C 417

MONASH UNIVERSITY
THESIS ACCEPTED IN SATISFACTION OF THE
REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
ON..... 14 September 2004.....
.....[redacted].....
Sec. Research Graduate School Committee

A Hybrid Model for Intelligent Decision Support: Combining Data Mining And Artificial Neural Networks

Sérgio I. Viademonte da Rosa

(B.CompSci, MsC.InfoSys)

This thesis is submitted in fulfilment of the requirement for the degree of

Doctor of Philosophy (Computing)

School of Information Management & Systems

Faculty of Information Technology

Monash University

Australia

June 2004

Errata

p 22 section (1.1) para 1, line 4: "a research" for "research"

p 23 section (1.2) para 3, line 1: "decision support systems" for "intelligent decisions support systems"

p 23 section (1.2) para 4, line 2: "systems have been" for "systems has been"

p 25 para 1, line 5: "DSS" for "IDSS"

p 31, line 1: "of such conceptual" for "of such a conceptual"

p 37, line 2: "literatures" for "literature"

p 38 para 1, last sentence: "system" for "systems"

p 60, line 3: "inputs" for "input"

p 70 para 2, line 2: "WAtewater" for "WAstewater"

p 75 section (2.6) para 2, line 13: "Kolonder" for "Kolodner"

p 78 section (3.1) para 1, line 3: "Frawley" for "Frawley et al"

p 81 para 1, line 3: "pre-processing these data until to effectively mine the data" for "pre-processing these data, to effectively mining the data"

p 115, line 4: "Kolonder" for "Kolodner"

p 124 para 2, line 3: "the rule, e.g." for "the rule, i.e."

p 124 para 2, line 5: "input nodes, e.g." for "input nodes, i.e."

p 148 para 3, line 10: "In such IDS" for "In such an IDS"

p 174 para 1, line 2: "bring few problems" for "bring a few problems"

p 197, line 15: "amount of rules" for "number of rules"

p 215 section (6.3.3) para 1, line 1: "categories, e.g. hypotheses" for "categories, i.e. hypotheses"

p 232 para 3, line 2: "specially" for "especially"

p 232 para 7, line 2: "sets" for "set"

p 310 para 2, line 2: "Fethcer" for "Fetcher"

p 293, line 1: "Kolonder" for "Kolodner"

Addendum

p 23 line 23: Add the reference: (Turban, Aronson and Liang, 2005).

p 24 line 11: Add at the end of para 1 the reference: (Turban, Aronson and Liang, 2005).

p 55 line 17: Add at the end of para 1 the reference: (Turban, Aronson and Liang, 2005).

p 212: Add at the end of line 24:

Appendix D, page 312, discusses the weighting values assigned to β using the domain Model1-60 as example.

p 302: Add after the reference (Turban and Aronson, 1998):

Turban, E., Aronson, J. and Liang, T. (2005). *Decision Support Systems and Intelligence Systems.* 7th Ed., New Jersey: Pearson Prentice-Hall.

I dedicate this thesis to:

my mother Conceicao Viademonte,

for her unreserved support on all the roads I have travelled on through my life, and for the ones still to come. She remained supportive even when some of those roads seemed a little unreasonable, such as a Brazilian guy crossing the world to pursue a PhD in Australia!

my sister Claudia V. Jung,

for her support, friendship, confidence and example in so many situations, and also for looking after our mother so well. I am very fortunate to have such a sister!

my niece Catarina V. Jung,

who joined our family in an early morning of July 2002, bringing a whole new light and happiness into our lives. She has certainly given me a different perspective on life. Suddenly, those guys and girls playing with their kids in Melbourne gardens during lazy sunny weekends became something different to my eyes! Thanks Cat!

The photos of Cat that have been periodically sent to me have been a great source of motivation and encouragement. I hope the work I have done in this thesis will somehow help her in life. This will be a good enough reason to have written a PhD thesis.

my father Irio J. da Rosa,

for his example of dedication and commitment, since my early days in the Brazilian country town of Mauricio Cardoso.

Abstract

The aim of this research is to devise a framework for intelligent decision support system (IDSS) that uses domain knowledge from organizational databases and applies this knowledge in problem solving. This research draws from the concepts of computational intelligence, knowledge discovery in databases and decision support.

This research is concerned with investigating how data mining (DM) and artificial neural networks (NN) can cooperate in order to minimize problems related to knowledge acquisition, and implement reasoning and learning in decision support systems.

As a result of this investigation a new decision support framework is proposed combining an association rule generator algorithm for data mining with an artificial neural network based system in a hybrid architecture, called the DM-NN model.

Data mining is applied to induce expert domain knowledge from organizational databases, minimizing the problem of knowledge acquisition. The discovered association rules are stored in rule-based knowledge base. A neural network based system is introduced to provide intelligent capabilities; in particular, learning and problem solving, taking advantage of neural networks capability of generalization, handling large combinations of data, and coping with noise data.

To validate and assess the performance of the DM-NN model it has been implemented in the context of aviation weather forecasting, identifying severe and rare weather phenomena at airport terminals, particularly fog phenomena. Severe and rare weather events are intrinsically problematic to predict because weather forecasters normally do not have extensive experience in forecasting such events; as a result false alarms or incorrect forecasts are more likely to occur. Furthermore, severe weather events are hazardous events that can potentially cause serious damage and unsafe conditions. These make forecasting of severe weather events a suitable area for intelligent decision support.

As part of this research specific studies on data sampling, data preprocessing for data mining and rule filtering thresholds were conducted and assessed through the implementation of the DM-NN model in aviation weather forecasting.

A quantitative approach was used to assess the DM-NN model performance, where the holdout method was employed using a database provided by the Australian Bureau of Meteorology (BOM), with 49901 weather observations from July 1970 until June 2000, taken at Tullamarine Airport.

The results achieved in this research demonstrated that the DM-NN model constitutes suitable technology to implement the IDSS framework. The performance of the DM-NN model in identifying fog phenomena demonstrated its potential applicability as a decision support framework.

Declaration

I certify that this thesis does not incorporate, without acknowledgement, any material previously submitted for a degree or diploma in any university, and that to the best of my knowledge and belief; it does not contain any material previously published or written by another person where due reference is not made in the text.

Signatu[redacted] Date 10/09/2004

Acknowledgments

I would like to thank,

Prof. Frada Burstein, my principal supervisor, for her confidence and invaluable support in so many situations. Her optimism and ability to believe in solutions (even when solutions seem very unlikely) have been essential throughout the years I spent doing my research at Monash University. I also thank her help in getting financial support in the times it was most needed. It has been a great experience to work with her.

The staff from the School of Information Management and Systems for their outstanding friendship and support. Luckily, they are too many to mention here, but I would like to name Vicki Raicevic who has reminded me about the various university forms to fill, submission deadlines and helped me to deal with traffic offences in Melbourne (much appreciated!). Thanks to Judith Bruce for being so patient and supportive, and to Trudi Robinson for being so helpful and kind.

The staff from the Central Operations Systems Branch of the Australian Bureau of Meteorology (Melbourne) for their cooperation with my research, providing access to data and expertise in the application area. I would like to thank Dr. Robert Dany for his valuable explanations and discussions about meteorology.

Fabio Beckenkamp, for his friendship (and there were times when friends were difficult to recognise), and for his support and discussions about the CANN project.

Patrick Cao, for his support, friendship and also for his useful comments and discussions (some times overexcited) about my research. His amazing knowledge about Chinese history had led to very interesting conversations, apart of PhD issues.

Ilona Jagielska for her helpful comments and advises.

Henry Linger, for his support in my research at Monash University.

Claudia Bitencourt, for being a good and supportive friend. I am very glad for her support, mainly during the times I have spent in Brazil.

Paulo Jung, his support in Brazil concerning my family has been essential during the all time I have been living overseas.

The Australian Commonwealth Department of Education, Training and Youth Affairs and Monash University for providing the financial support for the development of my PhD project.

Thankfully, life goes beyond the university campus. My life in Australia would be less enjoyable without the support of many people I have met and who became my friends. I would like to thank

Dorothy Andrew (Mrs. Dorothy) for her hospitality and for hosting me since I first came to Melbourne. Her support in so many situations has been invaluable.

All people from SIMS postgraduate research centre, they are really great people to work with and have been amazingly supportive.

Nick (Nicholas Jackson) for being such a good and reliable friend. I am glad for the many enjoyable times (and others not that enjoyable) we spent sailing at Port Phillip Bay, skiing in the Victoria highlands and trekking somewhere in country. Nick's energy and motivation have been really encouraging.

Mark, Carolyn and Janet for their always available support and friendship. Special thanks to Mark regarding his skills in the kitchen. This has been specially appreciated during the last few Christmas lunches.

Phil Braziones for his friendship and invaluable support since I came to Australia. He is certainly an interesting character.

Jocelyn San Pedro for being a helpful and good friend.

My peers from Keith Geyer School of Karate (Melbourne dojo) for their friendship and enormous support; I will always be in debt with them! I would like to thanks Lucas, David, Pam, Luke, Cindy, Dean, Jono, ... fortunately, they are too many to name all here!

A friend who gave me special support and demonstrated a great character and felling of reliability, since I first joined the JKA Malvern dojo. Thanks to Robert Sachs.

My thesis is also the result of the work I have done before I came to Australia. I would like to thank my friends from Germany for their support, friendship and for being always so reliable. Thanks to Klaus Haegele, Ines Hensler, Greg, Annika, Edmund and Nicole (and there are many others).

Everyone who kept asking me when I was going to submit my thesis!

During the time of my candidature I have met people that influenced my work in different ways for different reasons. Some of them brought remarkable effects into my live by their outstanding personality and character. In this regard I would like to thank

S. Keith Geyer, who, among many things, tough me that achievements result from determination, discipline and, essentially, desire. I thank him for his friendship and the privilege of been training under his guidance.

S. Stan Schmidt, for his example as human being. This will remain with me.

A PhD. project usually requires an extensive and intensive work. The support from family, supervisors, friends and colleagues are essential to carry out such a project. However, at the end of day, a PhD project is a work that is developed entirely by the student, under his/her own responsibility.

As such,

I thank God for provided me enough self confidence, self discipline and motivation in carrying on this work, as most of the time there were just us along the pathway.

and here we go!

Publications

The following papers address different stages of this research and were published during the period of candidature:

- (Viademonte et al., 2001a), discusses the stages of performing knowledge discovery in the aviation weather forecasting domain.
- (Viademonte and Burstein, 2001b), introduces the integration of artificial neural networks components for the KDD purposes in decision support context.
- (Viademonte and Burstein, 2001a), describes a framework for Knowledge Management (KM) based on the context of Knowledge Discovery in Databases (KDD).
- (Viademonte et al., 2001b), describes the integration of an Intelligent Decision Support System (IDSS) within the KDD process. The paper discusses the basic components of the hybrid computational architecture proposed for such an integration.

This research project was grounded by previous research projects I have participated in, which resulted in several publications:

- (Viademonte, 1994), describes the development and implementation of a hybrid model for decision support system that employs two knowledge representation formalisms, neural network and relational data structures.
- (Viademonte, Hoppen and Beckenkamp, 1997), describes the use of fuzzy logic to model semantic variables in hybrid expert systems.
- (Viademonte, Burstein and Beckenkamp, 2000), empirically verifies the suitability of mobile and distributed computing in decision support systems, through the implementation of an NN model using Voyager ORB.

Acronyms and Abbreviations

ADAM	Australian Data Archive For Meteorology
AI	Artificial Intelligence
ART	Adaptive Resonance Theory
BOM	Australian Bureau of Meteorology
BPN	Backpropagation Neural Model
CANN	Components for Artificial Neural Networks
CART	Classification And Regression Trees
CAV	Civil Aviation Authority
CBA	Classification Based Automated
CBR	Case Base Reasoning
CNM	Combinatorial Neural Model
DAO	Data Access Object
DEC	Digital Equipment Corporation
DM	Data Mining
DSS	Decision Support System
EPA	Victorian Environment Protection Authority
EPAM	Elementary Perceiver and Memorizer
ES	Expert Systems
ETL	Extract, Transform and Load
FAR	False Alarm Ratio
FL	Fuzzy Logic
FSEP	Forecast Streamling and Enhancement Project
GA	Genetic Algorithms
GDR	Generalized Delta Rule
GPS	General Problem Solver
HS	Hybrid Symbolic-Connectionist Systems
IAS	Intelligent Advisory System
IDS	Intelligent Decision Support
IDSS	Intelligent Decision Support System
IRP	Incremental Reward and Punishment
IS	Intelligent Systems

KDD	Knowledge Discovery in Databases
KG	Knowledge Graphs
KIPSE	Knowledge-based Interactive Problem-solving System
KM	Knowledge Management
MLP	Multilayered Perceptron
NN	Artificial Neural Network
ORB	Object Request Broker
PC	Probabilistic Computing
POD	Probability of Detection
RI	Rule Induction
SOM	Self-Organizing Feature Maps
SRP	Starter Reward and Punishment
TAF	Terminal Aerodrome Forecast
TASA	Telecommunication Alarm Sequence Analyser
TTF	Trend Type Forecast
VRFC	Victorian Regional Forecasting Centre
WATTS	Wastewater Treatment System

Table of Contents

Abstract **3**
Declaration **5**
Acknowledgments **6**
Publications **9**
Acronyms and Abbreviations **10**
Table of Contents **12**
List of Figures **17**
List of Tables **20**
1 Introduction **22**
1.1 Research Overview 22
1.2 Theoretical Background 23
1.3 Research Question and Objectives 26
1.4 Contributions 28
1.5 Research Design 29
1.5.1 Research Method 29
1.5.2 Outline of the Research Design 30
1.5.3 Performance Assessment Method 32
1.6 Structure of the Thesis 34
1.7 Chapter Summary 35
2 Intelligent Decision Support Systems **36**
2.1 Introduction 36
2.2 Intelligent Systems 36
2.2.1 Motivations of Intelligent Systems Development 38
2.2.2 Intelligent System Capabilities 40
2.2.3 The Role of Intelligent Computing Technologies in Building Intelligent Systems 43
2.2.4 Intelligent System Classifications 48
2.3 Hybrid Symbolic-Connectionist Systems 51
2.4 Artificial Neural Networks 54
2.4.1 Biological Conceptualisation of Neural Networks 55

2.4.2 Neural Networks Conceptualisation 57
2.4.3 Neural Network Structures 59
2.4.4 Neural Network Learning Approaches 61
2.4.5 Neural Network Models 65
2.4.5.1 The Backpropagation Neural Network Model 66
2.4.5.2 The Combinatorial Neural Model 67
2.5 Applying Intelligent Systems 69
2.6 Difficulties in Developing Intelligent Systems 75
2.7 Chapter Summary 77
3 Knowledge Discovery in Databases 78
3.1 Introduction 78
3.2 The Process of Knowledge Discovery in Databases 79
3.2.1 Classification of KDD Problems 81
3.3 Data Mining 82
3.3.1 Data Mining Tasks 82
3.3.1.1 Classification Tasks 84
3.3.2 Algorithms for Data Mining 85
3.3.2.1 The Apriori Algorithm for Association Rules Learning 89
3.4 Data Preparation 91
3.5 Dimensionality Reduction 95
3.6 Data Sampling 96
3.6.1 Sampling Methods 97
3.6.2 Designing the Training Dataset 102
3.7 Difficulties in Knowledge Discovery Applications 104
3.8 Chapter Summary 106
4 The Hybrid DM-NN Model for IDSS 107
4.1 Introduction 107
4.2 Overview of the Hybrid DM-NN Model 108
4.3 The DM-NN Architecture 110
4.3.1 Decision-Oriented Data Repository 112
4.3.2 Case Base 113
4.3.3 Data Mining Component 114

4.3.4 Knowledge Base 115
4.3.5 Intelligent Advisory System - IAS 116
4.4 The Knowledge Representation Schema 118
4.4.1 Knowledge Graphs 118
4.4.2 Representing Domain Knowledge 121
4.4.2.1 Representing Knowledge Through Association Rules 121
4.4.2.2 Representing Knowledge Through Neural Networks 123
4.4.2.3 Representing Knowledge Through a Hierarchy of Classes and Objects 125
4.5 Components for Artificial Neural Network – CANN 127
4.5.1 The CANN Architecture 128
4.5.1.1 Object-Oriented Modelling of Neural Networks 129
4.5.1.2 Domain Representation 132
4.5.1.3 Data Conversion 133
4.6 The Combinatorial Neural Model - CNM 134
4.7 The Functionality of the DM-NN Model 138
4.7.1 Applying the DM-NN Model 140
4.8 Chapter Summary 141
5 Applying the DM-NN Model in Aviation Weather Forecasting: The Knowledge Discovery Stage 142
5.1 Introduction 142
5.2 Issues in Aviation Weather Forecasting 143
5.2.1 Weather Forecast at Tullamarine Airport 145
5.2.2 Fog Formation and Definitions 147
5.2.3 The Intelligent Decision Support for Aviation Weather Forecasting 148
5.3 Discovering Knowledge from Meteorological Databases 149
5.3.1 The Domain of Aviation Weather Forecasting 152
5.3.2 Understanding Meteorological Data 153
5.3.3 Data Preparation 155
5.3.3.1 Feature Selection and Construction 155
5.3.3.2 Analysis of Null Values 157
5.3.3.3 Analysis of Variability 160
5.3.3.4 Data Transformation 167

5.3.4 Data Modelling for Data Mining 173
5.3.4.1 Sampling Data 175
5.3.4.2 Generating Data Mining Models 180
5.4 Building Knowledge Bases: Mining Data 181
5.4.1 Applying Data Mining 182
5.4.1.1 Selecting the Target Attribute 182
5.4.1.2 Features Selection 183
5.4.1.3 Selection of Attribute Values 183
5.4.1.4 Discretization of Numerical Attributes 185
5.4.1.5 Selecting Mining Parameters 187
5.4.2 Generating Knowledge Models 190
5.4.2.1 Knowledge Models V2 191
5.4.2.2 Knowledge Models V3 194
5.4.2.3 The Knowledge Bases 196
5.5 Chapter Summary 200
6 Applying the DM-NN Model in Aviation Weather Forecasting: The Intelligent Advisory System Stage 202
6.1 Introduction 202
6.2 Modelling the Domain of Aviation Weather Forecasting 205
6.3 Integrating the Domain Model of Aviation Weather Forecasting into the CANN Simulator 208
6.3.1 Domain Elements 209
6.3.2 Modelling Evidences 210
6.3.3 Modelling Hypotheses 215
6.4 The Learning Process 216
6.4.1 The CNM Learning Mechanism 216
6.4.2 Training CNM with Association Rules 218
6.5 Consulting and Testing Cases 222
6.6 Comments 227
6.7 Chapter Summary 228
7 Analysis of Performance 230
7.1 Introduction 230

7.2 The Assessment Method 232
7.3 Performance on Data Models 233
7.3.1 Performance According to Sampling Proportions 235
7.3.2 Performance on Training Sets 237
7.3.2.1 Performance of Data Mining Model1-60 237
7.3.2.2 Performance of Data Mining Model1-80 238
7.3.2.3 Performance of Data Mining Model2-60 239
7.3.2.4 Performance of Data Mining Model10-60 240
7.4 Performance on Data Mining Parameters 241
7.4.1 Performance According to Rule Confidence Degrees 242
7.4.2 Performance According to Rule Support and Rule Order 244
7.5 Some Specific Analysis with Weather Observations 248
7.5.1 Assessing the Visibility Observation 248
7.6 Discussions 252
7.7 Chapter Summary 255
8 Discussions About the DM-NN Approach 256
8.1 Concepts Related to the DM-NN Architecture 256
8.2 Knowledge Discovery Related Work 258
8.3 Improvements and Limitations 266
8.4 Chapter Summary 270
9 Conclusions 271
9.1 Introduction 271
9.2 Results 271
9.3 Summary of Contributions 277
9.4 Future Work 278
Bibliography 283
Appendices 306

List of Figures

Figure 1.1: Research design 31

Figure 2.1: Intelligent systems components as in (Medsker, 1995) 38

Figure 2.2: Fuzzy modelling for flow rate variation variable as in (Viademonte, Hoppen and Beckenkamp, 1997) 45

