were significantly more accurate than the X2R-generated rules and the ReFuNN-generated rules. A much poorer predictive performance was obtained when the ReFuNN technique was applied. There was no significant difference in the test accuracy of generated rules when comparing C4.5rules and NeuroLinear.

Table 8.4. Accuracy(%) of the rule sets extracted from the four existing rule extraction techniques in connection with the Pima Indians Diabetes data set (standard deviations are in parentheses)

<table>
<thead>
<tr>
<th>Technique</th>
<th>Acc. on training set (%)</th>
<th>Acc. on test set (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C4.5rules</td>
<td>82.88(2.1)</td>
<td>74.20(4.3)</td>
</tr>
<tr>
<td>X2R</td>
<td>62.70(7.6)</td>
<td>61.00(9.9)</td>
</tr>
<tr>
<td>ReFuNN</td>
<td>51.00(6.3)</td>
<td>48.25(0.5)</td>
</tr>
<tr>
<td>NeuroLinear</td>
<td>77.10(1.4)</td>
<td>71.10(3.4)</td>
</tr>
</tbody>
</table>
Table 8.5. The P-values associated with test accuracy (Pima Indians Diabetes data set)

<table>
<thead>
<tr>
<th></th>
<th>C4.5rules</th>
<th>NeuroLinear</th>
</tr>
</thead>
<tbody>
<tr>
<td>X2R</td>
<td>0.003</td>
<td>0.005</td>
</tr>
<tr>
<td>ReFuNN</td>
<td>0.000</td>
<td>0.000</td>
</tr>
</tbody>
</table>

Table 8.6 provides a summary of the rule complexity (number of rules and number of antecedents per rule) for each of the four rule extraction techniques. It can be seen that a large number of rules were extracted by the X2R technique.

Table 8.6. Complexity (number of rules and number of antecedents per rule) of the rule sets extracted from the four existing rule extraction techniques in connection with the Pima Indians Diabetes data set (standard deviations are in parentheses)

<table>
<thead>
<tr>
<th></th>
<th>Number of rules</th>
<th>Number of antecedents per rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>C4.5rules</td>
<td>10.70(1.829)</td>
<td>3.12(0.380)</td>
</tr>
<tr>
<td>X2R</td>
<td>38.25(4.270)</td>
<td>5.04(0.310)</td>
</tr>
<tr>
<td>ReFuNN</td>
<td>10.00(0.000)</td>
<td>8.25(1.140)</td>
</tr>
<tr>
<td>NeuroLinear</td>
<td>I 3.67(1.580)</td>
<td>1.00(0.000)</td>
</tr>
<tr>
<td></td>
<td>II 7.33(3.840)</td>
<td>1.56(0.527)</td>
</tr>
</tbody>
</table>

8.3.3 Wine Recognition Data

Table 8.7 provides a summary of the accuracy on both the training set and the test set for each of the four rule extraction techniques applied to the Wine Recognition data set. Table 8.8 presents the P-values when comparing test accuracy between pairs of techniques (C4.5rules versus X2R, C4.5rules versus ReFuNN, NeuroLinear versus X2R, and NeuroLinear versus ReFuNN). It shows that the rule sets extracted from three existing rule extraction techniques (the C4.5rules, the NeuroLinear, and the X2R) performed well on the Wine Recognition data set. However, the rules extracted by the C4.5rules technique and by the NeuroLinear technique had performances that were superior to the ReFuNN-generated rules and the rules extracted from the X2R technique.
Table 8.7. Accuracy(%) of the rule sets extracted from the four existing rule extraction techniques applied to the Wine Recognition data set (standard deviations are in parentheses)

<table>
<thead>
<tr>
<th>Rule Set</th>
<th>Acc. on training set (%)</th>
<th>Acc. on test set (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C4.5rules</td>
<td>98.90(0.6)</td>
<td>92.73(7.4)</td>
</tr>
<tr>
<td>X2R</td>
<td>90.20(7.3)</td>
<td>83.70(9.8)</td>
</tr>
<tr>
<td>ReFuNN</td>
<td>76.25(0.0)</td>
<td>70.00(6.1)</td>
</tr>
<tr>
<td>NeuroLinear</td>
<td>98.53(0.4)</td>
<td>93.09(9.4)</td>
</tr>
</tbody>
</table>

Table 8.8. The P-values associated with test accuracy (Wine Recognition data set)

<table>
<thead>
<tr>
<th></th>
<th>C4.5rules</th>
<th>NeuroLinear</th>
</tr>
</thead>
<tbody>
<tr>
<td>X2R</td>
<td>0.033</td>
<td>0.050</td>
</tr>
<tr>
<td>ReFuNN</td>
<td>0.000</td>
<td>0.000</td>
</tr>
</tbody>
</table>

Table 8.9 provides a summary of the rule complexity (number of rules and number of antecedents per rule) for each of the four rule extraction techniques.

Table 8.9. Complexity (number of rules and number of antecedents per rule) of the rule sets extracted from the four existing rule extraction techniques applied to the Wine Recognition data set (standard deviations are in parentheses)

<table>
<thead>
<tr>
<th>Rule Set</th>
<th>Number of rules</th>
<th>Number of antecedents per rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>C4.5rules</td>
<td>4.20(0.919)</td>
<td>2.18(0.215)</td>
</tr>
<tr>
<td>X2R</td>
<td>7.70(1.730)</td>
<td>2.95(0.597)</td>
</tr>
<tr>
<td>ReFuNN</td>
<td>6.00(0.000)</td>
<td>4.12(1.150)</td>
</tr>
<tr>
<td>NeuroLinear</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>4.40(0.966)</td>
<td>1.00(0.000)</td>
</tr>
<tr>
<td>II</td>
<td>4.20(1.135)</td>
<td>2.00(0.000)</td>
</tr>
</tbody>
</table>

8.3.4 Golf Course Problem
A summary of the accuracy on both the training set and the test set for each of the four rule extraction techniques applied to the Golf Course Problem is provided in Table 8.10. Table 8.11 presents the P-values when comparing test accuracy between pairs of techniques (C4.5rules versus X2R, C4.5rules versus ReFuNN, NeuroLinear versus X2R, and NeuroLinear versus
ReFuNN). It can be seen that the rules extracted by the C4.5rules technique and by the NeuroLinear technique were significantly more accurate than the X2R-generated rules and the ReFuNN-generated rules. There was no significant difference in the test accuracy of generated rules when comparing C4.5rules versus NeuroLinear, and ReFuNN versus X2R.

**Table 8.10.** Accuracy(%) of the rule sets extracted from the four existing rule extraction techniques applied to the Golf Course Problem (standard deviations are in parentheses)

<table>
<thead>
<tr>
<th></th>
<th>Acc. on training set (%)</th>
<th>Acc. on test set (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C4.5rules</td>
<td>74.51(0.6)</td>
<td>70.67(3.3)</td>
</tr>
<tr>
<td>X2R</td>
<td>73.00(6.9)</td>
<td>62.87(9.6)</td>
</tr>
<tr>
<td>ReFuNN</td>
<td>62.60(0.5)</td>
<td>60.40(4.8)</td>
</tr>
<tr>
<td>NeuroLinear</td>
<td>70.46(1.9)</td>
<td>70.16(3.4)</td>
</tr>
</tbody>
</table>

**Table 8.11.** The P-values associated with test accuracy (Golf Course Problem)

<table>
<thead>
<tr>
<th></th>
<th>C4.5rules</th>
<th>NeuroLinear</th>
</tr>
</thead>
<tbody>
<tr>
<td>X2R</td>
<td>0.013</td>
<td>0.015</td>
</tr>
<tr>
<td>ReFuNN</td>
<td>0.000</td>
<td>0.000</td>
</tr>
</tbody>
</table>

Table 8.12 provides a summary of the rule complexity (number of rules and number of antecedents per rule) for each of the four rule extraction techniques. It is shown that a large amount of rules were extracted by the X2R technique.

**Table 8.12.** Complexity (number of rules and number of antecedents per rule) of the rule sets extracted from the four existing rule extraction techniques applied to the Golf Course Problem (standard deviations are in parentheses)

<table>
<thead>
<tr>
<th></th>
<th>Number of rules</th>
<th>Number of antecedents per rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>C4.5rules</td>
<td>18.00(1.000)</td>
<td>3.73(0.019)</td>
</tr>
<tr>
<td>X2R</td>
<td>83.60(14.430)</td>
<td>3.78(0.160)</td>
</tr>
<tr>
<td>ReFuNN</td>
<td>20.00(0.000)</td>
<td>7.20(0.023)</td>
</tr>
<tr>
<td>NeuroLinear</td>
<td>I</td>
<td>9.50(1.380)</td>
</tr>
<tr>
<td></td>
<td>II</td>
<td>24.50(7.31)</td>
</tr>
</tbody>
</table>
8.3.5 Cook Islands Sea Cucumber Habitat Data

Table 8.13 provides a summary of the accuracy on both the training set and the test set for each of the four rule extraction techniques in connection with the Cook Islands Sea Cucumber Habitat data set. Table 8.14 presents the P-values when comparing the test accuracy between pairs of techniques (C4.5rules versus ReFuNN, NeuroLinear versus ReFuNN, and X2R versus ReFuNN). They show that the rule sets extracted from three rule extraction techniques: the C4.5rules, the X2R and the NeuroLinear, performed well on the Cook Islands Sea Cucumber Habitat data set. However, the rules extracted by the ReFuNN techniques had a poor performance.