Figure 2.3: Structure of a typical neuron as in (Freeman and Skapura, 1991) 56

Figure 2.4: Processing of a generic artificial neuron 58

Figure 2.5: Multi-layer feedforward partially connected topology 60

Figure 3.1: KDD stage from database to data warehousing 80

Figure 3.2: KDD stage from data warehouse to data mining, as in (Han, 1998) 81

Figure 3.3: Relation between learning algorithms and type of problems, as in (Meneses and Grinstein, 1998) 88

Figure 3.4: The Apriori algorithm for association rules (Agrawal et al., 1996) 90

Figure 3.5: Part of categorization tree of sampling methods proposed in Gu (Gu, Hu and Liu, 2001) 98

Figure 3.6: Learning curve representing the convergence condition, as in (Provost, Jensen and Oates, 2001) 101

Figure 4.1: Building descriptive models 109

Figure 4.2: Building predictive models 110

Figure 4.3: The components of the hybrid DM-NN architecture for IDSS 112

Figure 4.4: Knowledge acquisition through data mining 115

Figure 4.5: Internal architecture of the IAS component 117

Figure 4.6: The basic structure of a knowledge graph 119

Figure 4.7: Knowledge graph for the operation state of emergency as in (Viademonte, 1994) 120

Figure 4.8: Mapping rules into the CNM topology 124

Figure 4.9: Object-Oriented representation of the aviation weather forecasting domain 126

Figure 4.10: The relationship between Neuron and Synapses objects, as in (Beckenkamp, 2002) 130

Figure 4.11: Part of the Neuron class hierarchy implemented in CANN, as in (Beckenkamp, 2002) 131

Figure 4.12: Class hierarchy for domain representations, as described in (Beckenkamp, 2002) 132

Figure 4.13: The decision support cycle of the DM-NN model 139

Figure 4.14: The decision support cycle in the context of aviation weather forecasting 141
Figure 5.1: The knowledge discovery stage 143
Figure 5.2: Schema for generating case bases 150
Figure 5.3: Schema for generating knowledge bases 151
Figure 5.4: Algorithm to calculate the previous afternoon dew point 157
Figure 5.5: Frequency distribution for dry-bulb attribute 162
Figure 5.6: Frequency distribution for previous dewpoint attribute 162
Figure 5.7: Frequency distribution for dewpoint attribute 162
Figure 5.8: Frequency distribution for total cloud attribute 163
Figure 5.9: Frequency distribution for total low cloud attribute 163
Figure 5.10: Frequency distribution for sea level pressure attribute 163
Figure 5.11: Frequency distribution for rainfall attribute 164
Figure 5.12: Frequency distribution for visibility attribute 164
Figure 5.13: Frequency distribution for wind direction attribute 164
Figure 5.14: Frequency distribution for wind speed attribute 165
Figure 5.15: Frequency distribution for Rainfall_Range attribute in the whole population 170
Figure 5.16: Three hourly meteorological observation bulletin for Melbourne 171
Figure 5.17: Distribution of classes in the population 173
Figure 5.18: Stratifying the weather observations database 176
Figure 5.19: Generating data mining and testing sets from not fog stratum 178
Figure 5.20: Frequency distribution for the Hour attribute 184
Figure 5.21: Frequency distribution for Month attribute 185
Figure 5.22: Discretizing continuous values of wind speed 186
Figure 5.23: Rules distribution for Fog and Not Fog Class, in experiment V2 192
Figure 5.24: Association rules from Model1-60, experiment V2 194
Figure 5.25: Rules distribution in Fog Class, in experiments V2 and V3 196
Figure 5.26: Association rules in Train_Model1-60V270 199
Figure 6.1: The intelligent advisory system stage 204
Figure 6.2: The aviation weather forecasting domain model 206
Figure 6.3: Creating a project for domain Model1-60 in CANN 209
Figure 6.4: The functionalities of Domain GUI 210
Figure 6.5: Evidence list in the domain of aviation weather forecasting 211
Figure 6.6: Wind Speed evidence and its attributes 211
Figure 6.7: Assigning morbidity to Wind Speed Light 213

Figure 6.8: Fetching the Wind Speed evidence 214
Figure 6.9: Modelling Fog hypothesis 215
Figure 6.10: The initial non operational CNM neural network 220
Figure 6.11: Training the CNM to learn an association rule 220
Figure 6.12: The trained CNM neural network 221
Figure 6.13: CANN Learning GUI 222
Figure 6.14: A fog case consult in CANN 224
Figure 6.15: A case consult removing the wind speed light evidence 224
Figure 6.16: CNM evaluating a not fog case 225
Figure 6.17: CNM evaluating a case base in aviation weather forecasting 226
Figure 7.1: Error rates by data model 236
Figure 7.2: Error rates on testing data by levels of confidence degrees 243
Figure 7.3: Trend of error rates on testing data by levels of rule support and rule order 245
Figure 7.4: Error rates by levels of rule support and rule order 247
Figure 7.5: Average measures on testing data by Visibility on Model2-60 250
Figure 7.6: Pathways with higher confidence degrees leading to the fog class 251
Figure 7.7: Pathways with higher confidence degrees leading to the not fog class 251
Figure 7.8: Contrasting the POD with classificatory rates 254

List of Tables

Table 2.1: Contrasting symbolic and connectionist features 53
Table 2.2: Summary of some hybrid intelligent systems 74
Table 3.1: Relation between KDD problems and data mining tasks 84
Table 3.2: Experiments on building training datasets 104
Table 4.1: Illustrating a case: a reported occurrence of fog phenomena 114
Table 5.1: Weather observations 153
Table 5.2: Analyses about null values 158
Table 5.3: Analysis of sparse attributes in fog class 159
Table 5.4: Analysis of sparse attributes in not fog class 160
Table 5.5: Fog class statistics 161
Table 5.6: Categorical codes of rainfall attribute 168
Table 5.7: Frequency distribution of Rainfall_Range attribute 169
Table 5.8: Compass points classification 172
Table 5.9: Fog class datasets 176
Table 5.10: Sample models from not fog stratum 177
Table 5.11: Data models from not fog stratum 179
Table 5.12: Data mining models 180
Table 5.13: Discretization of numerical attributes for Model1-60 187
Table 5.14: Mining parameters and thresholds 190
Table 5.15: Rule sets by data mining models in experiment V2 191
Table 5.16: Rule sets by data mining models in experiment V3 195
Table 5.17: Knowledge bases from Model1-60 198
Table 5.18: Knowledge bases from Model1-80 198
Table 5.19: Knowledge bases from Model2-60 198
Table 5.20: Knowledge bases from Model10-60 198
Table 6.1: Wind Speed modelling for domain Model1-60 214
Table 6.2: Association rules from Model1-60 218
Table 7.1: Testing sets 233
Table 7.2: Data mining models performance 236
Table 7.3: Performance of data mining model Model1-60 238

Table 7.4: Performance of data mining model Model1-80 238
Table 7.5: Performance of data mining model Model2-60 239
Table 7.6: Performance of data mining model Model2-60 on fog class 240
Table 7.7: Performance of data mining model Mode10-60 240
Table 7.8: Rule confidence degree on testing data 242
Table 7.9: Performance of Model2-60 including Visibility 249

Chapter 1

1 Introduction

This chapter introduces the research developed in this thesis, and gives an overview of its theoretical background, objectives, contributions and motivations.

1.1 Research Overview

The focus of this research is to devise a framework for decision support that uses organizational databases as a source of information, that is capable of automatically building domain knowledge from such databases and applying that knowledge in problem solving. It is a research that draws from the concepts of computational intelligence, knowledge discovery and decision support.

This thesis introduces a new approach for intelligent decision support systems (IDSS), through the combination of data mining (DM) with artificial neural networks (NN) in a hybrid architecture, called the DM-NN model.

This thesis presents the proposed IDSS model, its components and its development in aviation weather forecasting.

In the context of this thesis intelligent decision support systems (IDSS) are computer-based decision support systems capable of incorporating specific domain knowledge and performing some type of intelligent behaviour, such as learning and reasoning about the knowledge they possess in order to support decision making.

To implement these intelligent capabilities intelligent computing technologies are employed. Intelligent computing technologies comprise diverse disciplines, such as artificial neural networks (NN), rule induction (RI), fuzzy logic (FL), genetic algorithms (GA), case

base reasoning (CBR), and expert systems (ES) used in building systems that exhibit a type of intelligent behaviour such as being able to learn and reason (Zurada, 1992; Zadeh, 2000; Medsker, 1995; Sun, 1995b).

1.2 Theoretical Background

Decisions are made through analyses of explicit domain knowledge, such as facts, data, contexts, and relationships relevant to the decision problem, i.e., *factual knowledge*. Decision making also involves the use of implicit domain knowledge from domain experts, i.e., *expert knowledge*. Computational tools for decision support usually incorporate expert knowledge from domain experts together with specific factual knowledge (Bonczek, Holsapple and Whinston, 1981; Holtzman, 1989).

Factual knowledge in most decision domains is complex, ill structured and incomplete, which makes it difficult to be fully understood, formalized and incorporated into a computational system (Bonczek, Holsapple and Whinston, 1981; Turban and Aronson, 1998; Sprague, 1993). On the other hand, expert knowledge acquisition from domain experts is not an easy task either. Early attempts in building expert systems revealed the difficulties in acquiring expert domain knowledge (Hayes-Roth, 1983; Tecuci and Kodratoff, 1995; Buchanan and Feigenbaum, 1978; Lenat, Prakash and Shepherd, 1986).

Besides incorporating domain knowledge, decision support systems are also required to perform some type of intelligent behaviour such as learning and reasoning about the knowledge they possess in order to support decision making.

The need to incorporate domain knowledge and intelligent capabilities in decision support systems have been identified in various forms by many researchers, such as Simon (1977), Turban (Turban and Aronson, 1998), and Sprague (1993) among others.

For instance, Simon (1977) identifies three stages in a decision making process: *intelligent*, *design* and *choice*. The capabilities of learning and reasoning are embodied in these stages, where the intelligence stage relates to the *search for conditions* that demand decisions and *identifying* opportunities; the design phase relates to *developing and analysing* possible courses of action,

which includes activities such as understanding the problem, assessing and testing solutions; and the *choice* phase is concerned with selecting a particular action or category.

Turban (Turban and Aronson, 1998) and Sprague (1993) introduced a decision support framework composed by various subsystems, including a data management subsystem that contains relevant data about a specific problem; a model management subsystem providing analytical capabilities; a knowledge management subsystem that provides intelligent capabilities, and a user interface subsystem. The knowledge management subsystem provides expertise to the DSS when dealing with complex situations; this expertise can be implemented by an expert system, by artificial neural networks or by some other intelligent system. A DSS that includes such knowledge management subsystem is recognized as an IDSS according to Turban (Turban and Aronson, 1998).

Teng (Teng, Mirani and Sinha, 1988) and Turban (Turban and Aronson, 1998) proposed an architecture for an IDSS in which a knowledge acquisition subsystem is linked to an intelligent supervisor, which is implemented through an inference engine. This architecture also comprises a model base and a database management system.

An architecture to support the decision making process by combining a case base, a database, and a rule base into an intelligent advisory system was proposed by Burstein (Burstein et al., 1998). The proposed architecture is built around a collection of organizational knowledge to make it accessible to decision makers. The aim of this architecture is to support decision making by its capacity to recall past decisions and use this historical knowledge in new decision problems.

The concept of model-based decision support relates to systems for decision support that incorporates three stages: *formulation*, *solution* and *analysis* (Shim et al., 2002). Formulation relates to the generation of problem and domain models, in a way they are acceptable to a model solver. Solution relates to the algorithmic solution of the model. This includes the use of technologies to effectively solve the problem; these technologies constitute combination of techniques from artificial intelligence and operational research to address complex problems. Analysis stage relates to the analyses and interpretation of model's solution and outcomes. This includes report generators, graphical displays, sensitive and "what-if" analysis.

It can be observed in the DSS models introduced above that they all incorporate a domain knowledge component (through case bases, rule bases, knowledge acquisition subsystem, or data management subsystem) and an intelligent system component (through an intelligent advisory system, intelligent supervisor, or knowledge management subsystem). Consequently, it is possible to observe that some of the main features to incorporate in DSS models are *domain knowledge* and *intelligent capabilities*.

The concepts of intelligence and intelligent capabilities used in this research draws from the field of artificial intelligence, which argues that the intelligent behaviour presented by an intelligent system relates to the abilities of gathering and incorporating domain knowledge, learning from acquired knowledge, reasoning about such knowledge and, when required, being able to issue recommendations and justify outcomes (Schank, 1982; Newell and Simon, 1972).

Thus, the required intelligent capabilities in an IDSS model can be summarized as follows:

- Incorporating specific domain knowledge, both factual and expert knowledge
- Learning and reasoning
- Issuing recommendations
- Drawing justifications

A possible approach for domain knowledge acquisition is to automatically induce specific domain knowledge directly from raw data (Fayyad et al., 1996; Quinlan, 1993; Tecuci and Kodratoff, 1995; Wu, 1995; Catlett, 1991). This approach is particularly interesting in data rich domains where large organisational databases are available. Potentially, large organisational databases contain useful information that can be used for decision making purposes, identifying strategically important information patterns (Fayyad, Mannila and Ramakrishman, 1997), such as cases representing previously experienced problem situations. Knowledge discovery in databases (KDD) is the process of extracting useful patterns and models from raw data, and making those extracted patterns understandable and suitable for the resolution of decision problems. KDD is a multi-stage process, in which data mining can be considered the core activity (Han, 1998), and it relates to the process and the set of techniques used to find (mine) underlying structure, information and relationships in normally large amounts of data.

Intelligent computing technologies have been applied in developing computational systems to support a wide range of problems, incorporating intelligent behaviour in these systems. For example, artificial neural networks (NN) have been explored to implement flexible learning and reasoning mechanisms into computational systems, such as decision support systems (Wang, 1994). NN excels in learning in uncertain or unknown environments and in performing approximate reasoning, regarding its numeric, association, and self-organizing nature (Medsker, 1995; Sun, 2001).

Most of the current literature about KDD relates to the development and optimisation of algorithms or experiences of KDD in practice, but relatively little work has been published relating integrated approaches of KDD and intelligent computing in the context of decision support.

The research conducted and described in this thesis is concerned with investigating the combination of knowledge discovery and intelligent computing technologies, in particular artificial neural networks, in developing a framework for decision support.

From this perspective, this research has investigated how *data mining* and *neural networks* can cooperate in order to minimize problems related to *knowledge acquisition*, *reasoning*, and *learning* in building decision support systems.

1.3 Research Question and Objectives

The formulated research question of this thesis is:

What are the components of a framework for intelligent decision support system capable of:

- *utilizing organizational databases as source of information*
- *facilitating automatic knowledge acquisition from those organizational databases*
- *reasoning and learning upon this knowledge*

to support decision making?

In order to answer the research question, the objectives of this thesis are:

- To devise a framework for intelligent decision support system (IDSS) capable of automatically building specific domain knowledge from data rich domains and applying this knowledge in problem solving
- To specify the appropriate technological components of that framework
- To empirically verify the framework in practice.

As a result of the investigation developed in this research, this thesis proposes a new framework for decision support combining an association rule generator algorithm for data mining with an artificial neural network based system in a hybrid architecture. This is called the DM-NN model.

Data mining is applied to induce expert domain knowledge from organizational databases, minimizing the problem of knowledge acquisition and at the same time facilitating the use of organizational databases for decision support. The association rules discovered are stored in rule based knowledge bases, which are accessed by a neural network model for learning purposes.

A neural network based system is introduced to provide intelligent capabilities, in particular, learning and problem solving, taking advantage of the generalization capability of neural networks, as well as the capability of handling large combinations of data, and coping with noise data (Medsker and Liebowitz, 1994).

One of the motivations for combining data mining (DM) and neural networks (NN) is that the IDSS model could benefit from the large amount of available data in organisational databases as a source of domain knowledge to facilitate knowledge acquisition. A second reason was that by combining DM and NN, we could benefit both from the logical and cognitive nature of symbolic systems through an association rules formalism for knowledge representation, and from the numeric, adaptive and self-organizing nature of NN for learning and reasoning purposes.

Additionally, the IDSS model aims to benefit from the whole KDD process by selecting cases from large organisational databases in a way that ensures data quality.

It is also part of the motivation of this research to continue with this author's previous work in which artificial neural networks and symbolic structures were combined in prototyping decision support systems (Viademonte, 1994; Viademonte, Leão and Hoppen, 1995; Viademonte, Hoppen and Beckenkamp, 1997; Pree, Beckenkamp and Rosa, 1997).

1.4 Contributions

The contributions of this thesis are concentrated in the areas of decision support systems (DSS), intelligent systems (IS) and knowledge discovery from databases (KDD). This thesis is an applied research and aims to contribute to both theory and practice.

This thesis adds to decision support theory by proposing a framework that represents a novel approach for decision support systems, suitable for data rich domains. It defines the computational architecture of the framework for DSS, its components, their operation and interactions. The application of the proposed approach in the development of an industry decision support system for aviation weather forecasting demonstrated its feasibility in a practical situation, and contributes to the practice of decision support systems development.

This thesis also contributes to the theory of intelligent systems, by proposing a novel architecture combining an association rule generator algorithm for data mining and a neural network model based system. This hybrid architecture implements learning, reasoning, recommendations and explanatory capabilities through a neural network environment, and facilitates automatic knowledge acquisition from selected cases through data mining. The development of the DM-NN hybrid approach contributes to the practice of hybrid systems development.

This research contributes to KDD practice by providing a description of the complete knowledge discovery process from both a practical and an analytical point of view. The data preparation procedures inherent in the KDD process to ensure data quality have been carefully developed in this thesis; which provides contributions to data pre-processing, in particular for the application domain selected in this research. Furthermore, a deep study of sampling design for data mining was undertaken and a specific sampling strategy was implemented, giving contributions in sampling design for KDD purposes.

As part of this research, the classificatory performance of diverse data models were empirically assessed, through experiments using different levels of confidence degrees, support degrees and rule order. These experiments provide guidance in selecting thresholds in data mining applications using association rule generators or similar algorithms.

1.5 Research Design

This research constitutes an exploratory and empirical research, which includes the development of a framework for decision support. The research design of this thesis includes the process that was developed to address the research question and objectives, and the performance assessment method that was applied.

1.5.1 Research Method

This project is an interdisciplinary, exploratory and applied research, which combines concepts from decision support systems, knowledge discovery in databases and intelligent system technologies, particularly NN.

The research method adopted in this thesis draws on methods used in information systems and artificial intelligence research (Cohen, 1996), in particular, the system development research method described by Nunamaker (Nunamaker, Chen and Purdin, 1991) in conjunction with the development cycle methodology proposed by Medsker (Medsker and Liebowitz, 1994).

Nunamaker (Nunamaker, Chen and Purdin, 1991) proposed a system development approach for information system research consisting of five well defined stages. These include the construction of a conceptual framework, the development of a system architecture, system analysis and design, system implementation, system observation and evaluation. According to this method a conceptual framework is the design of a system to demonstrate the validity of the solution, based on the suggested new methods, concepts and technologies. A system architecture places the system components into the correct perspective, specifies the system functionalities, and defines the structure relationships and dynamic interactions among system components. The design phase involves the understanding of the studied domain, the application of scientific and technological knowledge, the creation of alternatives, and

evaluation of proposed alternative solutions. Building the system is used to demonstrate the feasibility of the design and usability of the functionalities of a system project.

The development cycle methodology was proposed by Medsker (Medsker and Liebowitz, 1994). It particularly addresses the development cycle of NN based systems and models, and includes four main phases: concept, design, implementation and maintenance. In the concept phase an application domain is assessed and an appropriate NN architecture is selected according to the characteristics of the application. In the design phase the system is developed, a development environment is selected, the NN topology and learning algorithms are defined, data are gathered and prepared, and testing and training datasets are generated. The implementation phase concerns the NN execution, training and validation. And the maintenance phase consists of evaluating the system performance and adapting it when necessary.

These stages of the system development research method can be identified in the research design applied throughout this research. The construction of a conceptual framework for IDSS, and the definition of the framework's architecture and its technological components relates to the first stages of the system development research method. The framework development, implementation and evaluation in aviation weather forecasting relates to the three last stages of the system development research method.

As the proposed framework for IDSS includes a NN component, the development cycle methodology defined by Medsker (Medsker and Liebowitz, 1994) was applied in defining the NN environment (Chapter 6 discusses the NN model application in detail).