Table 8.13. Accuracy(%) of the rule sets extracted from the four existing rule extraction techniques in connection with the Cook Islands Sea Cucumber Habitat data set (standard deviations are in parentheses)

<table>
<thead>
<tr>
<th></th>
<th>Acc. on training set (%)</th>
<th>Acc. on test set (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C4.5rules</td>
<td>92.87(2.2)</td>
<td>82.68(9.2)</td>
</tr>
<tr>
<td>X2R</td>
<td>84.20(9.5)</td>
<td>80.90(13.1)</td>
</tr>
<tr>
<td>ReFuNN</td>
<td>69.38(9.4)</td>
<td>60.44(10.4)</td>
</tr>
<tr>
<td>NeuroLinear</td>
<td>90.16(1.6)</td>
<td>82.69(8.1)</td>
</tr>
</tbody>
</table>

Table 8.14. The P-values associated with test accuracy (Cook Islands Sea Cucumber Habitat data set)

<table>
<thead>
<tr>
<th></th>
<th>C4.5rules</th>
<th>NeuroLinear</th>
<th>X2R</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReFuNN</td>
<td>0.038</td>
<td>0.045</td>
<td>0.050</td>
</tr>
</tbody>
</table>

Table 8.15 provides a summary of the rule complexity (number of rules and number of antecedents per rule) for each of the four rule extraction techniques.
Table 8.15. Complexity (number of rules and number of antecedents per rule) of the rule sets extracted from the four existing rule extraction techniques in connection with the Cook Islands Sea Cucumber data set (standard deviations are in parentheses)

<table>
<thead>
<tr>
<th></th>
<th>Number of rules</th>
<th>Number of antecedents per rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>C4.5rules</td>
<td>4.30(1.767)</td>
<td>1.93(0.292)</td>
</tr>
<tr>
<td>X2R</td>
<td>6.30(1.889)</td>
<td>2.77(0.670)</td>
</tr>
<tr>
<td>ReFuNN</td>
<td>6.80(1.789)</td>
<td>5.76(0.220)</td>
</tr>
<tr>
<td>NeuroLinear</td>
<td></td>
<td></td>
</tr>
<tr>
<td>II</td>
<td>3.75(2.870)</td>
<td>1.00(0.000)</td>
</tr>
<tr>
<td>II</td>
<td>4.44(2.404)</td>
<td>1.70(0.483)</td>
</tr>
</tbody>
</table>

8.3.6 New Zealand Asthma Incidence Data

A summary of the accuracy on both the training set and the test set for each of the four rule extraction techniques in connection with the New Zealand Asthma Incidence data set is provided in Table 8.16. Table 8.17 presents the P-values when comparing test accuracy between pairs of techniques (NeuroLinear versus C4.5rules, NeuroLinear versus X2R, and NeuroLinear versus ReFuNN). It can be observed that the rule sets extracted from all four rule extraction techniques had relatively poor performances. However, the accuracy performance of NeuroLinear-generated rules was significantly superior to the rules generated by other techniques.

Table 8.16. Accuracy(%) of the rule sets extracted from the four existing rule extraction techniques in connection with the New Zealand Asthma Incidence data set (standard deviations are in parentheses)

<table>
<thead>
<tr>
<th></th>
<th>Acc. on training set (%)</th>
<th>Acc. on test set (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C4.5rules</td>
<td>57.97(1.1)</td>
<td>50.90(3.9)</td>
</tr>
<tr>
<td>X2R</td>
<td>53.20(1.2)</td>
<td>50.50(2.2)</td>
</tr>
<tr>
<td>ReFuNN</td>
<td>50.30(0.8)</td>
<td>49.90(0.6)</td>
</tr>
<tr>
<td>NeuroLinear</td>
<td>57.70(3.3)</td>
<td>52.70(5.1)</td>
</tr>
</tbody>
</table>
Table 8.17. The P-values associated with test accuracy (New Zealand Asthma Incidence data set)

<table>
<thead>
<tr>
<th></th>
<th>NeuroLinear</th>
</tr>
</thead>
<tbody>
<tr>
<td>C4.5rules</td>
<td>0.048</td>
</tr>
<tr>
<td>X2R</td>
<td>0.044</td>
</tr>
<tr>
<td>ReFuNN</td>
<td>0.019</td>
</tr>
</tbody>
</table>

Table 8.18 provides a summary of the rule complexity (number of rules and number of antecedents per rule) for each of the four rule extraction techniques.

Table 8.18. Complexity (number of rules and number of antecedents per rule) of the rule sets extracted from the four existing rule extraction techniques in connection with the New Zealand Asthma Incidence data set (standard deviations are in parentheses)

<table>
<thead>
<tr>
<th></th>
<th>Number of rules</th>
<th>Number of antecedents per rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>C4.5rules</td>
<td>10.40(1.817)</td>
<td>1.36(0.323)</td>
</tr>
<tr>
<td>X2R</td>
<td>82.00(10.720)</td>
<td>2.08(0.260)</td>
</tr>
<tr>
<td>ReFuNN</td>
<td>10.00(0.000)</td>
<td>8.20(0.003)</td>
</tr>
<tr>
<td>NeuroLinear</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>6.70(0.383)</td>
<td>1.00(0.000)</td>
</tr>
<tr>
<td>II</td>
<td>16.30(0.575)</td>
<td>3.70(0.283)</td>
</tr>
</tbody>
</table>

8.4 Experiments with the Market-based Rule Learning System

This section presents the experimental results for the MBRL system on the six learning problems as described earlier. The experiments were carried out to investigate three areas of interest:

1. Applying the MBRL system to improve the generalization capability and reduce the complexity of fuzzy rules extracted by the ReFuNN approach.
2. Using the MBRL system to provide a better overall performance of NeuroLinear rules when considering both the simplicity and accuracy of the rule set.
3. Testing the MBRL system’s capability as an independent learning model and a knowledge discovery tool when no prior knowledge is given.

For the third experimental area of interest, the initial rule bases had the same format as the
NeuroLinear-generated rules, which use the coefficients, boundaries, and discrete intermediate outputs representing the first-level rules, and the combinations of the discrete intermediate outputs forming the second-level rules. All coefficients and boundaries in the first-level rule set were initialized to 0, and all combinations of the discrete intermediate outputs in the second-level rule set were selected randomly.

For one of these six data sets, the Cook Islands Sea Cucumber Data, we also describe how the MBRL system was applied in combination with the Logistic Regression statistic method (Ryan 1997).

### 8.4.1 Iris Classification

**Market-based rule evolution starting with the ReFuNN-generated fuzzy rules**

Table 8.19 provides a comparison of accuracy, number of rules, and number of antecedents per rule for the ReFuNN-generated rules before and after using the MBRL system learning. Their associated P-values were also computed to see if there is any significant increase/decrease in accuracy. It can be seen that the rule performance after MBRL system learning had a noticeable improvement on both the training set and the test set. Although the number of rules reduced and the number of antecedents per rule increased after MBRL learning, these differences are not statistically significant.

<table>
<thead>
<tr>
<th></th>
<th>ReFuNN</th>
<th>Market-based evolution starting with the ReFuNN-generated rules</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acc. on the training set (%)</td>
<td>77.00(14.8)</td>
<td>93.11(8.3)</td>
<td>0.020</td>
</tr>
<tr>
<td>Acc. on the test set (%)</td>
<td>76.14(12.1)</td>
<td>91.41(3.8)</td>
<td>0.038</td>
</tr>
<tr>
<td>Number of rules</td>
<td>3.71(0.756)</td>
<td>3.37(0.482)</td>
<td>0.373</td>
</tr>
<tr>
<td>Number of antecedents per rule</td>
<td>3.40(0.801)</td>
<td>4.20(1.161)</td>
<td>0.237</td>
</tr>
</tbody>
</table>
Market-based rule evolution starting with the NeuroLinear-generated rules

Table 8.20 gives a comparison of accuracy, number of rules, number of antecedents per rule, and their associated P-values for the NeuroLinear-generated rules before and after using MBRL system learning. It is shown that the MBRL-refined rules achieved an accuracy of 98.15% on the training set and 96% on the test set, which are significant increases on both the training and test set when compared with the original NeuroLinear rules. In terms of rule complexity, there is no significant difference before and after MBRL system learning.

Table 8.20. A comparison of accuracy, number of rules, number of antecedents per rule, and their associated P-values for the NeuroLinear-generated rules before and after using the market-based procedure applied to the Iris data set (standard deviations are in parentheses)

<table>
<thead>
<tr>
<th></th>
<th>NeuroLinear</th>
<th>Market-based evolution starting with the NeuroLinear-generated rules (GA operated on rule base 1 and rule base 2)</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acc. on the training set (%)</td>
<td>96.67(0.8)</td>
<td>98.15(0.8)</td>
<td>0.002</td>
</tr>
<tr>
<td>Acc. on the test set (%)</td>
<td>92.00(6.9)</td>
<td>96.00(4.7)</td>
<td>0.024</td>
</tr>
<tr>
<td>Number of rules</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>3.70(0.483)</td>
<td>3.60(0.516)</td>
<td>0.343</td>
</tr>
<tr>
<td>II</td>
<td>3.30(0.675)</td>
<td>3.00(0.000)</td>
<td>0.193</td>
</tr>
<tr>
<td>Number of antecedents per rule</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>1.00(0.000)</td>
<td>1.00(0.000)</td>
<td>-</td>
</tr>
<tr>
<td>II</td>
<td>1.70(0.483)</td>
<td>1.70(0.483)</td>
<td>-</td>
</tr>
</tbody>
</table>

Market-based rule evolution starting with no prior knowledge

See Experiment 3 in Section 7.3.4 of Chapter 7.

8.4.2 Pima Indians Diabetes Data

Market-based rule evolution starting with the ReFuNN-generated fuzzy rules

A comparison of accuracy, number of rules, and number of antecedents per rule for the ReFuNN-generated rules before and after using the MBRL system learning is provided in Table 8.21. Their associated P-values were also computed to see if there is any significant increase or decrease in terms of accuracy and rule complexity. It is shown that the rule performance after MBRL system
learning had significant improvement on both the training set and the test set. In addition, the number of rules was significantly reduced after MBRL system learning.

Table 8.21. In connection with the Pima Indians Diabetes data set, a comparison of accuracy performance, number of rules, number of antecedents per rule, and their associated P-values for the ReFuNN-generated rules before and after using the market-based procedure (standard deviations are in parentheses)

<table>
<thead>
<tr>
<th></th>
<th>ReFuNN</th>
<th>Market-based evolution starting with the ReFuNN-generated rules</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acc. on the training set (%)</td>
<td>51.00(6.3)</td>
<td>74.02(2.3)</td>
<td>0.000</td>
</tr>
<tr>
<td>Acc. on the test set (%)</td>
<td>48.25(0.5)</td>
<td>72.21(2.3)</td>
<td>0.000</td>
</tr>
<tr>
<td>Number of rules</td>
<td>10.00(0.000)</td>
<td>3.25(0.500)</td>
<td>0.000</td>
</tr>
<tr>
<td>Number of antecedents per rule</td>
<td>8.25(1.140)</td>
<td>8.08(0.361)</td>
<td>0.435</td>
</tr>
</tbody>
</table>

• Market-based rule evolution starting with the NeuroLinear-generated rules

Table 8.22 provides a comparison of accuracy, number of rules, number of antecedents per rule, and their associated P-values for the NeuroLinear-generated rules before and after using the MBRL system learning. It can be seen that a significant improvement in accuracy was demonstrated on both the training set and the test set. In terms of number of rules, there was a significant reduction on the second-level rules after the MBRL learning. Although an increase on the number of first-level rules was observed, it is not statistically significant.
Table 8.22. In connection with the Pima Indians Diabetes data set, a comparison of accuracy, number of rules, number of antecedents per rule, and their associated P-values for the NeuroLinear-generated rules before and after using the market-based procedure (standard deviations are in parentheses)

<table>
<thead>
<tr>
<th></th>
<th>NeuroLinear</th>
<th>Market-based evolution starting with the NeuroLinear-generated rules (GA operated on rule base 1 and rule base 2)</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acc. on the training set (%)</td>
<td>77.10(1.4)</td>
<td>79.27(1.4)</td>
<td>0.010</td>
</tr>
<tr>
<td>Acc. on the test set (%)</td>
<td>71.10(3.4)</td>
<td>75.29(2.9)</td>
<td>0.006</td>
</tr>
<tr>
<td>Number of rules</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>3.67(1.580)</td>
<td>6.25(2.060)</td>
<td>0.079</td>
</tr>
<tr>
<td>II</td>
<td>7.33(3.840)</td>
<td>4.78(3.110)</td>
<td>0.000</td>
</tr>
<tr>
<td>Number of antecedents per rule</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>1.00(0.000)</td>
<td>1.00(0.000)</td>
<td>-</td>
</tr>
<tr>
<td>II</td>
<td>1.56(0.527)</td>
<td>1.56(0.527)</td>
<td>-</td>
</tr>
</tbody>
</table>

• Market-based rule evolution starting with no prior knowledge

With the MBRL system starting with randomly generated rules, the rule performance, number of rules, and associated P-values are provided in Table 8.23. It can be seen that the MBRL system was capable of finding a small set of rules to yield classification solutions.

Table 8.23. In connection with the Pima Indians Diabetes data set, a comparison of accuracy, number of rules, and their associated P-values for the randomly generated rules before and after using the market-based procedure (standard deviations are in parentheses)

<table>
<thead>
<tr>
<th></th>
<th>Randomly-generated rules</th>
<th>Market-based evolution starting with the randomly-generated rules (GA operated on rule base 1 and rule base 2)</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acc. on the training set (%)</td>
<td>0.00(0.000)</td>
<td>67.19(3.8)</td>
<td>0.000</td>
</tr>
<tr>
<td>Acc. on the test set (%)</td>
<td>0.00(0.000)</td>
<td>67.17(1.9)</td>
<td>0.000</td>
</tr>
<tr>
<td>Number of rules</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>50.00(0.000)</td>
<td>3.25(1.500)</td>
<td>0.000</td>
</tr>
<tr>
<td>II</td>
<td>30.00(0.000)</td>
<td>2.50(1.000)</td>
<td>0.000</td>
</tr>
</tbody>
</table>
8.4.3 Wine Recognition Data

- **Market-based rule evolution starting with the ReFuNN-generated fuzzy rules**

Table 8.24 gives a comparison of accuracy, number of rules, number of antecedents per rule, and their associated P-values for the ReFuNN-generated rules before and after using MBRL system learning. It can be seen that the accuracy performance of market-based evolved fuzzy rules has shown a significant improvement, and the number of rules has been decreased after market-based learning. However, a significant increase in the number of antecedents per rule has also been observed.

| Table 8.24. In connection with the Wine Recognition data set, a comparison of accuracy, number of rules, number of antecedents per rule, and their associated P-values for the ReFuNN-generated rules before and after using the market-based procedure (standard deviations are in parentheses) |
|---|---|---|---|
| | ReFuNN | Market-based evolution starting with the ReFuNN-generated rules | P-value |
| Acc. on the training set (%) | 76.25(0.0) | 91.75(10.1) | 0.026 |
| Acc. on the test set (%) | 70.00(6.1) | 91.10(10.1) | 0.021 |
| Number of rules | 6.00(0.000) | 3.80(0.447) | 0.000 |
| Number of antecedents per rule | 4.12(1.150) | 9.02(0.022) | 0.022 |

- **Market-based rule evolution starting with the NeuroLinear-generated rules**

A comparison of accuracy, number of rules, number of antecedents per rule, and their associated P-values for the NeuroLinear-generated rules before and after using the MBRL system learning is summarized in Table 8.25. The MBRL system did not significantly improve the predictive accuracy (accuracy on the test set) of NeuroLinear-generated rules. Although there were no significant improvements in the test accuracy rate, there was an evident reduction in the number of rules on the second-level rule sets.
Table 8.25. In connection with the Wine Recognition data set, a comparison of accuracy, number of rules, number of antecedents per rule, and their associated P-values for the NeuroLinear-generated rules before and after using the market-based procedure (standard deviations are in parentheses)

<table>
<thead>
<tr>
<th></th>
<th>NeuroLinear</th>
<th>Market-based evolution starting with the NeuroLinear-generated rules (GA operated on rule base 1 and rule base 2)</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acc. on the training set (%)</td>
<td>98.53(0.4) &lt;br&gt; 99.76(0.3)</td>
<td>0.000</td>
<td></td>
</tr>
<tr>
<td>Acc. on the test set (%)</td>
<td>93.09(9.4) &lt;br&gt; 94.80(7.5)</td>
<td>0.080</td>
<td></td>
</tr>
<tr>
<td>Number of rules I</td>
<td>4.40(0.966) &lt;br&gt; 4.20(1.033)</td>
<td>0.168</td>
<td></td>
</tr>
<tr>
<td>Number of antecedents per rule II</td>
<td>4.20(1.135) &lt;br&gt; 3.20(0.635)</td>
<td>0.008</td>
<td></td>
</tr>
<tr>
<td>Number of antecedents per rule II</td>
<td>1.00(0.000) &lt;br&gt; 1.00(0.000)</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Number of antecedents per rule II</td>
<td>2.00(0.000) &lt;br&gt; 2.00(0.000)</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

*Market-based rule evolution starting with no prior knowledge*

Starting with randomly generated rules, the rule performance, number of rules, and associated P-values before and after MBRL system learning are presented in Table 8.26. The learning ability of the MBRL system as an independent learning model was evidenced. Starting with the randomly generated rule sets, the MBRL system was capable of extracting small rule sets that achieved an accuracy that was comparable to the other rule extraction techniques.
Table 8.26. In connection with the Wine Recognition data set, a comparison of accuracy, number of rules, and their associated P-values for the randomly generated rules before and after using the market-based procedure (standard deviations are in parentheses)

<table>
<thead>
<tr>
<th></th>
<th>Randomly-generated rules</th>
<th>Market-based evolution starting with the randomly-generated rules (GA operated on rule base 1 and rule base 2)</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acc. on the training set (%)</td>
<td>0.00(0.000)</td>
<td>72.29(5.7)</td>
<td>0.000</td>
</tr>
<tr>
<td>Acc. on the test set (%)</td>
<td>0.00(0.000)</td>
<td>70.62(6.8)</td>
<td>0.000</td>
</tr>
<tr>
<td>Number of rules</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>50.00(0.000)</td>
<td>4.50(1.105)</td>
<td>0.000</td>
</tr>
<tr>
<td>II</td>
<td>30.00(0.000)</td>
<td>3.20(1.120)</td>
<td>0.000</td>
</tr>
</tbody>
</table>

8.4.4 Golf Course Problem

- Market-based rule Evolution starting with the ReFuNN-generated fuzzy rules

For the Golf Course Problem, a comparison of accuracy, number of rules, number of antecedents per rule, and their associated P-values for the ReFuNN-generated rules before and after using the market-based procedure is summarized in Table 8.27. It can be seen that there was a 7.4% mean accuracy increase on the test set after the market-based procedure was applied to the ReFuNN-generated fuzzy rules.

Table 8.27. In connection with the Golf Course Problem, a comparison of accuracy, number of rules, number of antecedents per rule, and their associated P-values for the ReFuNN-generated rules before and after using the market-based procedure (standard deviations are in parentheses)

<table>
<thead>
<tr>
<th></th>
<th>ReFuNN</th>
<th>Market-based evolution starting with the ReFuNN-generated rules</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acc. on the training set (%)</td>
<td>62.60(0.5)</td>
<td>67.60(2.0)</td>
<td>0.007</td>
</tr>
<tr>
<td>Acc. on the test set (%)</td>
<td>60.40(4.8)</td>
<td>67.00(10.2)</td>
<td>0.045</td>
</tr>
<tr>
<td>Number of rules</td>
<td></td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>Number of antecedents per rule</td>
<td>7.20(0.023)</td>
<td>7.36(0.275)</td>
<td>0.459</td>
</tr>
</tbody>
</table>
**Market-based Rule evolution starting with the NeuroLinear-generated rules**

Table 8.28 provides a comparison of accuracy, number of rules, number of antecedents per rule, and their associated P-values for the NeuroLinear-generated rules before and after using MBRL system learning. The MBRL system significantly improved the accuracy of rules on both the training set and test set. In addition, the number of rules on the second-level of rule sets was significantly reduced by the MBRL system.

**Table 8.28.** In connection with the Golf Course Problem, a comparison of accuracy, number of rules, number of antecedents per rule, and their associated P-values for the NeuroLinear-generated rules before and after using the market-based procedure (standard deviations are in parentheses)

<table>
<thead>
<tr>
<th></th>
<th>NeuroLinear</th>
<th>Market-based evolution starting with the NeuroLinear-generated rules (GA operated on rule base 1 and rule base 2)</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acc. on the training set (%)</td>
<td>70.46(1.9)</td>
<td>76.39(2.0)</td>
<td>0.016</td>
</tr>
<tr>
<td>Acc. on the test set (%)</td>
<td>70.16(3.4)</td>
<td>75.11(2.8)</td>
<td>0.020</td>
</tr>
<tr>
<td>Number of rules I</td>
<td>9.50(1.380)</td>
<td>9.50(1.380)</td>
<td>-</td>
</tr>
<tr>
<td>Number of rules II</td>
<td>24.50(7.31)</td>
<td>11.67(4.73)</td>
<td>0.019</td>
</tr>
<tr>
<td>Number of antecedents per rule I</td>
<td>1.00(0.000)</td>
<td>1.00(0.000)</td>
<td>-</td>
</tr>
<tr>
<td>Number of antecedents per rule II</td>
<td>3.50(0.548)</td>
<td>3.50(0.548)</td>
<td>-</td>
</tr>
</tbody>
</table>

**Market-based rule evolution starting with no prior knowledge**

Starting with randomly generated rules, the rule performance, number of rules, and associated P-values before and after MBRL system learning are presented in Table 8.29. The learning ability of the MBRL system as an independent learning model was again evidenced.
Table 8.29. In connection with the Golf Course Problem, a comparison of accuracy, number of rules, and their associated P-values for the randomly generated rules before and after using the market-based procedure (standard deviations are in parentheses)

<table>
<thead>
<tr>
<th></th>
<th>Randomly-generated rules</th>
<th>Market-based evolution starting with the randomly-generated rules (GA operated on rule base 1 and rule base 2)</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acc. on the training set (%)</td>
<td>0.00(0.000)</td>
<td>62.30(6.3)</td>
<td>0.000</td>
</tr>
<tr>
<td>Acc. on the test set (%)</td>
<td>0.00(0.000)</td>
<td>61.20(4.2)</td>
<td>0.000</td>
</tr>
<tr>
<td>Number of rules I</td>
<td>50.00(0.000)</td>
<td>9.40(0.125)</td>
<td>0.000</td>
</tr>
<tr>
<td>Number of rules II</td>
<td>30.00(0.000)</td>
<td>7.83(0.094)</td>
<td>0.000</td>
</tr>
</tbody>
</table>

8.4.5 Cook Islands Sea Cucumber Habitat Data

*Market-based rule evolution starting with the ReFuNN-generated fuzzy rules*

Table 8.30 provides a comparison of accuracy, number of rules, number of antecedents per rule, and their associated P-values for the ReFuNN-generated rules before and after using MBRL system learning. It can be seen that the accuracy performance of MBRL-refined fuzzy rules has shown a significant improvement (over 20%). The number of fuzzy rules was also significantly reduced. Although the number of antecedents per rule increased after the MBRL system, it is not statistically significant.