1.5.2 Outline of the Research Design

The research design of this thesis is concerned with technological components and their combination in a single framework for decision support. It aims to provide the technological infrastructure to implement the activities identified in building an IDSS, e.g., domain knowledge acquisition and intelligent capabilities.

The research design that was applied comprises four main phases. The first phase relates to the conceptual studies and literature review; the second phase to the development of a conceptual framework for IDSS defining its components and architecture. The third phase

relates to the implementation of such conceptual framework in an industry problem, and the fourth phase comprises the final discussions about the developed IDSS model and the conclusions of this research.

Figure 1.1 illustrates the research design of this thesis, identifying each research phase and its respective chapters.

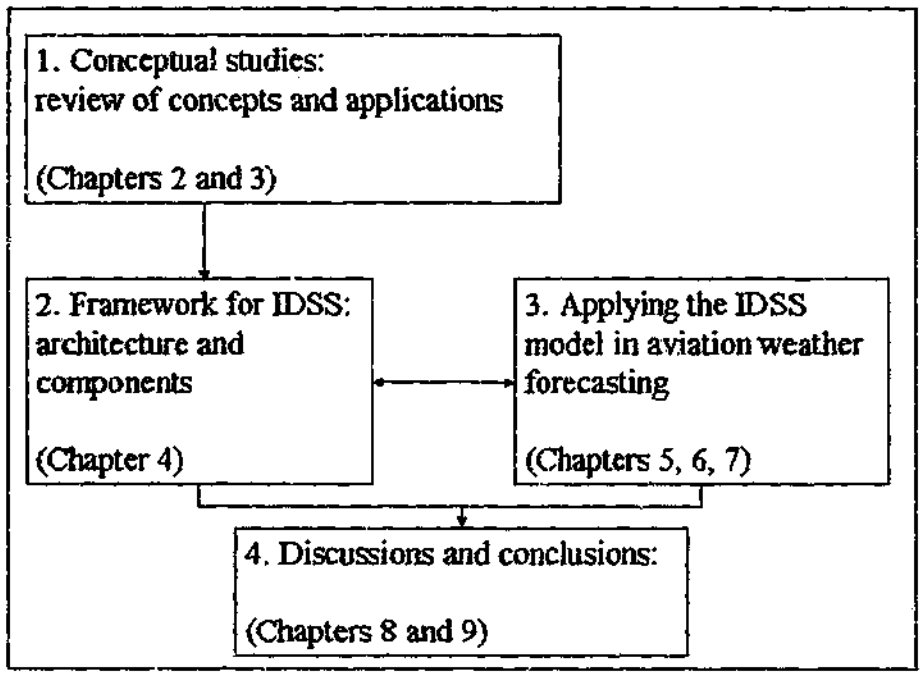

Figure 1.1: Research design

The first phase of this research was dedicated to conceptual studies and investigation of the relevant literature. The following topics were covered in this phase:

- Intelligent Decision Support Systems
- Intelligent Systems, including hybrid systems and symbolic-connectionist architectures, and their applications in practice
- Knowledge Discovery in Databases

As a result of these studies, a conceptual framework for IDSS was designed and implemented.

In the next phase, a particular industry problem was approached and studied, with the aim of verifying its suitability to evaluate the IDSS framework. A computational model was

designed, and technological requirements were assessed in order to implement that framework.

The computational model was implemented, applied and evaluated through the selected industry problem. This phase comprises three distinct stages: the knowledge discovery stage, the stage in which the neural network system was applied, and the assessment of performance.

In the knowledge discovery stage issues about data requirements, selection and modelling were addressed, including the activities related to data preparation for KDD, such as data preprocessing and data partitioning. The next stage relates to the application of the neural network component, where issues of domain modelling, learning, and reasoning were performed. The last stage presents and discusses the assessment of performance of the developed model for IDSS in the selected industry problem.

Each chapter covers part of the research phases developed along this thesis. Chapter 2 and 3 cover the conceptual studies and literature review; Chapter 4 describes the proposed framework, its technological components and interactions. Chapters 5, 6 and 7 are dedicated to the development of the IDSS model in the selected industry problem, i.e., aviation weather forecasting. Although these chapters cover distinct stages of the experiment developed in this research, together they form the whole theme of the IDSS model application in aviation weather forecasting. Chapter 8 presents a discussion about the proposed model for IDSS, and Chapter 9 presents the conclusions and contributions of this research.

The DM-NN model for IDSS proposed in this research has been applied in aviation weather forecasting, identifying rare and severe weather events at airport terminals, particularly fog occurrence.

1.5.3 Performance Assessment Method

The performance of the proposed model for IDSS has been assessed according to its capability to correctly classify meteorological observations, specifically fog cases in the context of aviation weather forecasting. It is a quantitative approach where the holdout method was employed (Weiss and Indurkhya, 1998). The system has been evaluated using a database provided by the Australian Bureau of Meteorology (BOM), with 49901 weather observations

from July 1970 until June 2000, taken at Tullamarine Airport in Melbourne (henceforward Tullamarine).

In the holdout method the data is randomly partitioned into two mutually exclusive subsets, named *training* and *testing* datasets. A given learning system is first trained using the training set and tested on the testing set, and the estimation of accuracy is based on the percentage of correctly classified cases and error rates in the testing set (Weiss and Indurkhya, 1998).

The DM-NN model comprises two main phases in its execution. First, descriptive models of the application domain are generated using *data mining* technology (*descriptive method*). Second, predictive models are generated with the application of *neural networks* (*predictive method.*) Several training sets were generated, each one resulting from a specific sampling proportion and configuration of data mining parameters. The design decisions in building the descriptive models have to be taken into consideration when assessing the performance of the proposed model.

The performance assessment approach employed in this research addresses the developed sampling design, the levels of rule confidence degree, rule support degree and rule order used during the data mining experiments in aviation weather forecasting. These issues impact on system performance, and consequently have to be taken into account when assessing that performance.

Furthermore, it is important to remember that this research does not propose new data mining or neural network algorithms, neither does it propose algorithm optimisations. As such, it does not claim any contribution to algorithm development and optimisation. The main focus of this research is the *combined* approach, with the DM-NN model performance as a whole, its usefulness, suitability and effectiveness in the selected decision making problem, rather than the performance of a single technology on its own. The contribution of this research lies in the novelty and the effectiveness of the proposed framework for decision support.

Additionally, it should be noted that the DM-NN model is proposed as an *interactive* and *evolutionary* framework for decision support. It is interactive because the participation of the

user (the decision maker) is important in various moments, e.g., in selecting cases from raw data, selecting features and discretization ranges, and tuning different data mining parameters. The proposed model is an evolutionary model, as NN technology is evolutionary by its nature through its adaptive capabilities. An interactive execution cycle is expected in most NN applications until the system achieves a stable learning state. As the environment changes, the model has to adapt to these changes through another learning cycle.

Consequently, the results obtained through the assessment of classificatory performance have to be analysed taking all these characteristics into account. These results illustrate the usefulness of the DM-NN model, especially its suitability and effectiveness in the selected decision making problem, rather than validate any particular algorithm.

1.6 Structure of the Thesis

This thesis is organized into 9 chapters comprising the research phases previously outlined in section 1.5.

The Introduction, Chapter 1, presented an overview of this research, its theoretical background, objectives, research question and motivations. It also presented the contributions, the research design, the research method being used and discusses the performance assessment method that has been employed. Finally, the structure of the thesis is described.

Chapter 2 introduces the concepts and research related to intelligent decision support systems. These concepts are introduced in two topics: intelligent systems and hybrid symbolic-connectionist systems. The main reason for doing this is that these two types of hybrid architectures are related to this research project.

The topic about intelligent systems introduces hybrid architectures combining different intelligent computing technologies. The topic about symbolic-connectionist systems introduces hybrid architectures combining connectionist and symbolic approaches.

Chapter 3 introduces concepts about knowledge discovery in databases and data mining, as well as issues about data preparation, dimensionality reduction, and data sampling.

Chapter 4 introduces the proposed hybrid DM-NN model for IDSS, its components, their respective roles and interactions. The applied knowledge representation schema, the employed neural network environment and neural network model are also discussed in this chapter.

The development and application of the proposed DM-NN model for IDSS in aviation weather forecasting comprises three stages. First the knowledge discovery stage is presented; next the intelligent advisory system stage is described, and lastly the system performance is discussed. Chapters 5, 6 and 7 describe this application.

Chapter 5 introduces the problem of aviation weather forecasting, and describes the stage of discovering knowledge from a meteorological database. The activities of data preparation, domain modelling, data sampling, and data mining are presented in this chapter.

Chapter 6 describes the intelligent advisory system stage, which is implemented through a neural network environment. Issues of domain modelling, learning, reasoning and user interaction are discussed.

Chapter 7 presents and discusses the analysis of performance of the DM-NN model in the context of aviation weather forecasting. Chapter 8 presents a discussion of intelligent systems related to the DM-NN model, and discusses some limitations of the developed model.

Finally, Chapter 9 presents the conclusions, as well as a summary of contributions and directions for further research.

1.7 Chapter Summary

This chapter introduced the research developed in this thesis. It gave an overview about the research project, and introduced its theoretical background, as well as its objectives, the formulated research question, motivations and contributions. This chapter also presented the applied research method and design, and the assessment method employed to evaluate the performance of the computational model for IDSS.

Chapter 2

2 Intelligent Decision Support Systems

This chapter introduces the concepts and research related with intelligent decision support systems. These concepts are introduced in two topics: Intelligent Systems (IS) and Hybrid Symbolic-Connectionist Systems (HS). This chapter also introduces the concepts of artificial neural networks (NN), including NN structures and learning approaches. Some of the most well known NN models are also briefly introduced.

2.1 Introduction

In the context of this thesis intelligent decision support systems (IDSS) are computer-based decision support systems capable of acquiring domain knowledge and performing learning and reasoning about the knowledge they possess.

To implement such intelligent capabilities intelligent computing technologies are employed. Intelligent computing technologies comprise diverse disciplines, such as artificial neural networks (NN), rule induction (RI), fuzzy logic (FL), genetic algorithms (GA), case base reasoning (CBR) and expert systems (ES) in building systems that exhibit a type of intelligent behaviour such as learning and reasoning (Zurada, 1992; Zadeh, 2000; Medsker, 1995; Sun, 1995b).

2.2 Intelligent Systems

Although the term *intelligent system* is largely used, there is no commonly accepted definition for the concept of intelligent systems. Often the terms *intelligent systems*, *hybrid systems*, *soft*

computing systems and *computational intelligence* are interchangeably used. The concept of intelligent systems varies according to different literatures.

According to Azvine (Azvine, Azarmi and Nauck, 2000) intelligent systems are computer systems based on symbolic artificial intelligence combined with hybrid techniques from the soft computing paradigm (Zadeh, 1973). Such a hybrid approach combines methods from disciplines such as fuzzy set theory, neuroscience, agent technology, knowledge discovery, and symbolic artificial intelligence. The aim of such a combination is to provide robust, adaptive and easily usable systems.

Goonatilake (Goonatilake and Khebbal, 1995) associates intelligent systems with adaptive machines, built through the combination of different types of intelligent computing technologies, such as neural networks, genetic algorithms and fuzzy logic.

Kandel (Kandel and Langholz, 1992) defines intelligent systems as hybrid computational architectures integrating the computational paradigm of expert systems and neural networks.

Bezdek (1994) presents a discussion in which the term "intelligent systems" is related with the concepts of artificial intelligence and computational intelligence. According to Bezdek (1994):

> *"a system is computationally intelligent when it deals with numerical (low-level) data, has a pattern recognition component, and when it begins to exhibit computational adaptivity, fault tolerance, speed approaching human-like turnaround, and error rates that approximate human performance."*

Additionally, Bezdek defines an artificially intelligent system as a computationally intelligent system that incorporates knowledge in a non-numerical way, has the ability of learning and dealing with new situations.

Although different literatures present distinct concepts about intelligent systems, it can be observed that the idea of combinations of diverse technologies such as neural networks, fuzzy systems, and evolutionary computing is a fundamental concept in these systems. Additionally, the notion of systems capable of reasoning, learning and making decisions in uncertain environments is inherent in the concept of intelligent systems.

Medsker (1995) provides a diagram introducing the various components of intelligent systems. Figure 2.1 illustrates such a diagram:

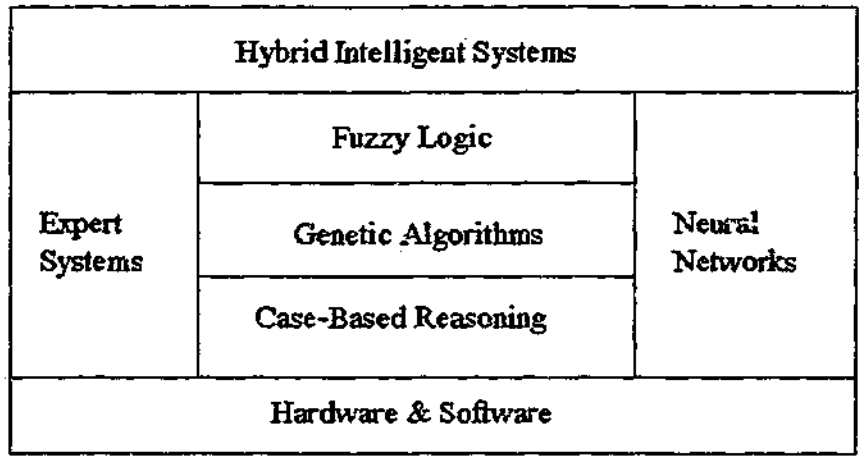

Figure 2.1: Intelligent systems components as in (Medsker, 1995)

This thesis adopts the concept of intelligent systems as computational architectures that combine diverse technologies to provide computerized systems with capabilities of learning, reasoning and decision making in uncertain environments. Furthermore, this research borrows from Medsker its theoretical framework for intelligent system.

2.2.1 Motivations of Intelligent Systems Development

The emergence of intelligent computing technologies and their respective combinations have their origins in different sciences. Contributions and motivations have come from researches and developments in decision analysis, cognitive science and artificial intelligence (AI), and from the development of computerized information system applications in different domains such as industrial control, financial modelling and cognitive modelling (Sun, 1995b; Goonatilake and Khebbal, 1995; Zurada, 1992).

Complex industrial and business problems have partially driven the development of intelligent computing technologies, to support the solution and management of complex tasks, such as weather forecasting, medical diagnoses, air traffic control, trading decision in financial markets, credit scoring and risk assessment in financial and insurance corporations and a myriad of other industrial applications (Goonatilake and Khebbal, 1995; Medsker, 1995; Sun, 1995b). These complex tasks often require computational systems with special capabilities such as interpreting incomplete readings, coping with ill defined and often conflicting information, using sets of compiled knowledge and heuristics to infer and approximate conclusions, handling time-dependent information, and in many situations being able to reach

satisfactory conclusions rather than optimal ones (Zadeh, 1973; Goonatilake and Khebbal, 1995).

The expert systems developed from the 1970s onwards are examples of computer systems applied to support complex tasks. Expert systems such as DENDRAL (Buchanan and Feigenbaum, 1978), which was developed to infer the molecular structure of unknown compounds from mass spectral; MYCIN, developed at Stanford Medical School (Turban and Aronson, 1998) and applied in the diagnosis and treatment of blood infections; and XCON (Barker et al., 1989) used at DEC (Digital Equipment Corp). for computer configurations, are examples of such systems.

The search to understand and explain human cognitive processes is the other driving force motivating intelligent systems research and development. Decision makers, engineers and computer scientists have attempted to model and develop powerful computer-based decision support tools (Holtzman, 1989), and cognitive scientists have attempted to model, verify and test cognitive theories and models through computer simulations (Newell, Shaw and Simon, 1958; Colby, 1965; Reitman, 1965; Colby, Watt and Gilbert, 1966; Weizenbaum, 1968), and (Newell and Simon, 1972). Theories of human cognitive processes have been implemented in computer programs and tested to assess if they work. If the computer program produces the same behaviour as a human, then the series of operations it implements may be considered as an accurate representation of human cognitive processes (Mayer, 1992).

Some AI projects developed during recent years are examples of such attempts. One is example the Logic Theorist, a theorem solver program developed by J. Shaw, A. Newell and H. Simon in 1956. Logic Theorist was not only designed for logic theorem proving, but also to verify how humans perform similar tasks by selecting appropriate heuristics rather than using an extensive search process (Crevier, 1993). The General Problem Solver (GPS) (Newell and Simon, 1963; Ernest and Newell, 1969) is another example. GPS aimed to demonstrate that certain general human reasoning procedures are not involved in any particular task, but encoded in a completely task independent manner. Furthermore, these general procedures could be implemented in a computer program able to solve a wide variety of different problems (Crevier, 1993; Mayer, 1992). Another example of cognitive modelling through

computer experiments is the Feigenbaum (1963) study about human learning. Feigenbaum built the EPAM (Elementary Perceiver and Memorizer) program to emulate how people memorize nonsense syllables. The Automated Mathematician program developed by (Lenat, 1981), which learned by discovering mathematical concepts, is another example of cognitive process implemented through a computer program. One of the most significant projects is the Cyc (short for encyclopedia) that attempted to implement a large amount of common sense knowledge through a large collection of self-evident facts stored in a complex hierarchy of a frame based knowledge representation mechanism (Lenat, Prakash and Shepherd, 1986; Lenat and Guha, 1988).

Symbolic AI systems (systems that apply a symbolic approach to represent and manipulate knowledge) have been successfully applied in specific and well-defined problem domains, such as expert systems in classification tasks, planning, and scheduling (Azvine, Azarmi and Nauck, 2000). However, there are some drawbacks, for example symbolic systems have difficulties in taking into account uncertainty, imprecision and vagueness. In much the same way a single approach is not enough to explain complex cognitive processes, it is also not enough to solve many complex industrial problems.

Rather than a single approach, some problems require a combination of tools, theories and technologies that can be applied under different circumstances. The necessity of a combined approach where two or more techniques are combined in a way that overcomes the limitations of individual techniques is the driving force behind intelligent system research and development.

2.2.2 Intelligent System Capabilities

Intelligent systems aim to overcome a series of limitations that have been identified in information systems applying a single technique. Some of these limitations include (Medsker, 1995; Sun, 1995b):

- The knowledge acquisition bottleneck, i.e. the necessity of a consultation with human experts for knowledge acquisition
- The need to synthesize new knowledge
- The need to dynamically modify knowledge whenever necessary

- Difficulties in handling imprecise, incomplete and uncertain information
- Difficulties in implementing reasoning capabilities
- Difficulties in coping with the complexity of practical problems.

While systems using a single approach succeed in overcoming some of the limitations mentioned above, they normally fail in implementing solutions to more than one of those problems. Each technique has particular properties that make them suitable for particular problems and not for others. For example, rule base systems are good at transparent knowledge representation, but have difficulties adapting to changes in the environment in which they are applied.

Goonatilake (Goonatilake and Khebbal, 1995) identified four key information-processing capabilities that can be handled by intelligent systems. These are: knowledge acquisition, brittleness, higher and lower level reasoning and explanation.

Knowledge acquisition concerns building a knowledge base, which is recognized as a fundamental activity in the development of intelligent systems. The traditional approach for knowledge acquisition from human experts brings several disadvantages, such as the requirement of human experts, and it is a time consuming and expensive activity.

Techniques such as neural networks, genetic algorithms and rule induction systems, which can generate rules or decision trees from raw domain data, have been successfully applied to automate the knowledge acquisition process in the development of intelligent systems.

Brittleness refers to the problem in which a system operates efficiently only in narrow domains under normally limited operational conditions. Slight changes in the domain require human intervention to accommodate the system (Holland, 1986). The main causes of brittleness, according to Goonatilake (Goonatilake and Khebbal, 1995), are the use of inadequate knowledge representation structures and reasoning mechanisms.

An example of the brittleness problem is a case when a system cannot cope with inexact, incomplete or inconsistent data, or when failures in a single processing component make the system non-operational. The distributed representation and reasoning of neural networks deal well with incomplete data and also allow the system to gracefully degrade. This means a neural network based system does not became non-operational when part of the neural network fails

(Freeman and Skapura, 1991). Besides neural networks, fuzzy logic and genetic algorithms are also alternatives to handle the brittleness problem.