Table 8.30. In connection with the Cook Islands Sea Cucumber Habitat data set, a comparison of accuracy, number of rules, number of antecedents per rule, and their associated P-values for the ReFuNN-generated rules before and after using the market-based procedure (standard deviations are in parentheses)

<table>
<thead>
<tr>
<th></th>
<th>ReFuNN</th>
<th>Market-based evolution starting with the ReFuNN-generated rules</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acc. on the training set (%)</td>
<td>69.38(9.4)</td>
<td>84.62(0.0)</td>
<td>0.000</td>
</tr>
<tr>
<td>Acc. on the test set (%)</td>
<td>60.44(10.4)</td>
<td>80.87(0.0)</td>
<td>0.000</td>
</tr>
<tr>
<td>Number of rules</td>
<td>6.80(1.789)</td>
<td>2.00(0.000)</td>
<td>0.004</td>
</tr>
<tr>
<td>Number of antecedents per rule</td>
<td>5.76(0.220)</td>
<td>9.30(3.38)</td>
<td>0.077</td>
</tr>
</tbody>
</table>
• Market-based rule evolution starting with the NeuroLinear-generated rules

A comparison of accuracy performance, number of rules, number of antecedents per rule, and their associated P-values for the NeuroLinear-generated rules before and after using MBRL system learning is summarized in Table 8.31. The MBRL system significantly reduced the number of rules on the second level.

Table 8.31. In connection with the Cook Islands Sea Cucumber data set, a comparison of accuracy, number of rules, number of antecedents per rule, and their associated P-values for the NeuroLinear-generated rules before and after using the market-based procedure (standard deviations are in parentheses)

<table>
<thead>
<tr>
<th></th>
<th>NeuroLinear</th>
<th>Market-based evolution starting with the NeuroLinear-generated rules (GA operated on rule base 1 and rule base 2)</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acc. on the training set (%)</td>
<td>90.16(1.6)</td>
<td>91.30(1.2)</td>
<td>0.152</td>
</tr>
<tr>
<td>Acc. on the test set (%)</td>
<td>82.69(8.1)</td>
<td>84.89(5.5)</td>
<td>0.134</td>
</tr>
<tr>
<td>Number of rules</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>3.75(2.870)</td>
<td>3.00(2.160)</td>
<td>0.215</td>
</tr>
<tr>
<td>II</td>
<td>4.44(2.404)</td>
<td>3.22(1.202)</td>
<td>0.030</td>
</tr>
<tr>
<td>Number of antecedents per rule</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>1.00(0.000)</td>
<td>1.00(0.000)</td>
<td>-</td>
</tr>
<tr>
<td>II</td>
<td>1.70(0.483)</td>
<td>1.70(0.483)</td>
<td>-</td>
</tr>
</tbody>
</table>

• Market-based rule evolution starting with no prior knowledge

The performance of the rule sets produced by the market-based evolution starting with no prior knowledge is summarized in Table 8.32. It can be seen that, as an independent learning model starting with no prior knowledge, the MBRL system’s learning ability is comparable to those of other rule learning and refinement techniques.
Table 8.32. In connection with the Cook Islands Sea Cucumber data set, a comparison of accuracy performance, number of rules, and their associated P-values for the randomly generated rules before and after using the market-based procedure (standard deviations are in parentheses)

<table>
<thead>
<tr>
<th></th>
<th>Randomly-generated rules</th>
<th>Market-based evolution starting with the randomly-generated rules (GA operated on rule base 1 and rule base 2)</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acc. on the training set (%)</td>
<td>0.00(0.000)</td>
<td>85.91(4.7)</td>
<td>0.000</td>
</tr>
<tr>
<td>Acc. on the test set (%)</td>
<td>0.00(0.000)</td>
<td>83.08(3.4)</td>
<td>0.000</td>
</tr>
<tr>
<td>Number of rules</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>50.00(0.000)</td>
<td>3.00(1.414)</td>
<td>0.000</td>
</tr>
<tr>
<td>II</td>
<td>30.00(0.000)</td>
<td>2.60(0.894)</td>
<td>0.000</td>
</tr>
</tbody>
</table>

• Market-based rule evolution starting with the equation produced by Logistic Regression

The purpose of logistic regression is to establish a quantitative relationship between a group of predictor variables (inputs) and a response (output). This quantitative relationship can be transformed to NeuroLinear-formatted rules in a straightforward manner. Then, the MBRL system can be used to discover evolved rules that represent better knowledge of the relationship between input variables and output.

We employed the standard statistical approach of backward stepwise logistic regression (Dowdy & Wearden 1991) in this experiment. This approach begins with all of the variables in the model and then removes the least significant terms until all remaining variables are statistically significant. An important assumption underlying the procedure is that some of the input variables do not have an important explanatory effect on the outcome. If this assumption is true, then it is presumed to be valid to leave only the significant variables in the model.

After examining the raw data for all 128 sites, it was found that the attribute gravel and the attribute mud/silt contained very few non-zero values: gravel had only two and mud/silt had only five non-zero values, and where these two attributes were present in the raw data, they often existed in very high percentages. It was thought that the inclusion of these variables in the starting model had a negative influence on the regression. So we removed the gravel and mud/silt attributes from the starting model.
The following equation was produced by the Logistic Regression method:

Probability of a site belonging to class 1
= logit (0.114%Rubble + 0.104%Consolidated Rubble + 0.063%Rock/Pavement + 0.131%Live coral + 0.080%Dead coral - 9.994)

where: logit (x) = e^x / (1 + e^x)

This equation made correct predictions for 87% of the 128 sites.

The above equation can be transformed to the following equivalent rules:

1. If 0.114%Rubble + 0.104%Consolidated Rubble + 0.063%Rock/Pavement + 0.131%Live coral + 0.080%Dead coral - 9.994 < 0, then class = 0,
2. If 0.114%Rubble + 0.104%Consolidated Rubble + 0.063%Rock/Pavement + 0.131%Live coral + 0.080%Dead coral - 9.994 > 0, then class = 1.

Based on the above two rules, the two levels of NeuroLinear-type of rules can be created as shown below:

The first-level rule set:

1. If 0.114%Rubble + 0.104%Consolidated Rubble + 0.063%Rock/Pavement + 0.131%Live coral + 0.080%Dead coral - 9.994 < 0, then the discrete intermediate output = 1,
2. If 0.114%Rubble + 0.104%Consolidated Rubble + 0.063%Rock/Pavement + 0.131%Live coral + 0.080%Dead coral - 9.994 > 0, then the discrete intermediate output = 2.

The second-level rule set:

1. If the discrete intermediate output = 1, then class = 0,
2. If the discrete intermediate output = 2, then class = 1.

The MBRL system was then run on the above rule sets. One representative rule set after MBRL system learning is shown as follows (the rule components that have been modified by the market-based procedure are shown in underlined text):
1. If 
   \[-0.598 \text{Exposure} - 0.594 \% \text{Sand} + 0.114 \% \text{Rubble} + 0.104 \% \text{Consolidated Rubble} + 0.063 \% \text{Rock/Pavement} + 0.131 \% \text{Live coral} + 0.080 \% \text{Dead coral} - 10.483 < 0,\]
   then class = 0,
2. Default rule (class = 1).

Thus in connection with the Logistic Regression method, the following equation was generated after applying the market-based learning:

\[
\text{Probability of a site belonging to class 1} = \logit (-0.598 \text{Exposure} - 0.594 \% \text{Sand} + 0.114 \% \text{Rubble} + 0.104 \% \text{Consolidated Rubble} + 0.063 \% \text{Rock/Pavement} + 0.131 \% \text{Live coral} + 0.080 \% \text{Dead coral} - 10.483)
\]

where: \( \logit (x) = \frac{e^x}{1 + e^x} \)

The new equation made correct predictions for 91.4% of the total 128 sites, which was a more than 4% increase over the original equation produced by the Logistic Regression method.

### 8.4.6 New Zealand Asthma Incidence Data

- **Market-based rule evolution starting with the ReFuNN-generated fuzzy rules**

Table 8.33 shows results of the MBRL system used with rules generated by ReFuNN. The P-values associated with accuracy, number of rules, and number of antecedents per rule before and after using the market-based procedure is also presented in the table. It can be seen that there was an accuracy performance increase on both the training set and the test set after the market-based procedure was applied to the ReFuNN-generated fuzzy rules.
Table 8.33. In connection with the New Zealand Asthma Incidence data set, a comparison of accuracy, number of rules, number of antecedents per rule, and their associated P-values for the ReFuNN-generated rules before and after using the market-based procedure (standard deviations are in parentheses)

<table>
<thead>
<tr>
<th></th>
<th>ReFuNN</th>
<th>Market-based evolution starting with the ReFuNN-generated rules</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acc. on the training set (%)</td>
<td>50.30(0.8)</td>
<td>53.30(1.4)</td>
<td>0.011</td>
</tr>
<tr>
<td>Acc. on the test set (%)</td>
<td>49.90(0.6)</td>
<td>52.30(0.8)</td>
<td>0.012</td>
</tr>
<tr>
<td>Number of rules</td>
<td>10.00(0.000)</td>
<td>10.00(0.000)</td>
<td>-</td>
</tr>
<tr>
<td>Number of antecedents per rule</td>
<td>8.20(0.003)</td>
<td>8.43(0.075)</td>
<td>0.738</td>
</tr>
</tbody>
</table>

**Market-based rule evolution starting with the NeuroLinear-generated rules**

Table 8.34 shows results of the MBRL system used with rules generated by NeuroLinear. The associated P-values is also presented in the table. It can be seen that a significant improvement in accuracy was demonstrated on both the training set and the test set. In terms of number of rules, there was a significant reduction in the second-level rule set after MBRL learning.
Table 8.34. In connection with the New Zealand Asthma Incidence data set, a comparison of accuracy, number of rules, number of antecedents per rule, and their associated P-values for the NeuroLinear-generated rules before and after using the market-based procedure (standard deviations are in parentheses)