Higher and lower level reasoning concerns the capability to execute tasks that require different levels of reasoning. Low level reasoning is characterized by parallel, distributed, and normally numeric processing. Pattern recognition tasks, such as understanding spoken language, recognizing faces and recognizing a large variety of colours are examples of human lower level reasoning. These activities require rather complex mental processing. Although we naturally perform those activities all the time, if asked to explain the reasoning processing involved we would have enormous difficulty to formally describe them.

At the other hand, high level reasoning relates to sequential, logical and normally symbolic processing. Activities like goal-oriented planning, scheduling, and vehicle navigation are examples of high level reasoning.

Expert system and symbolic based systems in general perform well in implementing higher level reasoning, as in medical diagnosis applications, but their ability to implement low level reasoning has been limited. Neural networks, for example, excel in recognising complex patterns, like hand-written characters or profitable loans in financial trading decisions (Goonatilake and Khebbal, 1995); both of which are too complex to be implemented through a symbolic approach.

Explanation concerns the ability to clarify to users how the system reasoning process works. Explanation capabilities are required not only for understanding of the solutions generated and recommended by the system, but also for the purpose of evaluating the system's reasoning and outcomes. It frequently works as a debugging feature for knowledge engineers and system developers.

In rule-based systems for example, explanation capability is normally implemented through a backtracking process, in which the system backtracks the rules that were fired during its reasoning process. Systems applying other approaches to knowledge representation, such as frames and semantic nets, have to implement more sophisticated explanation algorithms; and a much more complex solution is required in neural networks based systems, as in this case the system knowledge is distributed across the whole neural network structure.

Detailed discussions about explanatory features, especially in expert systems, can be found in (Clancy, 1983; Ye and Johnson, 1995; Hayes-Roth and Jacobstein, 1994). Additionally, (Zanella, Piero and Guida, 1997) and (Zanella and Gianfranco, 1999) provide a valuable discussion about explanation and justification needs in decision support systems and complex computer systems in general.

Besides the information processing capabilities described above, Azvine (Azvine, Azarmi and Nauck, 2000) includes the following features as required capabilities in intelligent systems:

- Applicability under uncertain and vague conditions
- Adaptability, e.g. learning from experiences and human intervention
- Autonomy, e.g. the capacity of acting on behalf of a user without direct intervention
- Multi-modal interfaces, such as vision, speech and natural language understanding

Moreover, Zadeh (1994) emphasizes that the guiding principle of soft computing is:

"To exploit the tolerance for imprecision, uncertainty and partial truth to achieve tractability, robustness, and low solution cost."

There is general agreement in the literature that has been discussed in this chapter that the ability to *handle the knowledge acquisition problem*, the ability to *cope with imprecise, noise and uncertain data*, the ability of *learning, reasoning, adaptation*, and presenting some kind of *explanation facility* are fundamental capabilities in intelligent systems.

2.2.3 The Role of Intelligent Computing Technologies in Building Intelligent Systems

Diverse computing technologies have been applied in developing intelligent systems, as mentioned in section 2.2. This thesis concentrates on the technologies of symbolic systems and artificial neural networks, with a brief introduction to fuzzy logic and genetic algorithms, which are also core technologies in the intelligent systems *toolbox*. While these technologies have produced encouraging results in specific tasks, certain complex problems like knowledge representation, common sense reasoning, real time problem solving, vision and language processing are too difficult to be solved by a single technique. Each of these techniques has

strengths and weakness; this section briefly introduces each of them and their respective roles in building intelligent systems.

Symbolic systems, such as expert systems, apply a symbolic approach to represent and manipulate knowledge. A mechanism of symbol structures, such as *production rules, semantic nets* and *frames* (Bonissone et al., 1999) is used to represent the domain knowledge. Moreover, in symbolic systems the knowledge manipulation and inference are implemented by a sequential processing of these symbolic structures. For that reason those systems are known as *symbolic systems,* symbol-processing AI, classical AI or even symbolic AI.

Symbolic systems are good at transparent and local knowledge representation, and also to implement explanation facilities. Symbolic knowledge representation mechanisms allow explicit declarative knowledge representation. On the other hand, they are not easily adaptive to changes in the environment, do not increase or learn with experience, do not represent generalization capabilities and thus are unlikely to allow graceful degradation, and normally require extensive involvement of experts for knowledge acquisition.

Neural networks constitute a set of processing units (also called neurons) connected in a non-linear fashion. The information processing in neural network models is performed in a parallel and distributed way (section 2.4 introduces neural networks in more detail). Neural networks excel in learning and self-organizing capabilities, generalization, interpolation and use of partial information; they are also fault and noise tolerant. However, in neural network models it is difficult to implement explanation facilities, because they do not have explicit and declarative knowledge representation structures. Domain knowledge in a neural network model is encoded incomprehensibly in a numerical fashion as weight vectors within the trained NN topology, and hence cannot easily be accessible (Medsker and Liebowitz, 1994).

Fuzzy logic is a superset of conventional (Boolean) logic that has been extended to handle the concept of partial truth - truth-values between "completely true" and "completely false." It was introduced by Dr. Zadeh in 1964 as a means to model the uncertainty of natural language; fuzzy theory is a methodology to generalize any specific theory from a crisp (discrete) to a continuous (fuzzy) form.

Fuzzy logic provides a means for representing vague and uncertain information, normally expressed in semantic terms, such as *high, higher, medium, low, lower, cold, hot, warm, very cold, very hot, likely, unlikely,* etc. These semantic terms (or variables) are used in everyday life for many purposes, such as decision situations. Semantic variables lack a precise definition, and often overlap each other, hence it is very difficult to properly model and incorporate semantic variables in computerized decision support systems.

For example, in a DSS in a hydroelectric power plant (Viademonte, Hoppen and Beckenkamp, 1997), a decision variable identified as *water flow rate* could assume three attribute (or classes): *high* when the water flow is higher than 40 $m^3/s/h$, *medium* when the water flow falls between 20 and 40 $m^3/s/h$, and *low* when the water flow is less than 20 $m^3/s/h$. Instead of assuming a sharp transition between the *water flow rates*, it can be assumed that the flow rate does not abruptly change between two classes, but gradually shifts from one point to another, so that there are grey areas in which a particular flow rate value might overlap between two classes. This can be illustrated by the value 43 $m^3/s/h$, which falls into the classification *high*, and "partially" falls into the class *medium*. Figure 2.2 illustrates this property.

According to Figure 2.2, the flow rate 43 $m^3/s/h$ belongs to the class *high* with 0.8 pertinence degrees (or confidence degrees), and also belongs to the class *medium* with 0.4 pertinence degrees.

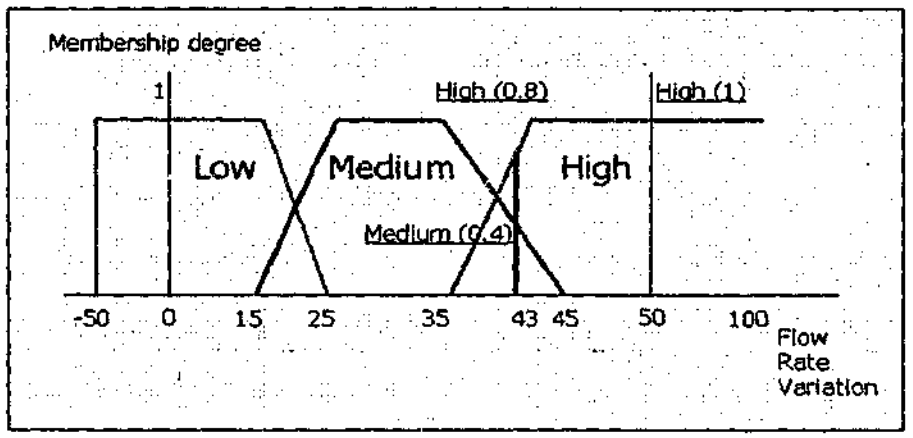

Figure 2.2: Fuzzy modelling for flow rate variation variable as in (Viademonte, Hoppen and Beckenkamp, 1997)

A knowledge based system might represent this fuzzy relation applying the two classes in different decision rules, for an example refer to (Viademonte, Hoppen and Beckenkamp,

1997). Otherwise, assuming a sharp transition between flow rate classes represents an over-simplification of a more complex real situation, which may be useless in some decision situations.

Fuzzy logic has been applied in decision support systems and, particularly, in expert systems (so called fuzzy expert systems) representing vague and imprecise information through the use of linguistic variables and inferring through fuzzy if-then rules (Medsker, 1995). Moreover, fuzzy logic is a suitable approach to implement approximate reasoning, incorporate real-world ambiguities and high level of abstractions in problem solving. Systems implementing fuzzy logic have been applied to a large range of problems including fuzzy control, fuzzy expert systems and decision making in stock market forecasting, risk management, targeted marketing, scheduling and weather forecasting, among many others (Cox, 1994; Viademonte, Hoppen and Beckenkamp, 1997; Turban and Aronson, 1998). In decision making situations fuzzy logic is particularly useful since decision making often handles information that is incomplete, vaguely defined, sometimes contradictory and where imprecise human estimation has been applied.

A drawback of fuzzy logic is that it does not offer the means to automatically discover the fuzzy sets and respective fuzzy memberships function, requiring a high level of human expertise and time.

Genetic algorithms (GA) are a computer technique inspired by biological concepts of natural evolution, the survival of the fittest. It is an alternative approach to search methods, based on local optimisations instead of a global optimisation where the main goal is to quickly find an adequate solution for a specific problem (Lawrence, 1991).

GA is basically an interactive procedure over a set of data structures that represent potential candidate solutions for a particular problem. The candidate solutions are called *chromosomes*, and are normally represented by binary strings (strings in which the only values are "1" and "0"). In each interaction, the candidates generate copies of themselves with slight modifications, in an operation called *generation*. The current candidates are then rated for their effectiveness as a solution to the problem at hand. Based on this evaluation, if no candidate is rated good enough for the problem solution, the process is repeated with new chromosomes

being generated. The algorithm follows this process until it finds a chromosome considered a good solution to the problem. During the generation process, GA applies a set of genetic operators, namely *reproduction*, *crossover*, and *mutation*.

In the *reproduction* operation the GA produces new generations of improved chromosomes (solutions) through the selection of parent's chromosomes with higher fitness ratings. In the *crossover* operation two chromosomes exchange part of their segments, which are randomly selected in each chromosome. The *Mutation* operation is rarely executed; its purpose is to avoid a stationary state in the algorithm by arbitrarily shifting values "1" and "0" instead of duplicating these bits of information. GA performs a heuristic search for a local optimal solution in the problem domain space. This optimisation characteristic makes GA an attractive approach for many complex problems, where the search for global solutions in very large solution spaces potentially leads to high computational cost.

Moreover, being able to maintain a population of solutions (chromosomes) makes GA able to cope with brittleness (Holland, 1986), allowing self-organization and adaptation. GA has been applied in problems such as job shop scheduling, distribution, resources allocation, project management, financial forecasting and investment analysis, among others (Lawrence, 1991).

Machine learning explores the mechanisms by which knowledge in a computer system is acquired through experience. Machine learning involves two kinds of information processing: *inductive* and *deductive* reasoning. In inductive reasoning general patterns and rules (rule induction is one type of machine learning approach) are determined from raw data. Deductive reasoning applies general rules to infer specific facts. For instance, knowledge acquisition from datasets is performed through an inductive approach, whereas a proof of theorem is performed through a deductive approach where known axioms are used.

Goonatilake (Goonatilake and Khebbal, 1995) proposed a rating schema for the intelligent techniques introduced in this section, considering the capabilities described in section 2.2.2. According to this schema, expert systems are good at high-level reasoning and explanations; neural networks are suitable for automated knowledge acquisition, coping with brittleness and low-level reasoning; fuzzy systems perform well in coping with brittleness, low-level reasoning

and explanations, and perform reasonably well in high-level reasoning. GA are good at automated knowledge acquisition, and perform reasonably well in coping with brittleness, high and low level reasoning and explanation capabilities. Machine learning technology, mainly rule induction, is a good option for automated knowledge acquisition, and also performs reasonably well in implementing explanations and high-level reasoning. But it does not effectively handle the brittleness problem and low-level reasoning.

2.2.4 Intelligent System Classifications

With the development of intelligent systems combining different techniques, various classification schemes for these hybrid architectures have been proposed. Medsker (Medsker and Bailey, 1992) proposed a classification scheme where hybrid architectures are classified according to their level of integration; this scheme addresses specifically hybrid architectures integrating neural networks and symbolic systems. The identified models are (Medsker and Bailey, 1992):

- **stand-alone**: in this model the symbolic and neural components are independent and there is no interaction between them. This strategy provides a means of comparing the performance of the two techniques in a specific application; additionally, because the techniques do not attempt to interface with each other the model can be quickly developed, and even commercial software packages can be used. However, this type of model does not benefit from the integration of both techniques, and neither the neural component nor the symbolic component can support the weakness of each other.
- **transformational**: this is a similar model to the stand alone model; both techniques are independent from each other. Transformational systems begin in one approach (e.g. neural component) and migrate to another approach (e.g. symbolic), or vice versa; consequently there are just two forms of transformational model: symbolic systems transformed in neural systems or neural systems transformed in symbolic systems. Transformational systems also do not benefit from the integration between the neural and symbolic approaches.

- **loosely coupled:** the systems in this class present an integration strategy where the application is decomposed into the neural component and symbolic components that communicate via data files. It is a sort of weak integration where the components exchange information via external data files. For example, an expert system can be applied in a particular diagnosis application, and the expert system recommendations can be submitted to a trained network model for data analysis or verification. Loosely coupled systems benefit from the integration strategy, but they require additional costs with communications and control of data redundancy and consistency.
- **tightly coupled:** tightly coupled models apply an independent strategy between the symbolic and neural component, where the integration between the techniques is done by memory resident data structures, instead of external data files as loosely coupled models. This integration strategy improves the interactive capabilities and performance of tightly coupled systems. However, the development and maintenance of tightly coupled systems is significantly complex as the internal data interface requires redundant information control and the system validation and verification is complex.
- **fully integrated:** in this model, symbolic and neural components share data structures and knowledge representation, and the communication between them is done by the dual nature (symbolic/subsymbolic) of the hybrid structure. For example, a particular concept is represented as a neuron in the NN topology and by a symbolic structure in the ES, such as a node in a semantic network, a frame instance or an object instance in an object hierarchy. The information between both components is exchanged by changing values and activations in this dual structure. Fully integrated models benefit from the self-adaptive and self-organizing capabilities of connectionist systems, as well as the comprehensible knowledge representation of symbolic systems, providing a full range of capabilities such as adaptation, generalization and explanations. The main limitation of fully integrated models is the complexity in developing the components' interface.

A more general classification is proposed by Goonatilake (Goonatilake and Khebbal, 1995), where not only symbolic systems and neural networks are contemplated, but also a diversity of techniques such as fuzzy logic, genetic algorithms, linear programming and other numeric techniques. Additionally, this classification scheme is more generic as it takes into account system functionalities, processing architecture and communication requirements, rather than just implementation issues as in Medsker's classification scheme. The proposed classes are (Goonatilake and Khebbal, 1995):

- **function-replacing hybrids**: this class relates to the functional composition of a single technique, where a function of a specific technique is replaced by another technique. Here the motivation is to overcome a weak property of a particular technique by combining it with another technique that has strengths in that specific property. For example, using genetic algorithms to automatically define fuzzy membership functions.
- **intercommunicating hybrids**: in this class a specific problem is decomposed in several tasks, where each task is assigned to a different technique to be solved. Each technique comprises an independent and self-contained module that exchanges information with other modules and performs a particular operation. A control mechanism coordinates the individual modules in order to solve the original problem.
- **polymorphic hybrids**: in this class, systems using a single processing architecture are used to implement the functionalities of different intelligent techniques, in this way polymorphic hybrid systems can emulate the functionalities of different processing techniques. The main goal of this class of systems is realising multi-functionality in a single architecture. An example of polymorphic hybrids is the application of neural networks to implement some kind of symbol processing.

Another classification scheme for integrating exclusively neural networks and symbolic systems was proposed by R. Sun (1995b), in which four types of integration are described. This classification scheme takes into account the type of NN models and the level of connection between the NN and symbolic components.

The first class comprehends a localist network[1] used for symbolic processing, where each node in the NN topology represents a particular concept. In this integration strategy the NN and symbolic structures are directly mapped between each other. A second class comprehends a distributed network that is used for symbolic processing, i.e., the neural component performs an equivalent function to symbolic processing. A third class relates to architectures where the NN module and symbolic module are kept separated. A fourth class relates to architectures using NN as basic small elements in symbolic modules, where a symbolic approach is applied, but the elements of this symbolic structure are replaced by small scale neural components.

The *loosely coupled* and the *intercommunicating hybrids* models relate most to the architecture for IDSS proposed in this thesis, where a neural network model (the Combinatorial Neural Model) has been combined with a separate DM module. Furthermore, the employed neural network environment (CANN) relates more to the *fully integrated* models, as the nodes in the neural network are also symbolically represented as objects in an object hierarchy, making a clear distinction between them very difficult.

2.3 Hybrid Symbolic-Connectionist Systems

Hybrid symbolic-connectionist systems are concerned with integrating the symbolic computational paradigm and artificial neural networks (also called the subsymbolic or connectionist paradigm), and investigating methods by which these two approaches can be combined to represent and manipulate knowledge to emulate human-like thought process and reasoning (Kandel and Langholz, 1992).

Sun (1995b) associates symbolic-connectionist systems with systems in which the combination of knowledge representation and learning techniques from both symbolic processing models and connectionist networks models are brought together to tackle problems that neither type of model by itself can handle well, such as the problem of modelling human cognition.

[1] *Localist network* is a concept defined by R. Sun (1995b) that corresponds to a particular hybrid architecture.

Although symbolic and connectionist paradigms share the same cognitive inspiration, e.g. to emulate cognitive behaviour in computing systems (Sun, 1995b), they are based on significantly different conceptual principles. These differences are on three levels: *understanding of cognition, style of processing* and *knowledge representation structures*. The symbolic paradigm assumes the existence of mental representations in biological cognition; hence artificial cognition is modelled through a sequential style processing of symbolic representations (Newell and Simon, 1972), where the operations are performed in a step-by-step fashion over a set of symbolic expressions, such as production rules in a rule-based expert system. Consequently, the symbolic paradigm is mainly concerned with the definition of appropriate symbolic structures for knowledge representation and manipulation. For instance, a particular relation such as "*A isa B*", which means that *A* is an element that belongs to the set of elements defined by *B*, and the individual concepts represented by *A* and *B* can be readily represented through a symbolic mechanism.

Symbolic systems are mainly suitable for applications in well defined and normally narrow domains, where reasoning is performed upon established knowledge structures. In symbolic systems knowledge is stored in knowledge bases separately from the reasoning mechanisms, called the inference machine (Hayes-Roth and Jacobstein, 1994), thus knowledge is modified by updating knowledge bases. Furthermore, these symbolic knowledge structures facilitate the creation of explanation capabilities to help the user understand the system reasoning process.

On the other hand, the connectionist paradigm is inspired by the understanding of biological neural networks where cognition emerges from the massive interaction of a large number of single neurons. Thus, the connectionist paradigm is concerned in developing parallel distributed processing architectures in which knowledge representation and manipulation emerge from the interactions of a large number of single processing units, e.g. *neurons* (McClelland, Rumelhart and Hinton, 1988). This parallel processing style of connectionist systems is the source of their flexibility. Moreover, massive parallelism gives connectionist systems a very robust characteristic, as the computation is spread over many neurons, so that failures in some neurons do not render the whole network unstable or even non-operative. Consequently, connectionist systems handle noisy and incomplete inputs and

allow for graceful degradation within a certain range. Another important feature in connectionist systems is the way knowledge is represented; not as declarative expressions, but numerically distributed across the network through weight vectors between neuron connections (McClelland, Rumelhart and Hinton, 1988). According to Reategui (1997), in such a model, the relation "*A isa B*" cannot be comprehensibly represented, although is possible to indicate that some state of *A* affects states of *B*.

Table 2.1 summarizes some of the weaknesses and strengths of symbolic and connectionist paradigms (Medsker, 1995):

Table 2.1: Contrasting symbolic and connectionist features

Symbolic	Connectionist
Easy to access, and declarative knowledge representation.	Knowledge implicitly represented in a numeric fashion as weights distributed across the connectionist topology.
Captures human knowledge-level concepts.	Analyzes large amount of data and establish patterns in situations where rules are unknown or costly to achieve.
Relative easy to implement explanation features.	Difficult to implement explanation features.
Requires extensive involvement of human experts in performing knowledge engineering.	Capabilities of learning from cases/examples and self-organization.
Inability to easily adapt to changes in environment, by modifying knowledge whenever it becomes necessary.	Easy to adapt to changes in the domain/environment, by (re)training the network with new cases/examples.
Inability to synthesize new knowledge and consequently do not increase in performance with experience.	May require excessive training times.
No generalization or graceful degradation.	Fault tolerant and graceful degradation.
Difficult to handle noisy and incomplete input data.	Copes well with noisy and incomplete data.

Nevertheless, despite their conceptual differences, symbolic and connectionist paradigms are complementary to each other in the way they represent knowledge; through the logical cognitive and mechanical nature of symbolic systems, and the numeric, association, robustness and self-organizing nature of the neural networks (Medsker, 1991).

Symbolic systems are suitable for tasks where declarative knowledge and explicit reasoning are desirable; connectionist systems are best suited for low-level processes, pattern matching, and association memories, or even in situations when declarative knowledge is too costly to achieve or human experts are not available. Combining symbolic and connectionist

approaches potentially yields a more powerful system than either one of its components in a single architecture; mainly when solving complex problems in domains characterized by imprecise, changeable and uncertain information.

2.4 Artificial Neural Networks

Artificial neural networks or connectionist models have been developed based on organizational principles that have been observed in biological neural networks[1]. The study of artificial neural networks began around 1940, and the work of McCulloch and Pitts (1943) is normally referenced as an initial mark in this field. The McCulloch and Pitts paper concentrated on neurophysiology, especially the representation of events in the biological nervous system. The paper describes a logical calculus of biological neural networks, presenting a mathematical analysis of how interconnected cells could perform logical operations by transmitting (or not transmitting) impulses between each other, a mechanism that a structure like the brain could perform. Following McCulloch and Pitts, another major development in neural networks came in 1949 with the publication of Donald Hebb's "The Organization of Behaviour," (Hebb, 1949). Hebb presented a theory of a physiological learning rule for synaptic modification, proposing the idea that brain connections change as we learn different tasks. This is known as Hebb's postulate of learning, which states that the effectiveness of a variable synapse between a pair of neurons is increased by the repeated activation of one neuron by the other across that synapse (Hebb, 1949; Crevier, 1993). In 1952 Ashby published "Design for a Brain: The Origin of Adaptive Behaviour" (Ashby, 1952). Ashby stated that adaptive behaviour is not inborn but learned, and through this learning process a system usually changes its behaviour for the better. This emphasizes the dynamic aspects of living organism and the concept of stability.

In the late 1950's the first connectionist systems appeared, one of these was the perceptron and the perceptron convergence theorem introduced by Rosenblatt, as a result of his work in pattern recognition (Rosenblatt, 1958). Also at that time, Bernard Widrow introduced the least

[1] The terms *artificial neural networks* and *neural networks* are interchangeably used in this thesis; when referring to natural neural networks the term *biological neural networks* is used.

mean-square algorithm and formulated the Adaline (adaptive learning element) neural network.

In 1969 Marvin Minsky and Seymour Papert published "Perceptron" (1969) in which they explored the limited capability of one-layer perceptrons. For the next few years the research and development of NNs remained concentrated in small groups of researchers. Then, in 1982, Hopfield published a paper where he presented the principle of storing information in dynamically stable networks, and introduced a neural network model with a feedback loop, which became known as Hopfield networks (Hopfield, 1982). Soon after, in 1986, Rumelhart, Hinton and Williams (1986) reported the development of the back-propagation algorithm which solved the limitations presented in multi-layer perceptrons[1]; also in this year Rumelhart and McClelland published the two volumes of "Parallel Distributed Processing: Explorations in the Microstructures of Cognition" (Rumelhart and McClelland, 1986). Since then, the field of NN has been increasingly studied by a large number of researchers and practitioners in the various fields of computer science, neuroscience, cognitive science, psychology, mathematics and engineering. Several applications have been developed and reported in areas such as medical diagnosis, financial data analysis, decision making, engineering and industry (Bonissone et al., 1999; Freeman and Skapura, 1991).

2.4.1 Biological Conceptualisation of Neural Networks

Artificial neural networks were inspired by their biological neural networks counterpart. The fundamental elements of biological neural networks are the brain nerve cells, e.g. neurons. Neurons are separated into groups, called *networks*. Each neural network contains thousands of highly interconnected neurons (Freeman and Skapura, 1991).

The typical structure of a neuron includes dendrites, the cell body, and the axon. The dendrites are the transmission channels for the incoming signals (information). Synapses are the contact regions with other cells and the axon is responsible for transmitting the output signal to other neurons. The cell body contains organs responsible for its maintenance, such as the mitochondria, responsible for supplying the cell with energy.

[1] The back-propagation algorithm was first described by Werbos in his Ph.D. thesis in 1974 (1974), which remained unknown by the academic community for several years.

The axon is surrounded by a membrane named the myelin sheath, which is interrupted by nodes of Ranvier along the axon. Connections among neurons occur at various locations on the cell and are known as synapses; synapses connect the axon of a neuron to parts of other neurons. The connection among neurons is done by electrical signal and involves a complex chemical reaction. Briefly, the membrane keeps the neurons in an equilibrium state, called the resting potential, by controlling the chemical exchange of sodium and potassium ions between neurons and their external fluid through a pump mechanism. This mechanism transports sodium ions out of the cell and potassium ions into the cell (Freeman and Skapura, 1991).

Nerve impulses through synapses result in changes in the potentials of cell the body of the receiving neuron. For instance, excitatory inputs reduce the potential difference across the cell membrane, resulting in a large influx of positive sodium ions into the cell, decreasing the polarization of the cell. On the other hand, inhibitory inputs increase the polarization of the cell.

The communication between neurons occurs at the synapse level, as a result of the release of neurotransmitters and its absorption by the postsynaptic cell. Neurotransmitters are a substance released by presynaptic cell that causes a chemical reaction at the receptor neuron, and an influx of positive ions into the cell will cause an excitatory effect. Otherwise, negative ions will cause an inhibitory effect. Both effects are summed at the axon level; if the result of this sum is greater than a certain threshold then an action potential is generated (Freeman and Skapura, 1991).

Figure 2.3 illustrates the structure of a typical biological neuron.

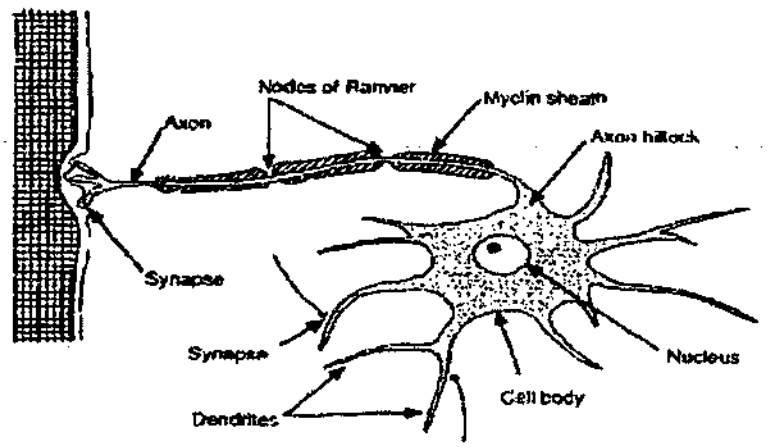

Figure 2.3: Structure of a typical neuron as in (Freeman and Skapura, 1991)

Although much research has been done to understand the functionality of biological neurons and biological neural networks, these subjects remain vastly unknown. Many areas of study have investigated the field of biological neural networks, such as medicine, chemistry and psychology. Computer sciences in general, and artificial intelligence in particular, have been an important test bed for many theories arising from these researches. In fact, artificial neural networks are a simplification of biological neural networks.

Deep study and discussion on biological neural networks and their relation with artificial neural networks can be found in (Anderson, 1995; Rojas, 1996; Wang, 1994; Haykin, 1994) and (Freeman and Skapura, 1991).

2.4.2 Neural Networks Conceptualisation

Using an analogy with biological neural networks, artificial neural networks consist of a collection of parallel processors, so called *neurons*[1], connected together in the form of a directed graph. The neurons are the nodes in the network structure and its fundamental element.

Artificial neurons receive inputs, analogous to the electrochemical impulses that the dendrites of biological neurons receive from other neurons. Then, the neuron processes the inputs and delivers a single output. The inputs can be data from a dataset, for example, or outputs from another neurons; and the outputs can be the final product of the connectionist system or can be an input to another neuron in the network.

The connection elements between neurons are called synapses, and each synapse has a *weight value* associated with it, also called *connection strength*. Connection strengths can be inhibitory, excitatory, or null. Inhibitory and excitatory connections are usually represented by a negative real number or positive real number, respectively. A zero value indicates no connection strength. The artificial neural process operates on two levels: the first level corresponds to an integrator of synaptic inputs weighed by connection strengths; the second is a function that operates on the output of the integrator, which calculates the neuron output value. Figure 2.4 illustrates the processing in an artificial neuron.

[1] The terms *nodes*, *units*, and *processing elements* are used as synonymous for neurons in diverse literature about neural networks.

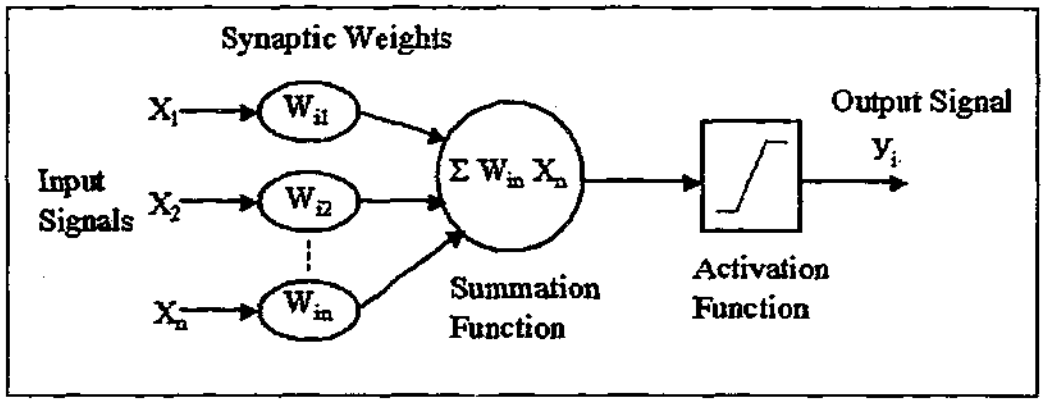

Figure 2.4: Processing of a generic artificial neuron

The integrator is a summation function that multiplies each input value X_i by its corresponding connection weight W_i and sums them, e.g., it calculates the inner product net_i (equation 2.1).

$$net_i = \sum_j x_j w_{ij}$$

Equation 2.1: Summation function

The result of the summation is a parameter of the activation function that calculates the neuron output value. Thus, the output value is calculated applying an activation function that has net_i as a parameter. Equation 2.2 shows the output function notation.

$$y_i = f_i(net_i)$$

Equation 2.2: Output function notation

There are several activation functions (f_i), and the most common ones are: the *Linear*, the *Step*, the *Ramp*, the *Sigmoid* and the *Gaussian* function (Simpson, 1992). The selection of a specific activation function determines the network operation, for example, determining a linear or a nonlinear relationship between the internal activation and the output.

One of the most common activation functions is the Sigmoid function, shown in equation 2.3. The Sigmoid function has a S-shape, it is a bounded, monotonic, no decreasing function that provides a nonlinear output value within a specific range. An example of the Sigmoid function is the Logistic function, which provides output values between 0 and 1, inclusive.

$$f(x) = \frac{1}{1 + e^{(-ax)}}$$

Equation 2.3: Sigmoid function

Where *a* denotes the slope parameter of the Sigmoid function, and is higher than 0 (usually equal 1).

2.4.3 Neural Network Structures

Neural networks are organized in layered structures, also called topologies, which are the building blocks of neural networks. The topologies are arrangements of neurons, including their connections and activation functions. The topologies, together with learning algorithms, define the overall neural network functionality. There are four basic neural network topologies: single-layer feedforward networks, multi-layer feedforward networks, recurrent networks, and lattice networks (Haykin, 1994).

Single-layer feedforward network. This is the simplest network model, in which a layer of inputs is connected to a layer of output neurons, where the network processing takes place. The single layer designation refers to the output layer, because there is no computation activity in the input layer. An example of this type of network topology is the linear associative memory neural network in which an input pattern is associated to an output pattern in the form of a vector.

Multi-layer feedforward network. In this architecture one or more intermediate layers (hidden layers) are applied. Multi-layer neural networks are in general used for learning nonlinear relationships between an input space and an output space; the analysis of the hidden units has shown that they are able to extract higher-order statistics, because of the extra set of synaptic connections and the dimension of neural iterations (Haykin, 1994), and are also capable of approximating any function (Hornik, Stinchcombe and White, 1989). Additionally, the mapping mechanism is usually very robust in representing generalizations of the presented examples. In a multi-layered neural network the input layer supplies elements of the input vector, which are the input signals to the neurons in the second layer. The output signals of the second layer are the inputs to the third layer, and the process follows this order until the

signals of the last layer are computed. The input-output relationship of each unit, not belonging to the input layer, is illustrated in equation 2.4:

$$y = f[\sum_{i=1}^{n} w_i x_i - \theta]$$

Equation 2.4: Multi-layer network processing function

Where y represents the output unit, n the number of inputs elements x_i, w_i the connection weights and θ the threshold and f the activation function for each unit's output function. Multi-layer networks can be fully connected or not; it is described as fully connected when each neuron in one layer is connected to all neurons in the adjacent forward layer, otherwise it is regarded as partially connected.

Figure 2.5 illustrates a partially connected multi-layer feedforward network, with a 4-2-3 topology, e.g., four input neurons, two hidden neurons and three output neurons. This neural network topology is particularly significant because the computational model developed in this thesis applies a multi-layer feedforward partially connected neural network model, with three layers.

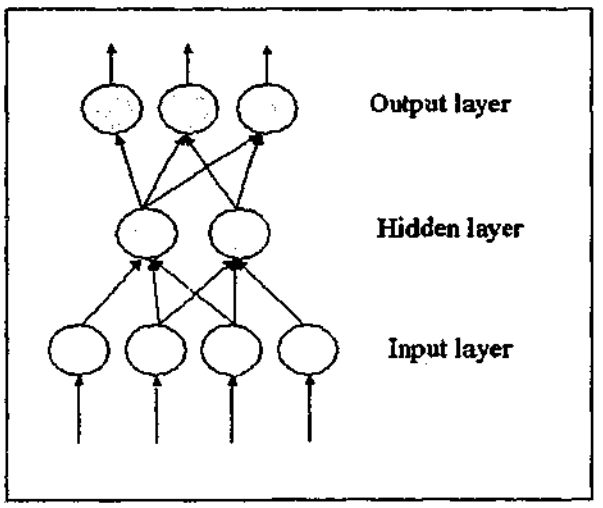

Figure 2.5: Multi-layer feedforward partially connected topology

Several design issues have to be evaluated when working with multi-layer neural networks. Some of the most difficult decisions are determining how many layers are enough for a given problem and how much data is enough to produce a sufficient mapping from the input layer

to the output layer. Several empirical works have demonstrated that three layers are enough to perform any nonlinear mapping with good degrees of accuracy (Simpson, 1992).

Multi-layer neural networks have been applied for pattern classification, pattern matching, and function approximation in solving many problems, such as prediction and forecasting (Wang, 1994).

Recurrent networks are similar to feedforward neural networks, but they have at least one feedback loop. Recurrent networks may have single or multi-layers, where each neuron feeds its output signals back to the inputs of the other neurons that might be in the same layer or in a previous layer. A neuron can also feed its output signal back to its own input, doing a self-feedback loop. Feedback loops greatly influence the learning capability of the network, and also its performance. Additionally, feedback loops employ unit-delay elements that bring a nonlinear dynamic behaviour to the network.

Lattice Structures consist of a multi dimensional array of neurons linked to a set of source nodes that supply the input signals to the array. The dimension of the lattice can be one-dimensional, two-dimensional, or higher-dimensions, in reference to the number of the dimensions of the space where the graph lies. Also, a lattice network model can be viewed as a feedforward model with the output neurons organized in rows and columns.

2.4.4 Neural Network Learning Approaches

The overall behaviour of an NN is given by its topology and the strengths of its synapses, and is associated with the implemented learning algorithm. One of the most interesting properties of NN is its ability to learn from the environment and, as a result, to improve its performance. Usually, everything a NN model needs to learn is a collection of representative examples of the environment about the problem being handled. After being presented with a collection of examples, the NN adapts itself to reproduce the desired outputs when presented with the example inputs. This ability gives a great advantage to NN when solving a problem, as it is not necessary to have a well defined process for algorithmically converting an input to an output (Freeman and Skapura, 1991).

Learning in NN is a result of the modification of the connection strengths of the synaptic junction between neurons, with respect to a given objective function. It is normally

accomplished by an interactive process of adjustments applied to the NN synaptic weights and thresholds, of finding weights that encode the knowledge that the system is expected to learn (Freeman and Skapura, 1991).

There are many NN learning algorithms, and they can be classified into two basic categories: *supervised* and *unsupervised* learning. It is not the objective of this thesis to provide a detailed discussion about NN learning algorithms, but it is necessary to briefly introduce some concepts about them. Firstly, the learning categories will be introduced, followed by some of the most important learning algorithms.

Supervised learning implies that the learning process is executed under the supervision of an external "teacher." The teacher is a component in the neural network structure that has knowledge about a particular domain (environment), which is represented by a set of input-output examples and which provides the neural network with a desired response. The neural network parameters are adjusted according to the training vector and an error signal, which is defined as the difference between the actual NN response and the desired response. This adjustment process is interactively performed until the NN reaches a state in which it emulates the teacher. Therefore, the teacher regulates the learning process, to inform the network of what has to be learned, and to assess the quality of the learning process. Some examples of supervised learning algorithms are error correction learning, reinforcement learning and stochastic learning.

Supervised learning is sub-classified in structural learning and temporal learning. Structural learning concerns finding the best relationship between the input and output data for each identified pattern. This learning approach has been applied in pattern matching and classification problems. Temporal learning is concerned with capturing a sequence of patterns needed to reach a particular desired outcome, as such, temporal learning assumes the existence of a dependence relationship between the current response of the network and the previous inputs and responses. Temporal learning has been applied in problems involving prediction and control.

Unsupervised learning, also identified as self-organized learning, is a process that relies only upon local information to discover its collective properties. There are no specific

examples of the function to be learned to guide the learning process. In this learning process the network iterates until it becomes tuned to certain statistical regularities of the input data, forming representations for encoding the input features and therefore automatically discovering the classes described by those features. Examples of unsupervised learning are Hebbian learning and competitive learning.

Following are brief descriptions of some of the most common NN learning algorithms, and various neural network models that implement these algorithms are identified.

The **error correction learning algorithm** aims to minimize a cost function based on the error signal, which is calculated by the difference between the desired response and the computed response of the neural network. As such, the algorithm adjusts the connection weights between neurons proportionally to the error signal. The objective is for the computed value of each output neuron in the network to match as much as possible the desired value for that neuron, considering various statistical properties. Two layer NN that apply error correction learning are able to capture linear mappings between input and output patterns, and multi-layer NN applying error correction learning are able to capture nonlinear mappings between input and output patterns (Simpson, 1992).

Perceptrons (Rosenblatt, 1962) and ADALINE (Widrow and Hoff, 1960) are two layer NN models that implement the error correction learning algorithm. Backpropagation (Werbos, 1974; Rumelhart, Hinton and Williams, 1986) is a multi-layer NN model that implements the error correction learning algorithm.

The **reinforcement learning algorithm** consists of an input-output mapping through a process of trial and error designed to maximize a performance index called the *reinforcement signal*, where the connection weights are reinforced in case of a correct response, and punished otherwise. Reinforcement learning is similar to error correction learning, but while error correction learning applies an algorithm where the outputs of each neuron on the output layer are considered, reinforcement learning applies a non-specific error information to determine the network performance, e.g., the reinforcement signal. This learning approach is based on Thorndike's law of effect, which says that if an action taken by a learning system is followed by a satisfactory state, then the tendency of the system to produce that particular action is

reinforced. Otherwise it is weakened (Haykin, 1994). The Adaptive Heuristic Critic (Barto, Sutton and Anderson, 1983) and the Associative Reward-Penalty (Barto, 1985) are examples of neural network models that implement reinforcement learning.