<table>
<thead>
<tr>
<th></th>
<th>NeuroLinear</th>
<th>Market-based evolution starting with the NeuroLinear-generated rules (GA operated on rule base 1 and rule base 2)</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acc. on the training set (%)</td>
<td>57.70(3.3)</td>
<td>60.20(6.3)</td>
<td>0.000</td>
</tr>
<tr>
<td>Acc. on the test set (%)</td>
<td>52.70(3.3)</td>
<td>57.10(1.0)</td>
<td>0.000</td>
</tr>
<tr>
<td>Number of rules I</td>
<td>6.70(0.383)</td>
<td>6.67(0.082)</td>
<td>0.722</td>
</tr>
<tr>
<td>Number of rules II</td>
<td>16.30(0.575)</td>
<td>8.21(0.031)</td>
<td>0.000</td>
</tr>
<tr>
<td>Number of antecedents per rule I</td>
<td>1.00(0.000)</td>
<td>1.00(0.000)</td>
<td>-</td>
</tr>
<tr>
<td>Number of antecedents per rule II</td>
<td>3.70(0.283)</td>
<td>3.70(0.283)</td>
<td>-</td>
</tr>
</tbody>
</table>

• Market-based rule evolution starting with no prior knowledge
The performance of the rule sets produced by the market-based evolution starting with no prior knowledge is summarized in Table 8.35. The learning ability of the MBRL system as an independent learning model is again demonstrated. Starting with no prior knowledge, the MBRL system achieved an accuracy that was comparable to other knowledge discovery and refinement techniques.
Table 8.35. In connection with the New Zealand Asthma Incidence data set, a comparison of accuracy, number of rules, and their associated P-values for the randomly generated rules before and after using the market-based procedure (standard deviations are in parentheses)

<table>
<thead>
<tr>
<th></th>
<th>Randomly-generated rules</th>
<th>Market-based evolution starting with the randomly-generated rules (GA operated on rule base 1 and rule base 2)</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acc. on the training set (%)</td>
<td>0.00(0.000)</td>
<td>53.89(10.2)</td>
<td>0.000</td>
</tr>
<tr>
<td>Acc. on the test set (%)</td>
<td>0.00(0.000)</td>
<td>52.78(0.5)</td>
<td>0.000</td>
</tr>
<tr>
<td>Number of rules I</td>
<td>50.00(0.000)</td>
<td>6.20(0.023)</td>
<td>0.000</td>
</tr>
<tr>
<td>Number of rules II</td>
<td>30.00(0.000)</td>
<td>5.35(0.168)</td>
<td>0.000</td>
</tr>
</tbody>
</table>

8.5 Empirical Evaluation

The preceding sections described various experiments conducted with the market-based rule learning system and four existing rule extraction methods: C4.5rules, X2R, ReFuNN and NeuroLinear. In order to evaluate the performance of the rules and related techniques, this section presents a collective analysis based on the experimental results.

Chapter 1 introduced the dimensions along which rule extraction and refinement techniques should be evaluated: predictive accuracy, comprehensibility, fidelity, and consistency. In addition to these criteria, there are other criteria that are often important considerations: scalability and computational efficiency.

8.5.1 Accuracy

In this context a rule set is considered to be accurate if it can correctly classify a set of previously unseen examples from the problem domain. Table 8.36 provides a summary of the mean accuracy performance on both the training set and the test set (unseen examples) for each rule extraction technique. In Table 8.36, the first number in the cells represents the mean accuracy performance on the training set, the second number in the cells represents the mean accuracy performance on the test set, and the symbol ‘*’ marks results in cases where the test accuracy of the MBRL refined rules is significantly superior to the accuracy of the ReFuNN-generated rules and NeuroLinear-generated rules at the 95% confidence level.
Table 8.36. Accuracy performance (%) of the rule sets extracted by different rule extraction techniques

<table>
<thead>
<tr>
<th>Acc. on the training set/Acc. on the test set</th>
<th>C4.5Rules</th>
<th>X2R</th>
<th>ReFuNN</th>
<th>Market-based evolution starting with the ReFuNN-generated rules</th>
<th>NeuroLinear</th>
<th>Market-based evolution starting with the NeuroLinear-generated rules (GA operated on rule base 1 and rule base 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iris Data</td>
<td>97.47/95.98</td>
<td>94.52/87.99</td>
<td>77.00/76.14</td>
<td>* 93.11/91.41</td>
<td>96.67/92.00</td>
<td>* 98.15/96.00</td>
</tr>
<tr>
<td>Pima Indians Diabetes Data</td>
<td>82.88/74.20</td>
<td>62.70/61.00</td>
<td>51.00/48.25</td>
<td>* 74.02/72.21</td>
<td>77.10/71.10</td>
<td>* 79.27/75.29</td>
</tr>
<tr>
<td>Wine Recognition Data</td>
<td>98.90/92.73</td>
<td>90.20/83.70</td>
<td>76.25/70.00</td>
<td>* 91.75/91.10</td>
<td>98.53/93.09</td>
<td>99.76/94.80</td>
</tr>
<tr>
<td>Golf Course Problem</td>
<td>74.51/70.67</td>
<td>73.00/62.87</td>
<td>62.60/60.40</td>
<td>* 67.60/67.00</td>
<td>70.46/70.16</td>
<td>* 76.39/75.11</td>
</tr>
<tr>
<td>Cook Islands Sea Cucumber Data</td>
<td>92.87/82.68</td>
<td>84.20/80.90</td>
<td>69.38/60.44</td>
<td>* 84.62/80.87</td>
<td>90.16/82.69</td>
<td>91.30/84.89</td>
</tr>
<tr>
<td>New Zealand Asthma Incidence Data</td>
<td>57.97/50.90</td>
<td>53.20/50.50</td>
<td>50.30/49.90</td>
<td>* 53.30/52.30</td>
<td>57.70/52.70</td>
<td>* 60.20/57.10</td>
</tr>
</tbody>
</table>

From Table 8.36 and the statistical significance analyses shown in relevant tables of the previous two sections, it can be seen that:

(1) The fuzzy rules refined by the MBRL system had a performance that was significantly better than the ReFuNN-generated fuzzy rules for all six problem domains. For application to the Pima Indians Diabetes Data, the Wine Recognition Data, and the Cook Islands Sea Cucumber Data, there was a greater than 20% accuracy increase on the test sets for the market-based procedure. For the Iris Data, and the Golf course Problem, the performance of market-based evolved fuzzy rules also showed a great improvement. In only one case, the New Zealand Asthma Incidence Data, did the market-based evolved fuzzy rules have less than a 5% increase in accuracy when compared with the original ReFuNN-generated fuzzy rules.
(2) When applying the GA on both of the two levels of rule sets (rule base 1 and rule base 2) associated with the NeuroLinear approach, the accuracy performance of the rules is significantly improved on the test sets of four of the problem domains after market-based learning has been conducted. Only for the Wine Recognition Data and the Cook Islands Sea Cucumber Data is there no significant difference in the mean accuracy rates before and after MBRL system learning. Notwithstanding the two cases that did show statistically significant improvement, in all six problem domains, the MBRL-refined rules had a performance that was superior to the original NeuroLinear-generated rules.

(3) In comparing the market-based refined NeuroLinear rules to the extracted rules from traditional rule extraction method C4.5rules, for three problem domains: Wine Recognition Data (P-value = 0.050), Golf Course Problem (P-value = 0.010), and New Zealand Asthma Incidence Data (P-value = 0.000), the market-based refined NeuroLinear rules were significantly more accurate than the C4.5-generated rules.

The X2R-generated rules performed well on smaller data sets such as the Cook Islands Sea Cucumber Data. However when X2R was applied to some larger data sets, its accuracy did not match that of other rule extraction methods e.g. NeuroLinear.

(4) There was a relatively poor predictive performance observed for all six rule extraction techniques when they were applied to the New Zealand Asthma Incidence data set (57.10% was the best result on the test set). This is probably intrinsic to the data set and may be due to noisy features in the data or insufficient (missing) input variables associated with the occurrence of asthma (Hales et al. 1998a).

Table 8.37 presents a comparison of accuracy performance of the rule sets before and after the MBRL system was applied and started without prior knowledge. It can be seen that a significant improvement was achieved on all six applications after the market-based learning was employed. By comparing Table 8.37 with Table 8.36, it can be seen that, as an independent learning model, the MBRL system’s learning ability is comparable to those of other techniques.
Table 8.37. Accuracy(%) of the rule sets before and after the market-based rule learning system was applied and started without prior knowledge

<table>
<thead>
<tr>
<th>Acc. on training set/Acc. on test set</th>
<th>Before</th>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iris Data</td>
<td>0.00/0.00</td>
<td>96.80/93.32</td>
</tr>
<tr>
<td>Pima Indians Diabetes Data</td>
<td>0.00/0.00</td>
<td>67.19/67.17</td>
</tr>
<tr>
<td>Wine Recognition Data</td>
<td>0.00/0.00</td>
<td>72.29/70.62</td>
</tr>
<tr>
<td>Golf Course Problem</td>
<td>0.00/0.00</td>
<td>62.30/61.20</td>
</tr>
<tr>
<td>Cook Islands Sea Cucumber Data</td>
<td>0.00/0.00</td>
<td>85.91/83.08</td>
</tr>
<tr>
<td>New Zealand Asthma Incidence Data</td>
<td>0.00/0.00</td>
<td>53.89/52.78</td>
</tr>
</tbody>
</table>

8.5.2 Comprehensibility
The comprehensibility of a rule set is determined by measuring the number of rules and the average number of conditions in rule antecedents. The rationale for this measure is: the smaller the number of rules, the more compact the rule set; and the smaller the number of conditions in the antecedent of a rule in the rule set, the simpler each rule is, and the more compact is the rule set. Table 8.38 shows the number of rules extracted by different rule extraction techniques. Table 8.39 presents the average number of antecedents per rule. In Table 8.38 and Table 8.39, the symbol ‘*’ marks results in cases where the number of the MBRL-refined rules and the average number of antecedents per rule is significantly different from those of the ReFuNN-generated rules and NeuroLinear-generated rules at the 95% confidence level.
Table 8.38. Number of rules extracted by different rule extraction techniques. Column I signifies *rule base 1* associated with the NeuroLinear-generated rules, and Column II signifies *rule base 2* associated with the NeuroLinear-generated rules.