The **stochastic learning algorithm** applies a random process using a probability distribution and an energy function to adjust the connection weights in multilayered neural networks. The energy function value is determined by the states occupied by the individual neurons of the network. The stochastic learning procedure is performed by randomly changing the output signal of a neuron in a hidden layer (assuming for example a three layer network), followed by an evaluation of the resulting difference in the neural network energy (calculated through the energy function). If the calculated energy after the change is lower, the algorithm keeps changing the output signals, otherwise it accepts the change according to a previously determined probability distribution. The process repeats this procedure for each pattern pair in the dataset until the network reaches a stable state, in which its performance is considered satisfactory. The Boltzmann machine (Ackley, Hinton and Sejnowski, 1985) NN model implements the stochastic learning algorithm.

The **Hebbian learning algorithm** is a mechanism of adjustment of the connection weights based on the correlation of neuron activation values. The algorithm has been attributed to the neuropsychologist Donald Hebb (1949) who established the principle that a change in a synapse's efficacy is prompted by a neuron's ability to produce an output signal (Simpson, 1992). According to this principle, if two neurons in a connection (synapse) are simultaneously (synchronously) and persistently activated, then the strength of that synapse is selectively increased. Otherwise, if the neuron's activation is asynchronous, then that synapse is selectively weakened or pruned.

A Hebbian synapse employs a time dependent mechanism as it relies on the exact time of the activation occurrence; it is also employs a local mechanism to produce local synaptic modifications. Moreover, Hebbian synapses apply an interactive mechanism to increase synaptic efficiency as it depends on the correlation between the presynaptic and postsynaptic activities.

Some NN models that implement the Hebbian learning algorithm include the Linear Associative Memory (Kohonen, 1972; Haykin, 1994), the Hopfield network (Hopfield, 1982) and the Bidirectional Associative Memory (BAM) (Kosko, 1988).

The **competitive learning algorithm** is a method of automatically creating classes for a set of input patterns (Simpson, 1992). In this approach, output neurons of a neural network compete among themselves to be the one to be fired; as a result, only a single output neuron is active at a time. This is in contrast to Hebbian learning, where several output neurons may be simultaneously fired.

The Competitive learning algorithm has three basic elements: a set of neurons that differentiate from each other by having some randomly distributed connection weight, and consequently respond differently to a given set of inputs; a limit threshold on the strength of each neuron, and a mechanism that allows the neurons to compete for the chance to respond to a given set of input signs, in a way that only one output neuron will be active at a time. In the simplest form of competitive learning the NN has a single output layer, where each neuron is fully connected to the input neurons. The network may also include lateral connections among neurons, performing inhibitory activation among neurons laterally connected. The forward connections are excitatory. For a particular output neuron to be considered the winner in an interaction related with a specific input pattern, its net activity must be the largest among all the neurons in the network. Thus its output signal is set to one and the other neuron's output signals are set to zero.

Examples of neural network model that implement competitive learning algorithm are the Adaptive Resonance Theory (ART) (Grossberg, 1982) neural network and the Self-Organizing Feature Maps (SOM), also known as the Kohonen neural network model (Kohonen, 1982).

2.4.5 Neural Network Models.

Many neural network models have been developed since the publication of the Perceptron (Rosenblatt, 1962). Each neural network model has specific purposes in terms of applicability and/or functionality; for example, applications concerning pattern recognition, classification, and optimisation. Each model also implements a specific learning algorithm, and has a specific topology.

In the work developed in this thesis, it is not necessary to enumerate the various neural network models, but to mention some based on the criteria of applicability, relevance to the work developed in this research, topology and type of learning. For that reason two neural networks models are introduced. The first is Backpropagation NN model, as this is one of the most used NN models and has been largely applied in classification problems, the same kind of problem addressed in the research described in this thesis. Next, the Combinatorial Neural Model (CNM) is introduced, as this is the NN model employed in this research.

The chosen NN models are briefly described, as it is not the objective of this thesis to give a full description about their mathematical foundations and algorithms, but to introduce their main characteristics and potentialities.

2.4.5.1 The Backpropagation Neural Network Model

The Backpropagation neural model (BPN) (Werbos, 1974; Rumelhart, Hinton and Williams, 1986) implements a multilayer, fully connected feedforward network, through a supervised learning approach based on the error correction algorithm. There are no feedback loops and no connections that bypass one layer to a higher level layer, and usually Backpropagation models are implemented with three layers. Its topology is very similar at the one illustrated in Figure 2.5.

The BPN learning procedure begins with a predefined set of input-output examples (patterns), using a two-phase propagate and adapt cycle, where each input pattern has an output pattern specifying its classification. An input pattern is presented in a form of input vector to the first layer in the network. It is then propagated through each upper (hidden) layer to the output layer, which computes the network output values, e.g., output pattern. The computed output pattern is then compared to a given desired output pattern and an error signal is computed for each output neuron, as a result of this comparison. This process repeats layer by layer until each neuron in the network has computed an error signal that describes its relative contribution to the total error.

Based on the computed error signal, connection weights are updated in order to reduce the difference between the computed output pattern and the desired output pattern, converging the network towards a state that allows all training patterns to be encoded.

The learning algorithm implemented by the BPN is called the Generalized Delta Rule (GDR). The GDR minimizes the square of the difference between the computed and the desired output values summed over the output neurons and all pairs of input/output vectors. The GDR algorithm applies the sequence of equations previously defined in section 2.4.2, usually employing a Linear or Sigmoid activation function. Then, the algorithm applies a local gradient function to calculate the errors in the output and hidden layers, and moves backwards recalculating the weights of the neural network based on those local gradients.

The mathematical description of the GDR algorithm and its derivation can be found in (Freeman and Skapura, 1991), and also in (Beckenkamp, 2002).

The multi-layered Backpropagation NN model is applicable in solving complex pattern-matching problems (Freeman and Skapura, 1991), and is also recognized as very suitable for classification purposes (Piramuthu, Shaw and Gentry, 1990). Classification problems occur in financial management, credit analysis, medical diagnoses, prediction, and forecasting, among others.

2.4.5.2 The Combinatorial Neural Model

The Combinatorial Neural Model (CNM) (Machado and Rocha, 1989; Machado, Barbosa and Neves, 1998) was developed and explored during the last decade; it is a relatively new neural network model. It was inspired by the knowledge acquisition methodology of knowledge graphs (Leao and Rocha, 1990), which was developed to provide means for the representation and combination of knowledge elicited from multiple experts.

The CNM is an acyclic multilayer feedforward network. It is usually implemented with three layers: an input layer, a hidden layer, and an output layer. The output layer contains neurons that represent different classes; the input layer contains neurons that represent the domain information that supports the output classes, and the hidden layers specify different combinations of input neurons than can lead to a particular class.

CNM implements a supervised learning approach based on the error correction algorithm, similar to the Backpropagation, in which punishment and reward accumulators are computed for each connection in the network and the current connection weights are computed through the normalisation of those accumulators.

The CNM in its original form employs two learning algorithms, the Starter Reward and Punishment (SRP) and the Incremental Reward and Punishment (IRP), which enables the model to build a network based on knowledge elicited from domain experts and the refinement of such knowledge with additional examples. The hidden layers apply a fuzzy AND operator to propagate incoming input signals to the output layer. The output layer employs a fuzzy OR operator to compute and propagate the maximum incoming signal, e.g., the neural network output.

The CNM has been employed in several experiments dealing with classification problems, such medical diagnoses (Leao and Reategui, 1993b), credit card scoring (Reategui and Campbell, 1995) and engineering problems (Viademonte, Leao and Hoppen, 1995) with a good level of efficiency and accuracy.

Besides the Backpropagation and CNM, several other neural networks have been developed during the last years. The Perceptron (Rosenblatt, 1962) was the first neural network model developed, and it consists of a feedforward (acyclic) network with two layers. The Perceptron implements a supervised reinforcement learning algorithm in which connection weights are reinforced when the system performs correctly and punished otherwise. The Perceptron could recognise linearly separable patterns and one of its applications was character recognition.

Another important neural network model was developed by Grossberg (1982) and is known as the ART (Adaptive Resonance Theory). The ART network was developed as a solution to the stability-plasticity dilemma (Carpenter and Grossberg, 1987), which concerns the stability and plasticity of a learning system regarding the significance of its inputs (Freeman and Skapura, 1991). The ART neural network model employs a cyclic (bi-directional connections) network topology with three layers, implementing a competitive non-supervised learning approach; where connections can inhibit neighbouring neurons in a competitive fashion. In the ART model the bottom-up connections attempt to classify input signals, while top-down connections attempt to learn how to build classes (to make cluster) from the input signal. ART neural networks have been applied in image processing and signal recognition (radar and sonar).

Kohonen (1982) proposed the Self-Organizing Feature Map (SOM) neural network model, a feedforward network consisting of two layers. SOM implements a competitive unsupervised learning, in which a spatial organization of neurons is used in the design of the network, which enables the structured representation of input patterns.

The SOM network is concerned with the theory of competition, in which interactions among competitive neurons could be used to construct a network that can classify clusters of input vectors. The SOM acts as a competitive network for classification purposes. The algorithm results in a topology-preserving map of the input data to the output units. As a simplified definition, in a topology-preserving map neurons located physically next to each other will respond to classes of input vectors that are likewise located next to each other. The neurons become selectively tuned to various input vectors during the competitive learning process. In the SOM, all the neurons in the neighbourhood that receive positive feedback from the winning neuron participate in the learning process. The locations of the neurons tend to become ordered with respect to each other, building a coordinate system for different input features over the lattice.

A self-organized feature map is therefore characterized by the formation of a topographic map of the input vectors, in which the coordinates of the neurons in the lattice correspond to features of the input patterns. SOM neural networks have been applied in problems as speech recognition, learning data distribution probabilities, and robot control.

2.5 Applying Intelligent Systems

This section presents some applications involving hybrid architectures for intelligent systems that are related to the architecture proposed in this thesis, either combining symbolic and connectionist approaches, or other intelligent technologies such as case-based reasoning and rule induction. It does not intend to be a fully comprehensive list, but is representative enough of the current state of research.

An application of hybrid systems for material inspection in manufacturing industry quality control is described by Han and Wee (1992), where expert systems techniques are used to incorporate various pattern recognition techniques and neural network as tools for data

analysis. The KIPSE system (Knowledge-based Interactive Problem-solving System) is a computer-based problem solving environment for data analysis, specifically for classification problems. The KIPSE environment represents the overall solution of a problem in a decision tree structure, where each node of the tree represents a partial solution to a particular problem. New technologies can be integrated in the system by defining a new node in the system tree structure, and through this mechanism KIPSE incorporates various classifiers models (Bayes classifier, Fisher's linear discriminant classifier, K-nearest neighbourhood, and neural network models). KIPSE is based on the divide and conquer approach, providing the user with the capability of subdividing the problem into small sub problems and selecting the best methods available for the solution of each. In this way, the entire problem solving process is viewed as a multi-step decision making process, where at each step some kind of exploratory action is executed. The KIPSE system is built on top of the KEE (Knowledge Engineering Environment, a commercial expert system shell), using frame based structure to represent domain knowledge as classes and member units, and unit's attributes. Attributes are specified in slots, which contain descriptive and procedural information, and the relations are expressed using slot values. KIPSE can be compared to the intercommunicating hybrid system classification proposed by Goonatilake (Goonatilake and Khebbal, 1995), as it decomposes a specific problem in several tasks, where each task is assigned to a different technique to be solved.

An intelligent hybrid system for wastewater treatment is introduced by Krovvidy (Krovvidy and Wee, 1992). The WATTS system (WAtewater Treatment System) combines inductive learning, case base reasoning and NN approaches to generate optimal treatment processes for wastewater. According to Krovvidy (Ktovvidy and Wee, 1992) the design of a wastewater treatment system involves identifying, selecting and sizing a set of treatment processes. These procedures involve a database search to identify candidates' treatment processes and a combinatorial search to select an optimal sequence of such processes. The WATTS system consists of two phases: *analysis* and *synthesis*. During the analysis phase WATTS generates (induces) production rules from a set of examples stored in a tractability database, using the ID3 (Quinlan, 1987) algorithm to generate decision trees augmented with a

rule generator. The production rules obtained are further combined with a set of external rules compiled by domain experts, thus generating a knowledge base. This phase basically consists of a knowledge acquisition module. Next, during the synthesis phase, a treatment process is developed through a heuristic search and/or neural network approaches. The heuristic search algorithm is used to synthesize the processes reducing influent concentrations; a Hopfield neural network model is further applied to optimise the synthesis phase problem (minimizing costs) when needed, or even to synthesize the process when the problem size is too big to be tracked by heuristic search. In this last situation, the neural network module is used to decompose the problem in different stages. Furthermore, the WATTS system also applies a case-based reasoner to retrieve old solutions from a case base, when solving a new problem. WATTS updates the case base with the solution after a new problem is solved. The WATTS system can be related to the intercommunicating hybrid approach, as each technology consists of self-contained modules that exchange information and perform their respective roles to generate the problem solution.

A project investigating the application of computational intelligence techniques for data mining is described by Jagielski (Jagielski and Jagieiska, 1998). The project applies genetic programming and neural networks in the development of a decision support system (DSS) for issuing smog alerts in the city of Melbourne, Australia. The DSS model uses genetic programming and a neural network model to generate decision rules out of a database with smog cases used by the Victorian Environment Protection Authority (EPA) for smog event prediction. The predictive capability of both approaches was tested and their predictive performance compared. Both approaches showed comparable predictive performance; however, the genetic programming was capable of symbolic representation of decision rules leading to a more self explanatory solution. The architecture developed in this project can be related to the stand-alone model, as the symbolic and neural network approach work in a completely independent way and the main reason for this hybrid model is the comparison of performance between both approaches.

A hybrid architecture integrating a neural network model, the Combinatorial Neural Model, with case-based reasoning for classification problems, is proposed by Reategui (1997).

In this hybrid architecture the neural network makes hypotheses and provides insights that are used in guiding the search for similar experiences in a library of previous cases. The case-based reasoning component is responsible for the selection of the most similar match for a given problem, and also to support a particular hypothesis made by the neural network component, or even to decide among concurrent hypotheses. Additionally, structures called diagnosis descriptors have been created in order to represent the knowledge stored in the neural network in an intelligible way.

Four components are employed in the hybrid architecture: a *domain knowledge unit* that represents domain knowledge through a hierarchical representation of all findings (attribute/value pairs) used to describe the cases. A *case library* stores case descriptions and their respective solutions. The *neural network model* is trained through the cases stored in the library, and used during the consultation process to make hypotheses of possible diagnostic solutions and also to guide the search for similar cases. The *diagnosis descriptors* keep a representation of the knowledge stored in the neural networks; they are applied for consultation purposes, and also for building explanations features. The hybrid case-based reasoning and neural network model has been applied in the health care area, supporting the diagnosis of congenital heart diseases. The neural network and the case-based reasoning components are independent. Their integration is done by data exchange performed through the knowledge units and the diagnosis descriptors, thus this hybrid system can be related to the intercommunicating hybrid approach.

Sun (1995a) introduces the CONSYDERR system (CONnectionist SYstem Dual-representation for Evidential Robust Reasoning), which was applied in natural language understa ıding, commonsense reasoning, and planning tasks. CONSYDERR implements a different fashion of integration between connectionist and symbolic components. It represents domain knowledge in a two level architecture: the top level is a network with a symbolic localist[1] representation, and the bottom level is a network with a connectionist distributed[2] representation, where concepts and rules are represented in the bottom level by sets of feature

[1] *Localist* is a representation schema in which one node represents each concept in a domain (Sun, 1995a).
[2] *Distributed* is a representation schema in which a set of non-exclusive overlapping nodes represents each concept (Sun, 1995a).

units overlapping each other (Sun, 2001). The localist level comprises nodes, and each node represents a concept in the domain. In the distributed level, called the microfeature level, each node is a fine grained representation of a concept in the top level. The localist network is linked with the distributed connectionist network by connecting each node in the top level (representing one concept) to all the feature nodes in the bottom level representing the same concept. When a node in the localist level is activated, all the nodes in the microfeature level connected to that concept are also acuvated; through this interaction schema between the two levels CONSYDERR implements a rule-based reasoning approach. Using this mechanism CONSYDERR represents knowledge through the interaction between nodes in the form of rules, which are expressed by the nodes of the symbolic layer, connecting rule antecedents to rule consequents.

CONSYDERR can be associated to the fully-integrated and intercommunicating hybrid model. The connectionist and symbolic components are separated modules, communicating through a node to node integration schema.

A different approach from CONSYDERR is the RECON system. RECON is a framework in which several technologies are combined to incorporate a wide range of functionalities in a single system. RECON is a database mining framework, consisting of a hybrid system that includes artificial intelligence and conventional data analysis techniques that can be used, independently or cooperatively, to extract information from databases (Kerber, Livezey and Simoudis, 1995). Its architecture consists of five main elements: a graphical user interface, six database mining co ponents implementing a different technique each, a server component, a knowledge repository, and interfaces for database management systems. Two of the database mining components are a deductive database processor and a case-based reasoner, which are used for pattern validation. The other four components are used for data exploration, and these are: a supervised inductive learning component, a clustering component, data visualization, and a statistical analysis component. Additional components, for example neural networks, can be incorporated in the system as necessary. The RECON's supervised inductive learning component is used to automatically explore the target database to extract knowledge. It employs a symbolic induction algorithm that represents a

classification model as a collection of rules. The clustering component is applied to partition the target database into subsets, clusters, with the members of each cluster sharing interesting properties. The obtained clusters are used either as input to other data exploration techniques or for summarising the contents of the database accordingly their respective characteristics. The visualization component provides visual summaries of data from the database and also data generated from other components, such as clusters. The visualization component employs interactive visualization techniques (Kerber, Livezey and Simoudis, 1995). The last component in the RECON framework is a statistical analysis component, used to summarize data and perform statistical analysis. RECON has been applied in a variety of problems, such as financial analysis, manufacturing process analysis, and analysis of demographic information (Kerber, Livezey and Simoudis, 1995). The RECON system can be related to the intercommunicating hybrid system classification, as its components are independently and loosely integrated into the framework, and the interaction among the various components are mainly done through the knowledge repository where the knowledge extracted by one component is available to other components.

Table 2.2 summarizes the hybrid systems introduced in this section, according to their classification, applied technologies and applications:

Table 2.2: Summary of some hybrid intelligent systems

System	Classification	Applied technologies	Applications
KIPSE (Han and Wee, 1992)	intercommunicating hybrid	expert system and neural networks	industry quality control
WATTS (Krovvidy and Wee, 1992)	intercommunicating hybrid	inductive learning, case base reasoning and neural networks	wastewater treatment
(Jagielski and Jagielska, 1998)	stand-alone	genetic programming and neural network	smog alerts and prediction
(Reategui, 1997)	intercommunicating hybrid and loosely-coupled	neural network, case base reasoning and diagnosis descriptors	medical diagnosis
CONSYDERR (Sun, 1995a)	intercommunicating hybrid and fully integrated	symbolic (localist network) component and neural network	natural language understanding, commonsense reasoning, and planning
RECON (Kerber, Livezey and Simoudis, 1995)	intercommunicating hybrid	artificial intelligent (case base reasoning, supervised inductive learning, clustering algorithm) and statistical data analysis techniques	financial data analysis, manufacturing process analysis, and analysis of demographic information

The next section discusses some difficulties in developing intelligent systems that relate to this research.

2.6 Difficulties in Developing Intelligent Systems

The combination of different techniques to overcome the limitations of each has been the driven force behind intelligent system development. However, some difficulties remain. Some of the difficulties are related to **knowledge acquisition** and **representation**.

Early attempts in building expert systems revealed the difficulties of capturing, representing and incorporating expert knowledge (Buchanan and Feigenbaum, 1978; Tecuci and Kodratoff, 1995). The knowledge acquisition problem has been typically addressed through the application of artificial neural networks and/or inductive learning algorithms to automatically induce expert domain knowledge directly from raw data (Fayyad, Mannila and Ramakrishman, 1997). This approach has been explored in Case Base Reasoning (CBR) (Kolonder, 1993) and machine learning (Catlett, 1991; Quinlan, 1979; Tecuci and Kodratoff, 1995) experiments.

Extracting cases from raw data is an attractive solution for the problem of knowledge acquisition; as cases can be generated from databases, stored in case bases and further incorporated into an intelligent system. Furthermore, a machine learning algorithm, such as an inductive algorithm, can be applied upon the case bases inducing specific domain knowledge. Thus, in this situation, cases constitute the set of examples from where domain knowledge can be learned and represented in the form of rules or other knowledge representation formalisms.

This approach eliminates the necessity of long and costly knowledge engineering processes and the need for human experts. However, it brings additional problems, such as:

- the necessity of a significant amount of raw data from where domain knowledge can be generated. Some inductive learning algorithms and neural network models, such as the Backpropagation, require a large training dataset, e.g., raw data from where knowledge can be automatically extracted. Moreover, some inductive learning algorithms, mainly the decision tree generators, require that all the positive

examples (possible classes in a specific domain) to be simultaneously presented in the training set.

Potentially, large organisational transaction databases can be used as a source of raw data for automatic knowledge acquisition. However, using organisational transactions databases as data sources has some drawbacks (Fayyad et al., 1996; Smyth and Goodman, 1992), such as:

- the use of modern information technologies allows almost unlimited generation and storage of digital data, consequently the amount and diversity of available data is significantly large, demanding specific attention and procedures. This problem is known as "data overload phenomenon"
- dealing with large databases and high dimensionalities of data
- most of the data that comes from organisational transactions databases represents online business processing. Transactional databases are modelled and designed according to the process they support; consequently they normally are not suitable for knowledge-based activities such as decision making in ill-structured decision problems, which are situations where intelligent systems are normally applied
- transactional databases normally present problems regarding data quality such as missing and noisy data (Pyle, 1999), that may compromise the quality of the obtained knowledge.

Regardless of these problems, when properly represented and collected, transactional data can be explored with automated tools to help structure a particular problem domain. Knowledge Discovery in Databases (KDD) is concerned with exploiting, normally, massive datasets in supporting the use of historical data for decision making. Therefore, KDD technology represents a potential approach to handle problems of automatically building domain knowledge from transaction databases, such as dealing with noise, missing data, and with large amounts and high dimensionalities of data.

The subjects of knowledge discovery in databases and data mining are discussed in the next chapter.

2.7 Chapter Summary

This chapter has introduced the concepts and research related with intelligent decision support systems. The concepts and research in intelligent decision support systems were introduced in two topics: Intelligent Systems (IS) and Hybrid Symbolic-Connectionist Systems (HS). IS introduced hybrid systems combining different intelligent computer technologies, and HS introduced hybrid systems combining connectionist with symbolic approaches. These two types of hybrid architectures are closely related to this research. While the proposed framework for IDSS relates to IS in a general perspective, the employed NN environment relates to HS.

This chapter also introduced concepts of intelligent computing technology, focusing on artificial neural networks (NN). Concepts of neural networks were discussed, and neural network structures, learning approaches, and models were also introduced. Some well known NN models were briefly presented.

This chapter also presented some applications and experiments involving hybrid architectures for intelligent systems that are related to the architecture proposed in this thesis, either combining symbolic and connectionist approaches, or other intelligent technologies such as case-based reasoning and rule induction. Finally, some difficulties in developing intelligent systems were discussed.

Chapter 3

3 Knowledge Discovery in Databases

This chapter introduces concepts about Knowledge Discovery in Databases (KDD) and Data Mining (DM). The activities related to the KDD process and various data mining algorithms are introduced and discussed. Issues about data preprocessing are also discussed. The objective is not to extensively discuss these concepts, but to introduce the concepts relevant to the research described in this thesis.

3.1 Introduction

With major advances in engineering, computational and database technologies, organisations are now able to better generate, collect and store digital data. According to Frawley (Frawley, Piatetsky-Shapiro and Matheus, 1992) the amount of information in the world doubles every 20 months, and the size and number of databases probably increases even faster. This situation is known as the *data overload phenomenon* (Fayyad, Mannila and Ramakrishman, 1997). This phenomenon produces adverse effects on information use and decision quality, because information users are often exposed to more information than they can effectively process (Turetken and Sharda, 2004). As such, this creates the need for computational technologies to support data analysis, coordination, utilization, knowledge generation and management, in order to extract information useful for decision making (Gottgtroy, Rodrigues and Sousa, 1998).

Knowledge Discovery in Databases (KDD) addresses the problem of data overload. It is concerned with issues of extracting useful information in the form of patterns and models from normally large databases, and making those discovered patterns understandable and

suitable for decision problems resolution (Fayyad et al., 1996). In short, KDD concerns the theory and application of technologies for supporting analysis and decision making within, normally massive, datasets.

The term KDD can be formally defined according to Fayyad (Fayyad et al., 1996) as:

"the non-trivial process of discovering novel, implicit, potentially useful, and comprehensive knowledge from normally large amounts of data."

Knowledge is considered as any piece of information that may be identified as an interesting pattern according to a predefined user measure and criteria (Frawley, Piatetsky-Shapiro and Matheus, 1992).

KDD is a growing area of research and application that combines methods from disciplines such as statistics, artificial intelligence, machine learning, databases, data visualization, uncertainty modelling, high performance computing, knowledge representation, and a range of activities related to data analysis.

3.2 The Process of Knowledge Discovery in Databases

There are several definitions about which activities are included in the process of KDD; some literature describes data mining as the main or even the only activity in this process, and considers data mining and KDD as synonymous. Other literature considers KDD as the whole process of extraction of knowledge from data, and data mining the discovery stage of this large process.

From a general point of view, knowledge discovery in databases can be understood as a process that includes the stages of domain understanding, data selection and pre-processing, data mining, knowledge evaluation and the use of discovered knowledge.

Fayyad (Fayyad et al., 1996) defines KDD as a large, interactive and iterative process; and the various steps of this process include *data warehousing*, *data selection*, *cleaning*, *pre-processing*, *transformation* and *reduction*, *data mining*, *model selection*, *evaluation* and *interpretation*, and the *use of obtained knowledge*. Therefore, data mining is considered here as one of the various activities in the whole KDD process.

Additionally, Han (1998) includes the following stages in the KDD process: *data cleaning* and *integration, data warehousing, data selection, data mining* and *pattern evaluation.*

Normally, the KDD process is divided into two major stages. First, data is gathered from (organisational) databases and external sources, prepared and then loaded into the data warehouse. It includes the activities of selecting data from transactional databases, cleaning the data, adding data from external sources (enrichment), coding and finally loading the data into the data warehouse. This stage is frequently referred as ETL (Extract, Transform and Load) in the literature. Figure 3.1 illustrates the various tasks of getting data from transaction databases into the data warehouse.

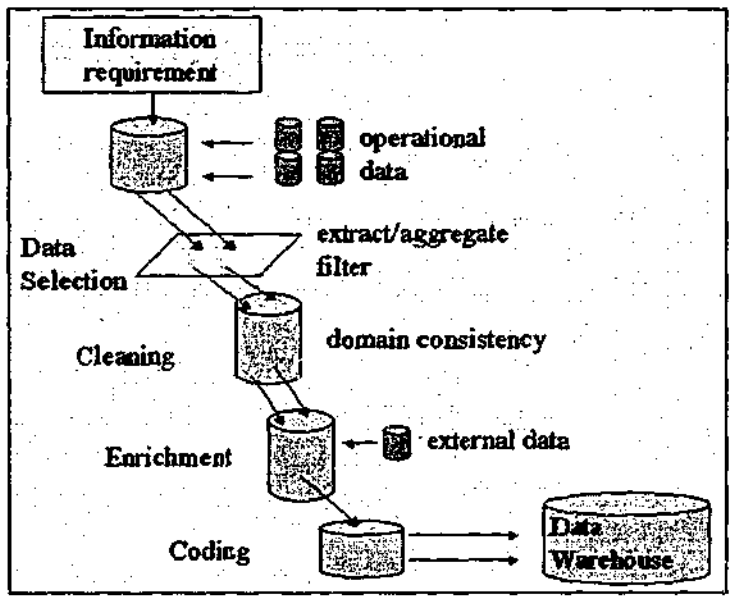

Figure 3.1: KDD stage from database to data warehousing

The next stage includes the activities of extracting data from the data warehouse, loading the data into a data mining environment, mining the data, analysing and evaluating extracted knowledge. The final stage is the application of the obtained knowledge in knowledge based activities, such as decision support. Figure 3.2 illustrates the stage from the data warehouse to data mining.

Different stages and even definitions about the process of KDD can be found in the literature, however there is general agreement with the steps proposed by Fayyad (Fayyad et al., 1996). It has to be noted that many KDD applications do not follow a particular order; neither are all the activities equally performed. For example, in many cases a data warehouse to

extract cleaned and pre-processed data does not exist. In other cases, historical transactional data may not be readily available, or too expensive to obtain.

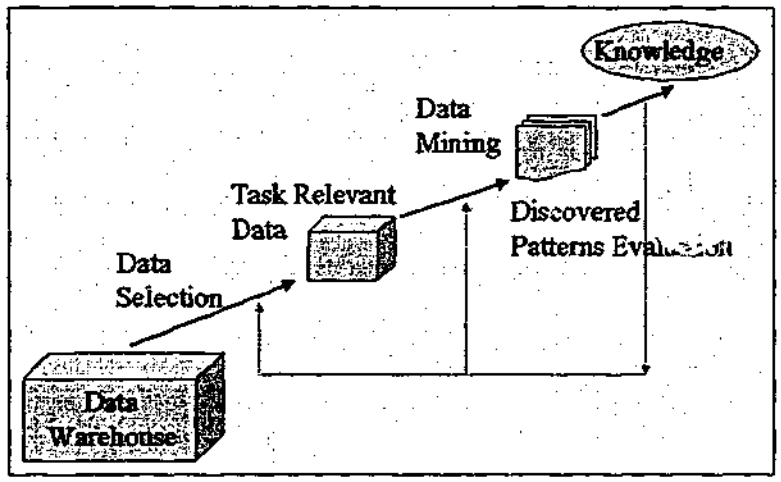

Figure 3.2: KDD stage from data warehouse to data mining, as in (Han, 1998)

The research project described in this thesis draws from the concept defined by Fayyad, previously mentioned. Moreover, for simplicity, the term *data preparation* is used to refer the activities from selecting data for data mining, pre-processing these data until to effectively mine the data.

Therefore, this thesis considers KDD as the *whole* process of extraction of knowledge from data, and data mining the *discovery* stage of that process.

3.2.1 Classification of KDD Problems

The type of problems addressed by KDD technology can be classified into two major groups according to Meneses (Meneses and Grinstein, 1998): *prediction* and *description.* Prediction concerns building a *predictive model* that will predict unknown values of a variable, from the known values of other variables. Description is concerned with finding patterns that describe the model represented by the data or even the process generating the data, e.g., it is concerned with building a *descriptive model.*

Weiss (Weiss and Indurkhya, 1998) proposed a categorization in which candidate problems are classified into two general categories: *prediction* and *knowledge discovery*, where knowledge discovery describes a stage prior to prediction, in which the available information is insufficient for prediction.

For example, the activity of *knowledge acquisition*, previously discussed in section 2.2.2, concerns building and populating knowledge bases. It involves the investigation of a particular domain, determining what concepts are important in that domain, creating a formal description of the objects and relations, and finally building a knowledge base. As such, it falls in the *description* classification according to Meneses (Meneses and Grinstein, 1998) or in the *knowledge discovery* classification, according to Weiss (Weiss and Indurkhya, 1998).

In this thesis, the categories of *prediction* and *description* are used to specify KDD problems. Therefore, knowledge discovery and knowledge engineering are considered *description* problems. And the term *knowledge discovery* is regarded as the highest level purpose of any KDD application.

3.3 Data Mining

Data mining is considered the core activity of the KDD process (Han, 1998), and can be understood as the process and the set of techniques and algorithms used to find (mine) the underlying structure, information and relationships in normally large amounts of data, e.g., data mining is the knowledge discovery *stage* within the whole knowledge discovery process.

Data mining, as KDD, is an inter-disciplinary field with a wide variety of techniques for extracting knowledge from databases; these techniques are drawn primarily from disciplines such as statistics, machine learning, and artificial neural networks. Each technique has its particularities, and it is suitable to a certain range of tasks in a certain context. For that reason, it is important to identify the tasks in a data mining application in order to select the appropriate techniques.

3.3.1 Data Mining Tasks

The literature enumerates distinct data mining tasks. It is not the purpose of this thesis to cover all the possible definitions of data mining tasks, but to introduce some definitions and tasks that are usually referred in most of the literature and relate to the work developed in this research.

This thesis draws on the definition given by Fayyad (Fayyad et al., 1996) in which data mining tasks are categorized as *classification*, *association*, *regression*, *clustering*, *deviation detection*,

database segmentation, summarization, visualization, text mining, dependency modelling, and *sequence analysis.*

Besides these, Hand (Hand, Mannila and Smyth, 2001) enumerates as data mining tasks the following: *density estimation, clustering* and *segmentation, classification, regression, pattern discovery,* and *retrieval by content.*

And Han (1998) proposes the following categories: *summarization, characterization, association, classification, clustering, prediction, trend and deviation analysis,* and *pattern analysis.*

Again, different enumerations can be found in the literature, but the categories of tasks listed above are the most common mentioned and accepted in the KDD literature.

For example, to illustrate the relation between KDD problems, data mining tasks and techniques, consider a typical database marketing application: cross-selling. Cross-selling is the activity in which a particular retail company wants to maximize sales of products from its customer base. One possible approach for the cross-selling problem is analysing the purchase patterns of products frequently purchased together.

According to the KDD classification schema mentioned in section 3.2.1, the cross-selling problem can be categorized either as a *description* or a *knowledge discovery* problem, where the goal is to *discover* and *describe* the purchase patterns of different products. Additionally, the cross-selling problem can be categorized as a data mining *clustering* task. Cluster analysis is one possible approach to this problem, and requires finding clusters of products frequently purchased together. Besides that, the cross-selling problem can also be handled by an *association* approach, finding products frequently associated with each other in a certain number of transactions. In this last case, the cross-selling problem is categorized as a data mining *association* task, instead of clustering.

It is important to understand that there is a wide variety of data mining problems, each of which with respective tasks. For each task a group of techniques and algorithms exist, and some algorithms are more suitable for a certain task than others. The selection of a particular algorithm, technique or group of techniques depends on the specific KDD problem and the specific data mining tasks, as well as the complexity of the problem under consideration.

Weiss (Weiss and Indurkhya, 1998) proposed an useful schema to classify the data mining tasks according to specific KDD problems. Table 3.1 shows that schema.

Table 3.1: Relation between KDD problems and data mining tasks

Data Mining Tasks	KDD Problems
Classification	Prediction
Regression	Prediction
Deviation detection	Description
Database segmentation	Description
Clustering	Description
Association rules	Description
Summarization	Description
Visualization	Description
Text mining	Description

According to this schema the KDD problems are classified into two major groups, prediction and description. The data mining tasks of classification and regression are considered as prediction problems, meanwhile the remaining data mining tasks such as clustering, visualization and database segmentation are considered as description problems in the KDD context.

3.3.1.1 Classification Tasks

Particularly relevant to the research project described in this thesis are classification tasks, as the computational model for decision support developed here was applied in a classification problem.

When building predictive models, prediction of categorical variables is identified as *classification*, while prediction of quantitative variables is identified as *regression* (some of the literature uses *estimation* when referring to prediction of quantitative variables, considering regression as a particular case of estimation (Kennedy et al., 1998)).

The goal in **classification tasks** is to build a classifier that, given a certain domain, systematically predicts what class a particular new case falls in. Classification problems concern the construction of a *classifier procedure* that is applied to a sequence of correctly classified cases, in which new cases are assigned to one of a set of pre-defined classes based on its observed

attributes or features. Classifiers are defined based on the measurement data on N cases observed in the past together with their actual classification.

A classifier can be defined as a partition of the measurement space x, which contains all possible measurements vectors, into J disjoint subsets $A1, A2,..., Aj$ and $x = UAj$ such that for every measurement feature vector $x \in Aj$ the predicted class is J.

3.3.2 Algorithms for Data Mining

Besides issues of data availability and data quality, the success of a KDD application also relies on the usefulness of the chosen algorithm for the class of problems being considered. This section briefly introduces some of the most common algorithms and their respective data mining tasks[1]. One algorithm for learning association rules is described in more detail, as it is the chosen data mining algorithm for the hybrid computational model for decision support proposed in this thesis.

Three major approaches are currently used for data mining. These are methods based on statistics, machine learning and artificial neural networks. From a statistical perspective *Fisher's linear discriminants* and *Bayesian inference* are some of the most common techniques for data mining (Michie, Spiegelhalter and Taylor, 1994), for instance, the algorithms *Autoclass* (Cheeseman and Stutz, 1996) and *Bayesian Networks* (Heckerman, 1996) have been largely applied in data mining for classification purposes.

From a neural network perspective several models have been used in data mining, among them *MLP* (multilayered perceptron) implementing the *Backpropagation* algorithm (Werbos, 1974; Rumelhart, Hinton and Williams, 1986). This has been applied as a nonlinear model for *prediction problems*, specifically in regression and classification tasks. *Self Organizing Feature Maps* (Kohonen, 1982) is usually applied for *clustering* tasks, and *Radial Basis Functions* (Freeman and Skapura, 1991) has also been applied for *prediction problems*, either in classification and regression tasks.

[1] This thesis assumes an "off-the-shelf" view of data mining algorithms. A more complex view is to see an algorithm as a component-based structure, in which algorithms are composed of a combination of well-defined components, such as model structure, score function, search methods, and data management technique. Therefore, a data miner should analyse which components fit the specifics of his/her problem, instead of which specific "off-the-shelf" algorithm to choose. Hand (Hand, Mannila and Smyth, 2001) discusses the component-based view of data mining algorithms.

The *k-nearest neighbourhood* algorithm (Dasarathy, 1990), decision tree generators such as *C4.5* (Quinlan, 1993) and *CART* (Breiman et al., 1984), and rule induction algorithms (with a machine learning background) are also techniques applied in several data mining tasks.

For example, CART (Classification And Regression Trees) (Breiman et al., 1984; Hand, Mannila and Smyth, 2001) is a widely used algorithm in data mining. CART produces models with a tree-based structure, for *regression* and *classification* tasks, as its name suggests. The CART tree model consists of a hierarchy of univariate binary decisions, which means that each node in the tree specifies a binary test on a single variable. An example represented by a data vector *X* follows a particular path in the tree structure from the root node to a leaf node based on how the values of each its components match the binary test of the internal nodes. Each leaf node specifies a particular class label. CART derives the tree structure from the data by choosing the best variable (attribute) for splitting the data into two groups, starting at the root node. CART uses several splitting criteria (which are significantly complex), and all of them produce the effect of partitioning the data at an internal node into two disjoint subsets. The splitting procedure is then recursively applied in each of the child nodes, until it reaches the final nodes, e.g., the leaves. The size of the tree is a result of a pruning process, which attempts to build optimal tree sizes, because large trees may overfit the data, and too small trees may have insufficient predictive capability. Full descriptions about CART algorithm can be found in (Breiman et al., 1984) and also in (Hand, Mannila and Smyth, 2001).

Another algorithm often applied in *classification* tasks is the *k-nearest neighbour* (Dasarathy, 1990). The k-nearest neighbour algorithm is a classic nonparametric method, in which the training procedure consists in storing all input-to-output pairs from the training set into a database. When a prediction (classification or regression) is needed on a new input case, the result is achieved based on the K nearest training patterns in the database. As such, the main decision in the algorithm is to set the parameter K and the type of distance metric. The parameter K indicates the number of nearest neighbours used to classify the input cases; the best value for K is normally determined empirically using validation techniques such as cross validation. The distance metric, which computes the meaning of nearest neighbour, is normally the Euclidean measure that measures the distance between two D-dimensional

vectors. The standard deviation is also used as distance measure. Basically, for each new case the algorithm searches for the K nearest patterns to the input case using, for example, the Euclidean distance measure. The algorithm computes the confidence for each class *i*, as *Ci/K*, where *Ci* is the number of classes among the K nearest patterns belonging to class *i*. As a result, the classification for the input case is the class with the highest confidence value, which means that the input case is assigned the same classification as its K nearest neighbours. For regression problems, the output value is calculated based on the average of the output values of the K nearest patterns. Detailed discussion about the k-nearest neighbour algorithm can be found in (Dasarathy, 1990) and in (Molina, Blanca and Taylor, 1994).