<table>
<thead>
<tr>
<th>Data Set</th>
<th>C4.5Rules</th>
<th>X2R</th>
<th>ReFuNN</th>
<th>Market-based evolution starting with the ReFuNN-generated rules</th>
<th>NeuroLinear</th>
<th>Market-based evolution starting with the NeuroLinear-generated rules (GA operated on rule base 1 and rule base 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>I</td>
<td>II</td>
<td>I</td>
<td>II</td>
<td>I</td>
<td>II</td>
</tr>
<tr>
<td>Iris Data</td>
<td>5.00</td>
<td>5.60</td>
<td>3.71</td>
<td>3.37</td>
<td>3.70</td>
<td>3.30</td>
</tr>
<tr>
<td>Pima Indians Diabetes Data</td>
<td>10.70</td>
<td>38.25</td>
<td>10.00</td>
<td>* 3.25</td>
<td>3.67</td>
<td>7.33</td>
</tr>
<tr>
<td>Wine Recognition Data</td>
<td>4.20</td>
<td>7.70</td>
<td>6.00</td>
<td>* 3.80</td>
<td>4.40</td>
<td>4.20</td>
</tr>
<tr>
<td>Golf Course Problem</td>
<td>18.00</td>
<td>83.60</td>
<td>20.00</td>
<td>20.00</td>
<td>9.50</td>
<td>24.50</td>
</tr>
<tr>
<td>Cook Islands Sea Cucumber Data</td>
<td>4.30</td>
<td>6.30</td>
<td>6.80</td>
<td>* 2.00</td>
<td>3.75</td>
<td>4.44</td>
</tr>
<tr>
<td>New Zealand Asthma Incidence Data</td>
<td>10.40</td>
<td>82.00</td>
<td>10.00</td>
<td>10.00</td>
<td>6.70</td>
<td>16.30</td>
</tr>
</tbody>
</table>
Table 8.39. Average number of antecedents per rule for rule sets (excluding default rules) extracted by different rule extraction techniques. Column I signifies rule base 1 associated with the NeuroLinear-generated rules, and Column II signifies rule base 2 associated with the NeuroLinear-generated rules.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>C4.5Rules</th>
<th>X2R</th>
<th>ReFuNN</th>
<th>Market-based evolution starting with the ReFuNN-generated rules</th>
<th>NeuroLinear</th>
<th>Market-based evolution starting with the NeuroLinear-generated rules</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>GA operated on rule base 1 and rule base 2</td>
</tr>
<tr>
<td>Iris Data</td>
<td>1.45</td>
<td>1.79</td>
<td>3.40</td>
<td>4.20</td>
<td>1.00</td>
<td>1.70</td>
</tr>
<tr>
<td>Pima Indians Diabetes Data</td>
<td>3.12</td>
<td>5.04</td>
<td>8.25</td>
<td>8.08</td>
<td>1.00</td>
<td>1.56</td>
</tr>
<tr>
<td>Wine Recognition Data</td>
<td>2.18</td>
<td>2.95</td>
<td>4.12</td>
<td>* 9.02</td>
<td>1.00</td>
<td>2.00</td>
</tr>
<tr>
<td>Golf Course Problem</td>
<td>3.73</td>
<td>3.78</td>
<td>7.20</td>
<td>7.36</td>
<td>1.00</td>
<td>3.50</td>
</tr>
<tr>
<td>Cook Islands Sea Cucumber Data</td>
<td>1.93</td>
<td>2.77</td>
<td>5.76</td>
<td>9.30</td>
<td>1.00</td>
<td>1.70</td>
</tr>
<tr>
<td>New Zealand Asthma Incidence Data</td>
<td>1.36</td>
<td>2.08</td>
<td>8.20</td>
<td>8.43</td>
<td>1.00</td>
<td>3.70</td>
</tr>
</tbody>
</table>

From above two tables and the statistical significance analyses shown in relevant tables of previous two sections, it can be seen that:

(1) For the fuzzy rules, after market-based rule refinement, the number of rules is significantly reduced for three problem domains (Pima Indians Diabetes Data, Wine Recognition Data, and Cook Islands Sea Cucumber Data). The number of fuzzy rules remains the same for two applications (Golf Course Problem and New Zealand Asthma Incidence Data), and decreases insignificantly for the Iris Data; while the average number of antecedents per rule increases for all the applications. But only for the Wine Recognition Data is the increase statistically significant.
For the NeuroLinear-generated rules, for the all six problem domains, there was a significant decrease in the number of NeuroLinear rules on the second level rule set (rule base 2) after market-based learning. The decrease can be as large as 50% (see the Golf Course Problem in Table 8.38). For the first level rule set (rule base 1), although increases or decreases were observed after the MBRL system learning on different data sets, the differences were not statistically significant. Table 8.39 shows that, for the all six applications, no differences were observed in terms of the average number of antecedents per rule for the two levels of NeuroLinear rules.

As already discussed in Chapter 2, instead of using multiple levels of rules, sometimes it is more convenient to use a single level rule set which maps input values to output values directly. The single level NeuroLinear rules can be generated by combining the two levels of individual rule sets together, and in some cases a default rule might be added. The number of rules of the single level NeuroLinear rules is determined by the number of rules on the second level rule set. In other words, if the number of rules on the second level rule set is decreased by the MBRL system, the number of rules of the single level rule set is guaranteed to decrease.

In summary, from the empirical observation of considering the number of rules and the average number of antecedents per rule, it is plausible to conclude that the MBRL system, as a post-processing tool, is capable of improving the compactness of ReFuNN-generated fuzzy rules and NeuroLinear rules.

It is important to emphasize that this thesis does not claim that the MBRL system can make the refined rules generally comprehensible to everyone. This is similar to the situation for the NeuroLinear-formatted rules expressed as constraint equations that involve a set of multiplicative coefficients which may or may not appear to be more comprehensible, according to one’s tastes. Rules generated by the C4.5rules method or the X2R method do not involve any weight coefficient. However, one should not conclude that such rules are always more comprehensible than rules with coefficients. When the decision boundaries in the feature parameter space are oblique (i.e. not orthogonal), simple rules with coefficients are likely to be more meaningful than a large set of rules that divide the decision regions into many small rectangular regions. The
coefficients of a rule may also reveal more about the data in hand, for example, they may identify which attribute is the most influential factor in determining a certain class label.

The MBRL system represents an effort to make a step forward in reducing the complexity (number of rules and number of antecedents per rule) of existing rule sets, which may help human experts in problem domains to inspect and understand the inference rules more easily.

8.5.3 Fidelity
A rule set is considered to display a high level of fidelity if it can mimic the behaviour of the learning model, such as an artificial neural network, from which it was extracted by capturing all of the information embodied in the model. In other words, rule fidelity is a measure of the agreement between the classification of the test set by the learning model and by the rule set extracted from the learning model. Table 8.40 presents the mean accuracy performance (on the test set) of the fuzzy neural networks (FuNN models), the feed-forward neural networks, and of the rules extracted by the ReFuNN technique, the NeuroLinear technique, and the MBRL system. In Table 8.40, the symbol ‘*’ marks results in cases where the accuracy of the ReFuNN-generated rules is significantly inferior to the accuracy of the FuNN fuzzy neural networks, and the accuracy of the NeuroLinear-generated rules is significantly inferior to the accuracy of the feed-forward neural networks at the 95% level, respectively. Similarly, the symbol ‘●’ marks results in cases where the accuracy of the MBRL refined rules is significantly different from the accuracy of the FuNN fuzzy neural networks and the feed-forward neural networks at the 95% level, respectively.
Table 8.40. Accuracy performance (%) on the test set of the learning models and of the rules extracted by different techniques

<table>
<thead>
<tr>
<th>Dataset</th>
<th>FuNN model</th>
<th>ReFuNN</th>
<th>Market-based evolution starting with the ReFuNN-generated rules</th>
<th>Feed-forward neural network model</th>
<th>NeuroLinear</th>
<th>Market-based evolution starting with the NeuroLinear-generated rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iris Data</td>
<td>92.90</td>
<td>* 76.14</td>
<td>91.41</td>
<td>94.67</td>
<td>92.00</td>
<td>96.00</td>
</tr>
<tr>
<td>Pima Indians Diabetes Data</td>
<td>76.19</td>
<td>* 48.25</td>
<td>72.21</td>
<td>75.40</td>
<td>* 71.10</td>
<td>75.29</td>
</tr>
<tr>
<td>Wine Recognition Data</td>
<td>93.70</td>
<td>* 70.00</td>
<td>91.10</td>
<td>95.40</td>
<td>93.09</td>
<td>94.80</td>
</tr>
<tr>
<td>Golf Course Problem</td>
<td>92.00</td>
<td>* 60.40</td>
<td>67.00</td>
<td>73.39</td>
<td>70.16</td>
<td>75.11</td>
</tr>
<tr>
<td>Cook Islands Sea Cucumber Data</td>
<td>88.33</td>
<td>* 60.44</td>
<td>80.87</td>
<td>84.34</td>
<td>82.69</td>
<td>84.89</td>
</tr>
<tr>
<td>New Zealand Asthma Incidence Data</td>
<td>53.02</td>
<td>* 49.90</td>
<td>52.30</td>
<td>52.90</td>
<td>52.70</td>
<td>* 57.10</td>
</tr>
</tbody>
</table>

Table 8.40 shows that the disagreements in the predictions of the neural network and of the extracted NeuroLinear rules are relatively small (only one case, the accuracy of NeuroLinear had a significantly decrease), which points to a higher fidelity for NeuroLinear rule sets. However the results of the ReFuNN-generated rules reflect the information loss during the transmission from fuzzy neural networks (FuNN models) to simpler fuzzy rules.

After the MBRL system learning, only one application (Golf course Problem) had a accuracy performance significantly inferior to the accuracy of the FuNN model. For the other five applications, there is no significant difference in accuracy between the MBRL-refined fuzzy rules and the FuNN models. In one case (New Zealand Asthma Incidence data set) where the MBRL-refined NeuroLinear rules performed significantly superior to the feed-forward neural network model. In four cases (the Iris Data, the Golf Course Problem, the Cook Islands Sea Cucumber Data, and the New Zealand Asthma Incidence Data) the NeuroLinear rules refined by the market-based procedure achieved a better mean accuracy performance than the original feed-forward neural network learning models.
8.5.4 Consistency
Under Towell and Shavlik’s criteria (1993), an extracted rule set is deemed to be consistent if, under differing training sessions, the artificial neural network (or other learning model) generates rule sets which produce the same classification of unseen examples.

An often mentioned drawback of the neural network approach for pattern classification is its non-deterministic nature (Haykin 1999). When the network is initialized with different sets of random weights, it is very likely that the training process terminates at a different local minimum. This is also true for the market-based rule evolution and refinement approach due to the different starting parameters and GA operations. However, instead of taking this as a drawback, it can be viewed as an advantage over a deterministic approach, such as a decision tree method or X2R method. When the number of samples available for training is small and the number of attributes is relatively large (as is often the case in real applications), there may be many equally good sets of rules that achieve similar accuracy rates. Rules generated from a neural network or a MBRL system allow us to obtain these different sets of rules. Some of the rules may reveal more useful information about the data or may give more insight into the problem than other rules, and these approaches present the opportunity to make these discoveries.

8.5.5 Scalability and Computational Efficiency
In terms of scalability, as indicated in Chapter 6, the MBRL system has not been demonstrated to be generally scalable to large data sets. As a post-processing tool that refines existing rule sets, the lack of scalability of the MBRL system does not raise a particular concern for data mining applications. As discussed in the preceding chapters, the MBRL system is combined with the fuzzy neural network rule extraction algorithm, ReFuNN, to provide a general framework for fuzzy inference-based knowledge discovery. The MBRL system is also combined with the feed-forward neural network rule extraction algorithm, NeuroLinear, to provide a framework for improving the quality of rules extracted from feed-forward neural networks. Like many other neural network rule extraction techniques, ReFuNN and NeuroLinear, lack scalability. The complexities of the rule sets extracted by these techniques tend to increase with network size. Therefore, when applying the above frameworks to large data sets, sampling techniques (Domingo et al. 2002) will often have to be used to reduce the size of the problem.
Although it has been demonstrated that the MBRL system has potential to be used as an independent learning model to accomplish complex computational tasks and extract symbolic rules at the same time, its efficiency for data mining on large data sets is limited. Application of the genetic algorithm would involve long search times when the MBRL system faces huge input data sets. It is no doubt that the MBRL system would perform more efficiently and effectively after the sizes of large data sets are reduced by the relevant techniques.

There is no denying the fact that the techniques for extracting rules from artificial neural networks and refining rules by the MBRL system, which entails a network training and pruning process and/or a genetic algorithm process, is computationally more expensive than some other approaches that extract rules directly from a data set, such as C4.5rules method or the X2R rule generator. However, the expense in computing time often leads to an improvement in rule quality. For example, based on a Pentium III platform (Intel 686 processor Model 7) with 128 Mbytes of memory running the Windows NT operating system, it took about 1 second for the X2R method (a C++ implementation) to generate 36 rules (with an average of 5.94 conditions per rule) on the Pima Indians Diabetes Data with 64% accuracy on the training set and 62% accuracy on the tests set, while the feed-forward neural network model employing BFGS training and pruning algorithm (a C++ implementation) took 45 seconds to generate 6 NeuroLinear rules (with an average of 2.6 conditions per rule) with 75% accuracy on the training set and 72% on the test set. However, it took the market-based learning procedure (a Java implementation) about 2 minutes to simplify the NeuroLinear rule sets (4 rules with an average of 2 conditions per rule) with similar accuracies. This observation indicates the need for a balanced view about accuracy, simplicity and time efficiency. If the simplicity of representation improves, the loss of accuracy or computationally expense may be tolerable.

When time is particularly scarce, the decision tree or X2R procedure may be the method of choice. Otherwise, it is probably worthwhile trying other approaches because of their advantages (more compact rules or better predictive accuracy).
8.6 Summary

In this chapter, studies in rule extraction and refinement for the six learning examples have been presented. The empirical evaluation has been conducted to evaluate the proposed MBRL system and other existing rule extraction techniques. It has been shown that, as a post-processing tool, the market-based rule learning (MBRL) system has the capability to enhance the quality of symbolic rules on various learning problems. The MBRL system can improve the predictive accuracy of fuzzy rule sets as well as reduce the number of rules. Its capability of simplifying NeuroLinear rule sets and improving the accuracy has been demonstrated through different problem-solving examples. The learning ability of the MBRL system starting with no prior knowledge has also been evidenced by the experimental results. These have shown that the MBRL system can be used not only as an additional tool to refine fuzzy neural network or feed-forward neural network extracted rules, but also as an independent learning model to accomplish complex computational tasks and extract inference rules at the same time.

To select the “best” rule generation technique will depend on the preferences of the analyst. A balance between different rule performance criteria is needed, and always relying on any single criterion is likely to be unwise. Comparisons of experimental results have demonstrated that both the C4.5rules method and the X2R approach can be fast methods for generating good quality rules on small to medium size data sets. However for larger data sets, the NeuroLinear-generated rules can achieve good accuracy rates with far fewer rules. The use of fuzzy logic makes possible the treatment of imprecise, uncertain or incomplete information, which is very common in real classification problems. Although in some experiments, the accuracy of the ReFuNN-generated fuzzy rules was not satisfactory, their predictive accuracy was improved by the assistance of the MBRL system.

Since the goodness of a rule set sometimes is rather subjective and/or application-dependent, there may be advantages in using several or all of the available procedures in order to mine as much ‘knowledge’ as possible from the available data. The next chapter will conclude the research results and provide pointers for future research directions.
Part IV
Conclusions

Part IV contains Chapter 9, which summarizes the contributions of this thesis, limitations of the work presented, and proposed future work.
Chapter 9 Conclusions

9.1 Summary

One of the main themes in this research investigation has been the feature selection process. A feature selection approach that employs neural networks has been presented, and three associated pruning schemes that make automatic selection of the pruning threshold have been proposed. The proposed neural network techniques have been evaluated and compared with the $\chi^2$-statistic-based discretization algorithm, called Chi2, by experimenting with six practical applications. The analysis of these experimental results has shown that both the Chi2 and the neural network feature selection approaches are straightforward to apply and effective. In addition these results demonstrates that in the course of rule discovery, neural networks can serve multiple roles, not only as a connectionist learning technique but also as a useful tool to select the most relevant features.

Since the Chi2 algorithm has been shown to be a useful technique for feature selection, additional studies have been undertaken to evaluate its effectiveness in other aspects of feature selection. For example, how to apply the Chi2 algorithm to a large geographical data set in order to perform spatial data filtering has been discussed in detail. The case studies have shown that spatial data filtering can successfully reduce spatial data “vertically” (reducing the number of data items) and “horizontally” (reducing the number of features), so that neural network computation can be more efficiently performed. In addition, a novel approach of employing the Chi2 algorithm to select membership functions for fuzzy systems has been proposed and studied. In connection with the applications of fuzzy neural networks, experiments have demonstrated that an automatic selection of the number and widths of the membership functions by the Chi2-based membership function selection method can lead to an improvement of the generalization ability of FuNN fuzzy neural networks.

The development of a novel market-based rule learning (MBRL) system and the investigation of its capability of evolving and refining rules have been key elements of this research. As a classifier system-inspired model, it introduces a novel element by importing existing rule sets generated by other rule extraction techniques into the system. This basic change not only makes
the MBRL system begin with pre-established rule sets with a relatively limited complexity, rather than a random set, but also enhances the likelihood of being able to interpret the evolved rules. Moreover, the MBRL system produces various modifications in each of the layers of the structure. With the modifications introduced by the MBRL system, the problems existing in current classifier systems can be solved or lessened. In order to derive information for setting appropriate starting values for system parameters, a mathematical analysis of the MBRL system’s steady-state behaviour has also been presented.

The MBRL system has been proposed as a post-processing tool to be used with fuzzy neural networks (FuNN models) and the fuzzy neural network rule extraction technique, ReFuNN, in order to provide a general framework for fuzzy inference-based knowledge discovery. Similarly, as a post-processing tool, the MBRL system has also been proposed for use with feed-forward neural networks and the feed-forward neural network rule extraction technique, NeuroLinear, in order to improve the quality of extracted rules from feed-forward neural networks. In terms of (a) rules that have been extracted by the ReFuNN approach from fuzzy neural networks (FuNN models), and (b) rules extracted using the NeuroLinear approach from feed-forward neural networks, we have illustrated:

• how to encode and feed the input information and initialize rules into the system,
• how ‘payment’ and ‘receipt’ are distributed among the rules,
• how the GA performs, and
• how training is carried out within the system.

Based on six data sets drawn from a range of representative learning examples, the MBRL system has been experimentally evaluated using ReFuNN-generated rules and NeuroLinear-generated rules. In order to test the MBRL system’s learning abilities as an independent learning model, experiments have also been conducted by applying the MBRL system starting with no prior knowledge. For comparison and evaluation purposes, the six data sets have also been used in connection with rule extraction using existing techniques, such as C4.5rules and the rule generator X2R. The systematic analysis of experimental results has shown that:

• As a post-processing tool, the MBRL system can lead to improved rule performance.
• The MBRL system can greatly improve the predictive accuracy of fuzzy rule sets as well as reduce the number of rules.
• The MBRL system has a significant capability of simplifying NeuroLinear-generated rule sets as well as improving their predictive accuracy on various computing tasks.
• By illustrating how the MBRL system could find solutions for six learning examples from scratch, the MBRL system has been shown to have potential as an alternative generic learning technique that can be used to complement, or as an alternative to, conventional connectionist models to accomplish complex computational tasks.

In the following sections we first summarise the scientific contributions of this research. Secondly we discuss the limitations encountered in the research. The thesis ends with suggestions for possible future research.

9.2 Main Contributions
This research has investigated various techniques for discovering inference rules from data sets. Following the three steps of a knowledge extraction process, namely pre-processing (feature selection), rule discovery process, and post-processing (rule refinement), the research has addressed some important and current difficulties in these three steps and has introduced and integrated a ‘market trading’ technique with existing techniques from the field of rule discovery and refinement with respect to data mining. The main contributions of this research can be summarized as follows:

- A novel approach of employing the Chi2 algorithm to select membership functions for fuzzy systems has been proposed in Section 5.3 of Chapter 5. The Chi2-based membership function selection method produces automatically an appropriate selection of the number and widths of the membership functions. In applications of fuzzy neural networks, it has been demonstrated that this technique can improve the generalization ability of FuNN fuzzy neural networks.
- A novel market-based rule learning (MBRL) system has been proposed and developed in Chapter 6. As a classifier system-inspired model, it introduces the fundamental change of importing existing rule sets generated by other rule extraction techniques into the system and making various modifications in each of the layers of the structure. With the modifications
introduced by the MBRL system, the problems existing in current classifier systems can be solved or lessened. Its capability to evolve a set of co-adapted rules across a wide range of problem domains has been explored. It has been demonstrated that the MBRL system has the potential to be an alternative generic learning technique that can be used to complement, or serve as an alternative to, conventional connectionist models to accomplish complex computational tasks.

- A general framework for fuzzy knowledge discovery that comprises fuzzy neural networks (FuNN models), the fuzzy neural network rule extraction technique, ReFuNN, and the MBRL system, has been proposed in Section 7.2 of Chapter 7. It has been shown that as a post-processing tool, the MBRL system can improve the predictive accuracy of fuzzy rule sets as well as reduce the size of the rules.

- A general framework for neural knowledge discovery that combines the feed-forward neural networks with the feed-forward neural network rule extraction technique, NeuroLinear, and the MBRL system, has also been proposed in Section 7.3 of Chapter 7. The experimental results have demonstrated that as a post-processing tool the MBRL system can lead to improved rule performance by reducing the number of rules while increasing generalization ability.

The above proposed data mining techniques (the Chi2-based membership function selection method, the MBRL system, and the associated frameworks for knowledge discovery and refinement) have been successfully applied to practical scientific applications. Five related research papers (Purvis et al. 1999; Purvis et al. 2001; Zhou et al. 2001; Bradshaw et al. 2001; Bradshaw et al. 2002) have been published in international journals, four related research papers (Zhou et al. 1997a; Hales et al. 1998a; Zhou & Purvis 1999; Drumm et al. 1999) have been presented at international conferences, and two technical reports (Zhou et al., 1997b; Hales et al. 1998b) have been produced.

Following the three steps of the knowledge discovery process, a detailed summary of the research contributions is as follows:
Pre-processing (feature selection)

- The $\chi^2$ statistic-based feature selection algorithm, Chi2, has been applied for application to large spatial data sets by performing spatial data filtering in order to reduce the size of the data space without sacrificing the discriminating power of the original data.
- A Chi2-based membership function selection method has been proposed to make automatic selection of membership functions for fuzzy systems.
- Three different pruning algorithms to make automatic selection of the pruning threshold for the efficient implementation of feature selection via neural networks have been developed, and the effectiveness of the proposed pruning algorithms has been demonstrated by the experimental results on six learning examples.
- The neural network feature selection approach has been compared empirically with the $\chi^2$ statistic feature selection approach. The analysis of experimental results has shown that each of the methods has its own merits and potential applications.

Discovery Process

- Several different rule extraction approaches including (a) a method based on C4.5 decision trees, (b) a rule generator X2R, (c) a fuzzy rule extraction method ReFuNN, and (d) a neural network rule extraction method, NeuroLinear, have been applied to extract inference rules from application data sets in six problem domains. The experimental results have been compared and evaluated. The results have shown that for simple tasks both the C4.5rules method and the X2R approach are rapid methods for generating high quality rules, whilst for more complex data sets, variants based on neural networks are likely to be the preferable choices.
- A market-based rule learning (MBRL) system has been proposed and developed. Its structure, properties, and assumptions have been presented. The differences between the proposed MBRL system and Holland’s classifier system have been discussed in detail. With the changes introduced by the MBRL system, problems existing in current classifier systems, in particular, interpretation difficulty, can be solved or reduced.
- Based on the six data sets drawn from a range of representative learning examples, the proposed system’s independent learning capability has been investigated by starting the system with no prior (seeded) knowledge. The investigation has shown that the proposed MBRL system is not only capable of performing learning tasks without prior knowledge, but can also induce inference rule sets.
Post-processing

- As a post-processing tool, the MBRL system has been proposed in combination with the fuzzy neural network rule extraction algorithm, ReFuNN, in order to provide a general framework for fuzzy knowledge discovery. An approach for representing fuzzy rules and feeding fuzzified input information to the MBRL system has been presented.
- As a post-processing tool, the MBRL system has also been proposed for refining the two levels of rules extracted by the NeuroLinear algorithm from feed-forward neural networks. Neurolinar rule representations and input information encodings have been described. The rule matching procedures and rule strength distributions have been studied, and the GA operating process based on two-level rule bases has been presented.
- Six learning problem domains have been used for conducting experiments to evaluate the effectiveness of the MBRL system as a post-processing tool. An analysis of the experimental results has shown that the MBRL system is a potentially useful additional tool that can be used to refine (fuzzy) neural network extracted rules and possibly discover and add some new, better performing rules. As a result, it can lead to improvements by increasing the accuracy of the rule inference and/or improving the comprehensibility of the rules.

9.3 Limitations

In this research, the following limitations have been observed in connection with the MBRL system:

- The MBRL system, when applied alone to large data sets is not generally scalable. However, as a post-processing tool that refines existing rule sets derived from other techniques, the scalability of MBRL system does not raise a particular concerns for data mining applications.
- It has been observed that the learning behaviour of the MBRL system can sometimes be highly dependent on the starting parameters of the system. Although the mathematical analysis of the system’s steady-state behaviour has provided information concerning the setting of appropriate values, their optimal values and how these parameters influence the behaviour of the system remain undetermined.
- The MBRL system, as it is currently implemented, only applies to single-level or two-level rule bases. Multiple-level rule bases (greater than two levels) and more complex learning applications have not been implemented. Such multiple-level rule bases have, however, rarely
been discussed in the literature.

9.4 Suggestions for Future Work
Knowledge discovery and data mining is an active subject of research (Mitchell 1999). There are many interesting challenges and opportunities that lie ahead. In the three steps of the knowledge discovery process studied in this thesis there are a number of possible improvements that could be made. These and other related recommended future work are discussed in the following:

Pre-processing (feature selection)
• As a foundation of neural network feature selection and rule extraction, more investigation of pruning algorithms is worthwhile. With respect to the selection of the optimal pruning-threshold \( \eta^2 \) in the PBS pruning scheme (see Section 4.2.2 of Chapter 4) in order to avoid over- or under-removal of the weights of the network, instead of a linear function, it may be worthwhile to use more advanced mathematical functions that change their values during the neural network training process, thereby removing larger numbers of connection weights during the early training stages and eliminating fewer connection weights when the local minimum in the error surface or a predefined accuracy is approached. The combination of the PBS and VBS pruning schemes during the neural network training is also worth investigating. At the beginning of the training process, the PBS scheme can be employed to ensure a certain proportion of total weights to be removed. When the total number of existing weights is sufficiently small or the values of these weights are close to each other, the VBS pruning scheme can be adopted to eliminate the connection weights effectively.

• In this research, we have studied two effective feature selection approaches: the Chi2 algorithm and the neural network feature selection approach. Because of the increasing demand for feature selection, the evaluation of the vast range of different feature selection methods may prove to be useful (Liu & Motoda 1998).

• In recent years, feature transformation in data pre-processing has attracted a broad interest (Liu & Motoda 1998). Feature construction and feature extraction are two variants of feature transformation. Feature construction is a process that discovers missing information about the relationships between features and augments the space of features by inferring or creating additional features (Matheus 1991; Wnek & Michalski 1994). Feature extraction is a process that extracts a set of new features from the original features through some functional mapping
After feature extraction, we have \( B_1, B_2, \ldots, B_m (m < n) \), \( B_i = F_i(A_1, A_2, \ldots, A_n) \), where \( F_i \) is a mapping function. Feature transformation and feature selection are not totally independent issues. They can be viewed as two sides of the representation problem. We can consider a set of features as a representation language. In some cases where this language contains more features than necessary, feature selection helps simplify the language; in other cases where this language is not sufficient to describe the problem, feature transformation helps enrich the language by constructing compound features. The use of feature transformation and feature selection depends on the goals of the analyst. The former aims to preserve the topological structure of the data, whereas the latter aims to enhance the predictive power. Further investigation in this area would be of value.

As indicated earlier, scalability is a challenge for pre-processing and discovery techniques, in particular for neural network rule extraction techniques when domain classification tasks contain millions of cases or features. When facing a huge input data set, a typical way to proceed with the data mining process is to reduce the size of the data in such a way that the reduction does not change the result drastically. Data sampling is one such data reduction method. As argued by several researchers (Wang et al. 1998; Liu & Motoda 2002), random sampling is difficult to use due to the difficulty of determining an appropriate sample size. The adaptive sampling techniques proposed by Domingo et al. (2002), which obtains samples sequentially in an on-line fashion and determines from the obtained examples whether it has already seen a large enough number of examples, may be worth further investigation.

**Discovery Process**

As a new learning model and knowledge discovery tool, the MBRL system is still in its early stages of development and exploration. Compared with the vast number of theoretical and experimental studies on other learning models, such as decision trees and feed-forward neural networks, it is unlikely that this initial research on the MBRL system has explored the full extent of its potential. Further experiments with different learning problems are worth pursuing in order to gain a better understanding of the MBRL system behaviour in different circumstances.

The MBRL system requires a large number of system starting parameters to be specified. These starting parameters can greatly influence the learning capabilities of the system. More extensive study would help to investigate the optimal set of system parameter values and how they...
influence the systems’ behaviour. One possibility that might be worth investigating is to make the value of parameters changeable during the learning process. The parameters that determine the behaviour of the genetic algorithm are good candidates for such modification. For example, the mutation probability can be set to a large value at the beginning when there is no problem-specific knowledge embodied in the rule bases. Later, it may be desirable to automatically reduce its value during the learning process as more and more knowledge is incorporated in the rule bases. Several approaches to self-adaptation of GA parameters (Booker 1987; Davis 1989; Fogarty 1989; Davis 1991) have been associated with efforts to optimize parameters during the evolution process. Davis’s (1991) approach, which assigns to each operator a “fitness” based directly on the degree to which an operator is successful at producing good offspring, is worth exploring in connection with the application of the MBRL system.

• In the MBRL system, a single rule is taken as an individual of genetic manipulation, which follows Holland’s classic classifier system - the Michigan Approach (Holland & Reitman 1978). However, the Pitt Approach (De Jong 1988), which takes the entire rule set as an individual for genetic evaluation is worth investigating. Roubos and Setnes (2002) has shown that the Pitt Approach can sometimes evolve sets of inference rules with improved accuracy and less complexity.

• As an independent learning model, the MBRL system has only been tested on the basis of two-level NeuroLinear-type rules. Its ability to learn solutions based on other rule formats or representations is worth further exploration. It is possible that the MBRL system can lead to better learning solutions if it is based on multiple-level rule bases. This is because the multiple-level rule bases may provide a wider space for the MBRL system to search chains of well performing rules in order to achieve in-depth learning. This is an area that may offer promise for additional experimental investigation.

• The GA adopted in this research uses floating point and integer formats for genome representations of potential solutions, however other types of representation, such as the binary representation (Goldberg 1989a), trees or graphs (Forrest 1990), may prove to be more effective for some specific types of problem solving, and should also be investigated.

Post-processing
• As a post-processing tool, the MBRL system has been applied to refine ReFuNN-generated rules and NeuroLinear-generated rules in this research. Many other rule extraction techniques in conjunction with the MBRL system have yet to be investigated. For example, the rule extraction technique X2R was found to have poor predictive accuracy and to generate a large number of rules when it is applied on some large data sets. It would be interesting to incorporate X2R-generated rules into the MBRL system, with the goal of reducing the large size of the rule base as well as improving the predictive accuracy. Norman Packard’s approach (Packard 1990), which used a GA to search the space of sets of conditions of the rules, is worth exploring in this connection of using the MBRL system to refine X2R-generated rules.

In conclusion, data mining is a fruitful area of research with an abundance of practical applications (Liu & Motoda 1998). The combination of evolutionary, competition-based mechanisms with existing connectionist and inference-based tools holds considerable promise for the future.
Bibliography


http://www.netsci.org/Science/Compchem/feature02.html