Besides neural networks, machine learning and statistical approaches, data visualization algorithms also have been applied for data mining purposes. Data visualization techniques are powerful approaches to visualize and discover patterns in datasets, and there is a wide range of visual techniques that can be applied in the various stages of the KDD process. They can be used to get a first perspective about data features and their distribution during the pre-processing stage, or to find data clusters, correlations and dependencies among attributes during a data mining stage, or to visualize the final model produced by the data mining algorithm.

Some of the most common visualization techniques are 2D (two dimensional) and 3D (three dimensional) scatter-plots and scatter-plot matrix (Cleveland and McGill, 1988), and various multi-dimensional visualization techniques that map a multi-dimension dataset in a 2D or 3D mapping, such as Circle Segments (Ankerst, Keim and Kriegel, 1996) and Chernoff Faces (Chernoff, 1973). There are also hierarchical techniques that subdivide N-dimensional space into subspaces, and present the subspaces in a hierarchical fashion, such as "Worlds within Worlds" (Beshers and Feiner, 1990), InfoCube (Rekimoto and Green, 1993) and Treemaps (Schneiderman, 1992), among other techniques.

For example, Turetken and Sharda (2004) present an interesting work addressing the overload problem through information visualization. This work introduces the FISPA system (Fisheye-based information search processing aid), which manages information overload resulting from web searches. In brief, FISPA organizes search results by grouping them into a

hierarchy based on their individual contents, and then presents a visual overview of the groups facilitating the users to zoom on specific groups of interest.

Moustakis (Moustakis, Letho and Salvendy, 1996) presented a useful survey comparing different learning algorithms and types of problems. In this survey, six classes of algorithms and three types of problems were selected. The selected algorithms were the k-nearest neighbour (KNN), decision trees (DTrees) (any algorithm that fall in this category such as CART and C4.5), association rules (ARules), neural networks (NN), genetic algorithms (GA), and inductive logical programming (ILP). The selected types of problems were: classification, problem solving and knowledge engineering.

According to this survey, k-nearest neighbour, decision trees and association rules algorithms, generally, perform better in classification problems than in problem solving and knowledge engineering. Neural networks also showed good performance in classification, and genetic algorithms showed to be appropriate for problem solving tasks. Inductive logical programming outperformed the other algorithms in knowledge engineering tasks. Figure 3.3 illustrates the results of this survey.

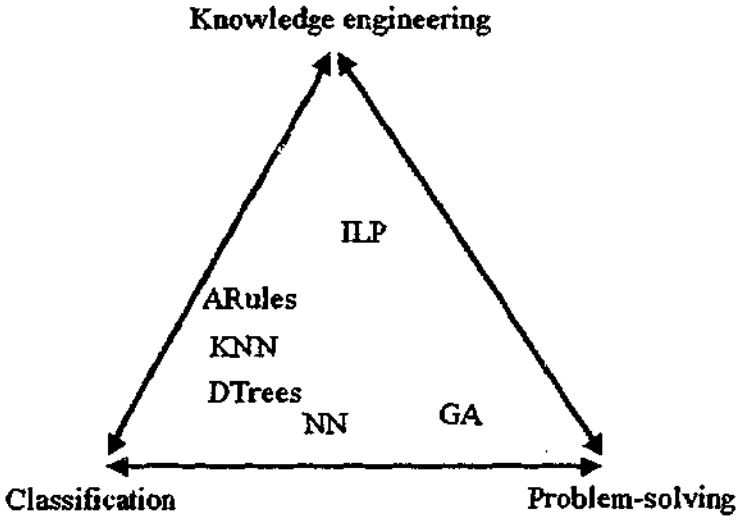

Figure 3.3: Relation between learning algorithms and type of problems, as in (Meneses and Grinstein, 1998)

There are several studies and surveys examining data mining algorithms, their performance and suitability relating to specific problems. Further discussions about data mining algorithms and their applicability can be found in (Kennedy et al., 1998) and also in (Michie, Spiegelhalter and Taylor, 1994).

3.3.2.1 The Apriori Algorithm for Association Rules Learning

Association rules are an approach to discover regularities in databases and represent them in the form of "If-Then" rules, finding sets of the most commonly occurring groupings of items. Originally association rules were applied to analyse large amounts of supermarket basket data (Agrawal et al., 1996), describing how often items are purchased together. Association rules became one of the most used approaches for data mining, being applied in a range of different situations (Klemettinen, Mannila and Toivone, 1999).

The process of finding association rules consists firstly in finding all the sets of items, named *itemsets*, that occur in the database with a frequency equal or higher than a pre-defined frequency threshold, called *minimum support*. Such itemsets are known as *large itemsets*. An itemset with k items is named *k-itemset*. The next phase consists of forming rules among the large itemsets previously found (Agrawal, Imielinsk and Swami, 1993).

The general structure of association rules generator algorithms consists of making multiple scans over the database. In the first scan the supports of individual itemsets are computed, finding the *large itemsets*, i.e. items with minimum support. Each subsequent scan starts from a seed set of large itemsets found in the previous scan. This seed set of itemsets is used to generate new potentially large itemsets, named *candidate* itemsets, and the support of these candidate itemsets are then computed through a scan over the database. At the end of the scan, the large candidate itemsets are selected, and they became the seed for the next scan. The process follows this fashion until no new large itemsets are found in the database.

A formal definition of the problem of mining association rules is given by Agrawal (Agrawal, Imielinsk and Swami, 1993) as follows:

Let $I = \{I_1, I_2, ..., I_m\}$ be a set of m distinct items;

$D = \{T_1, T_2, ..., T_n\}$ a set of T distinct transactions, where:

each transaction $T_i = \{I_{i1}, I_{i2}, ..., I_{ik}\}$, is a set of items I, where $I_{ij} \in I$ and $T \subseteq I$.

Associated with each transaction T is a unique identifier *TID*.

It is said that a transaction T contains X, a set of some items in I, if:

$X \subseteq T$.

An **association rule** is an implication of the form $\boldsymbol{X \Rightarrow Y}$,

where $X, Y \subset I$ and $X \cap Y = \varnothing$.

The rule $X \Rightarrow Y$ has **support** S if $S\%$ of the transactions in D contain $X \cup Y$.

The rule $X \Rightarrow Y$ has **confidence** C if $C\%$ of transactions in D that contain X also contains Y.

The confidence is the ratio of the number of transaction that contain $X \cup Y$ to the number of transactions that contain X, given by the following expression:

$C = support\ (X \cup Y)\ /\ support\ (X)$

As a result, given a set of transactions D, the problem of mining association rules is to generate all association rules that have support and confidence greater or equal than a pre-defined minimum support and minimum confidence.

One of the most well known association rules generator algorithms is the Apriori algorithm (Agrawal et al., 1996). The general structure of the Apriori algorithm follows the general structure of association rules generator algorithms, except that any subset of a large itemset must also be large. During each iteration through the dataset only large candidates found in the previous iteration are selected to generate new candidates. The structure of the Apriori algorithm is as follows (Agrawal et al., 1996):

Let k denote the number of items in an itemset, e.g. itemset size, such that an itemset of size k is termed a *k-itemset*.

Let L_k denote the set of large k-itemsets, and C_k the set of candidate k-itemsets.

L_1 = {large 1-itemsets};
for ($k = 2$; $L_{k-1} \neq \varnothing$; k++)
    C_k = set of new candidates from L_{k-1};
    for all transactions $T \in D$
        for all k-subsets I of T
            if ($I \in C_k$) then *s.count* ++;
    end
    L_k = { $c \in C_k$ | *c.count* ≥ minimum support};
end
Set of all large itemsets = $\mathrm{U}_k L_k$;

Figure 3.4: The Apriori algorithm for association rules (Agrawal et al., 1996)

There are several association rules generator algorithms, such as the AIS (Agrawal, Imielinsk and Swami, 1993), SETM (Houtsma and Swami, 1993), DHP (Park, Chen and Yu, 1995), AprioriTid (Agrawal et al., 1998) and the Apriori (Agrawal et al., 1996) algorithm.

This thesis applies the Apriori (Agrawal et al., 1996) algorithm for association rules.

Empirical evaluations demonstrated that the Apriori algorithm outperforms the AIS and SETM algorithms, performs equally to the AprioriTid, and has good scale-up properties (Agrawal and Srikant, 1994).

Further discussion about the Apriori algorithm and its performance evaluation can be found in (Agrawal and Srikant, 1994), and further discussions about association rules can be found in (Agrawal, Imielinsk and Swami, 1993; Agrawal et al., 1998; Mohammed et al., 1996) and (Adamo, 2000).

The data mining component of the DM-NN model for IDSS proposed in this research applies a stand alone implementation of the Apriori algorithm.

3.4 Data Preparation

The data preparation stage in many KDD projects is usually time and budget consuming, as frequently the input dataset present problems like noise and null values, which require intensive data transformation and cleaning procedures. The importance of the data preparation stage should not be underestimated, because if the quality of the data is low or the problem is incorrectly formalized, the algorithms used for data mining will behave inefficiently or even produce incorrect results.

Data preparation concerns the quality of database attributes[1] being used for data mining purposes. The use of real databases, i.e., databases not artificially built for experiments purposes but provided from real world applications, brings several challenging problems.

According to (Pyle, 1999) the purpose of data preparation is to manipulate and transform row data so a mining tool can best handle the information in the dataset. Performing data preparation means carefully analysing each data, its information content and how accurately

[1] The term "attribute" is used in this thesis to indicate a database field. However, the terms "variables", "fields" and "features" are also used for the same purpose, when it contributes to a better understanding of the subject being discussed or when referencing a work that applies those terms.

the attributes represent the expected information. Berry gives some suggestions about what characteristics a dataset for data mining should have, according to (Berry and Linoff, 2000):

- all data should be in a single table or database view
- each row should correspond to an instance that is relevant to the application domain
- columns with a single value should be ignored
- columns with different values for every row should be ignored, although the information they contain may be extracted into derived columns
- for predictive modelling, the target column should be identified and all synonymous columns removed.

There are many characteristics inherent in the nature of data; some of them may represent problems that need to be addressed when preparing data for data mining purposes. In general these characteristics concern *volume of data* and *data quality issues*, such as irrelevant fields, sparsity, overfitting, monotonicity, dimensionality, outliers, anachronisms and variables with empty or missing values (Fayyad et al., 1996). These characteristics, usually, imply removing or transforming attributes from the dataset. Each of these characteristics is now briefly discussed.

A high volume of data is one of the challenges facing the KDD community. It relates to the capability of inductive learning algorithms to *mine large or very large databases* in order to discover useful knowledge. Large databases represent a difficult problem for data mining applications for several reasons, ranging from the quality of their data to the time and computer resources needed to process them. Possible solutions that have been suggested to this problem are: breaking the dataset to reduce the amount of data (data partitioning), speeding up the algorithm, and parallel processing. Detailed discussion about the approaches of data partitioning, algorithm speeding up, and parallel processing can be found in (Provost and Kolluri, 1999) and (Gottgtroy, Rodrigues and Sousa, 1998).

Closely related to the problem of mining large databases is the problem regarding *dimensionality*. This concerns the need to reduce (or even increase) an attribute's dimensions. When a database has a very large number of attributes, it is said its dimensionality is high and consequently the problem being modelled through this database also has a high

dimensionality. A high dimensional dataset increases the size of the search space at an exponential rate for most of the induction algorithms, increasing the probability of finding false and erroneous patterns.

Methods to reduce the dimensionality have been suggested, such as reducing the number of attributes (known as vertical reducing methods or features selection), data partitioning methods (known as horizontal reduction or case reduction), and processing data subsets sequentially or concurrently (Provost and Kolluri, 1999). Normally, the problem of mining large datasets is in reducing the dimensionality, however there are some cases where the dimensionality of an attribute needs to be increased. Increasing dimensionality, in general, applies to non-numerical data when presenting information that is better understood in more than one dimension.

Dimensionality reduction, specifically case reduction and sampling, are further discussed in the next section, as it is particularly relevant to the research described in this thesis.

The problem of *overfitting* is normally related to the limited amount of information. When a learning algorithm searches for the best parameters for one particular model using a limited amount of data it may overfit the data, resulting in poor performance. This means that the learning algorithm memorized the training set instead of generalizing the concepts. In this situation the algorithm normally performs very well in the training set, but performs poorly when presented to new cases. Suggested solutions for the overfitting problem include cross-validation, or even artificially increasing the data population (Berry and Linoff, 2000).

One of the basic decisions in data preparation is whether or not to remove a certain attribute. In general, an attribute must be removed when it presents few values in its range of values, or it is a constant, or most of its values are empty or missing (Pyle, 1999), e.g. the attribute suffers from *lack of variability*. Attributes that contain only a single value or with entirely missing values can be removed from the dataset as the lack of variation in content implies no information for modelling purposes (Pyle, 1999).

Handling *missing* and *empty* values is one of the most common problems in data preparation. An empty value is a value that has no corresponding real world value, while a missing value means a value that has not been stored in the dataset but which exists in the

domain. Unfortunately, determining if any particular attribute is empty, rather than missing, requires domain knowledge, and this is extremely difficult to detect automatically. Missing and empty values can be removed or replaced in the dataset; basically this decision relies on the amount of these values in the dataset and in domain knowledge as well. Domain knowledge is necessary because it is possible that in a particular application empty or missing values can have some information content, and in such cases the values must be kept in the dataset. Refer to (Pyle, 1999) and (Catlett, 1991) for further discussions on how to handle missing and empty values.

Other common problem found in data preparation for data mining concerns situations when the information density of a certain attribute is low; this is called *sparsity*. This means that most of the instances values are empty, but some values are recorded. Normally, in this case, these attributes have insignificant values and can be removed. However there are cases when sparse values represent significant information, such as in fraud detection, where the exceptions are the most important information to be captured. Essentially, to remove a sparse attribute or not is a decision based on confidence levels (Pyle, 1999).

There are techniques to deal with sparsity problems, such as dimensionality reduction or collapsing a sparse attribute with another attribute. Detailed discussions about techniques to deal with sparsity can be found in (Pyle, 1999; Weiss and Indurkhya, 1998) and (Berry and Linoff, 2000).

Another problem often found in data preparation is the occurrence of monotonic attributes, or *monotonicity*. Basically, a monotonic attribute is an attribute that increases without bounds. The most common examples of monotonic attributes are date stamps; it means attributes that are linked with the passage of time, such as dates. Usually, monotonic attributes are transformed into non-monotonic form for use in data mining.

When an attribute presents a very low frequency occurrence of a value, or a set of values, that is far away from the bounds of other values in the column, this is known as an *outlier* attribute. Outliers usually indicate a mistake, but there are cases when an outlier attribute could represent valid information. Normally, domain knowledge is necessary to decide wether an outlier value, or set of values, represents an error or not. Basically, there are two ways of

handling an outlier attribute: to remove it or to remap the value reducing the distance gap between the outlying value and the bulk of values.

Another problem in data preparation is called *anachronism*, which can be described as *temporal displacement* according to Pyle (1999). An anachronistic attribute is one that contains no information, or the information is simply not available. Anachronistic attributes should be removed from the dataset.

There are several works in the literature that address the problems of preparing data for knowledge discovery and data mining, appointing solutions, methodologies and case studies. Detailed and further discussion on those issues can be found in (Pyle, 1999; Fayyad, Haussler and Stolorz, 1996; Gottgtroy, Rodrigues and Sousa, 1998) and (Berry and Linoff, 2000).

The problems discussed above arise, mainly, when dealing with significant amounts of data. Consequently, it brings a primary issue into consideration that refers to the sufficient amount of data for data mining. The next section discusses how much data is necessary for data mining.

3.5 Dimensionality Reduction

In data modelling for data mining it is essential to know how much data is enough to build descriptive or predictive models. Basically, the dataset must be large enough in order to capture sufficient information about each individual variable (attribute) and the interactions among them (Provost, Jensen and Oates, 2001; Pyle, 1999).

Dimensionality reduction attempts to build datasets that accomplish this, to reduce data dimensions while preserving data concepts (Weiss and Indurkhya, 1998). Dimensionality reduction deals with the selection of features, values and instances. This section specifically refers to instance selection (case selection), which addresses the issue of whether all cases (instances) residing in a database are needed for effective mining.

Generally speaking, a larger amount of data is better because large data potentially can provide more evidence about the induced concepts (Weiss and Indurkhya, 1998). If a small amount of cases is used there is a higher risk that some concepts will not be represented in the dataset, and consequently not captured by the learning algorithm.

However, there are some issues to consider. Firstly, it is necessary to know how much data is available. Second, it is important to be sure that there are enough instances representing the events of interest. While a huge amount of data has the potential for better results, there is no guarantee that it will generate better results than small data. Furthermore, increasing the number of cases also increases the complexity of the solutions, given that complexity is measured in terms of concepts used to describe solutions. There is also a trade off between complexity and interpretability; more complex models decrease the interpretability of the solutions.

Finally, one of the most important reasons to consider instance selection is that the number of available cases may be too large to compute in an acceptable time frame, as large volumes of data imply computational costs regarding memory allocation and complexity of learning algorithms. It has to be noted that computational complexity (run time) of most learning algorithms is linear according to the size of the training set (Provost and Kolluri, 1999).

The main approach for instance selection is data sampling (Weiss and Indurkhya, 1998). However, sampling a database, mainly a large database, is not a trivial task. The sampling method has to generate data samples that capture the necessary information from the original population, and the sampling design has to be suitable to a particular problem in a certain context.

The next section introduces some concepts and methods of data sampling.

3.6 Data Sampling

In KDD applications, data sampling is related to the available and required amount of data to build descriptive or predictive models. Sampling can be understood as a technique that selects a part to make inferences about the whole (Gu, Hu and Liu, 2001), to study the characteristics of an entire population by examining only a part of it.

Data sampling is important for several reasons. It is used to overcome problems caused by high dimensionality of attributes as well as large volumes of data. Data sampling is useful when the interests and characteristics of a population are not available or are too costly to

obtain (as the characteristics of a sample are normally easier and cheaper to obtain) or for any reason that prevents access to data about the whole population. Besides this, sampling also can reduce costs and time without reducing accuracy. Additionally, most of the data mining applications require a set of data from which to build a model (mining dataset) and another set of data on which to test it (testing dataset). For those reasons sampling is very often necessary.

In order to build a valid model is necessary that the sample reflects the whole population; the sample must be representative of the population. If this is not the case, the model does not reflect what will be found in the population and is likely to give inaccurate results. According to Gu (Gu, Hu and Liu, 2001) there are two characteristics that must be present in a sample regardless of the adopted sampling strategy: a sample must be unbiased and must have a small sampling variance.

Building data mining models requires that the dataset used for modelling capture the full range of interactions between the attributes, e.g. the *variability*. Statistically, variability is defined in terms of how far the individual instances of a population are from the mean of that population.

To ensure that a particular sample is representative of a particular population it is also necessary to not introduce any bias during the sampling procedures. For this random sampling technique has to be used. For data mining purposes it is enough to ensure that the variability present in the source dataset is, to a certain level of confidence, present in the sample dataset and any bias was introduced from the source dataset into the sample dataset (Gu, Hu and Liu, 2001; Pyle, 1999).

3.6.1 Sampling Methods

A sampling strategy requires a careful design in order to deliver a reliable estimation from its original population. If an estimate is close to the value of the population characteristics for a set of samples, so the sampling strategy is recognized as a suitable strategy in that specific case. However, when the estimate values from one sample to another varies significantly, then uncertainty is higher. This is why it is important to select a proper sampling strategy for a particular application, as different sampling strategies may produce different estimations about the same population.

Gu (Gu, Hu and Liu, 2001) presents a useful categorization of sampling methods, in which sampling methods are grouped according different perspectives, such as *non-probability* and *probability,* emphasizing the probabilistic strength of each sampling method, *general purpose* to *specific domain, one stage* to *multi-stage,* and *adaptive* to *non-adaptive.* This categorization constitutes a representative description of sampling methods, and provides a good understanding of the relationship between them and their applicability. Figure 3.5 illustrates part of the proposed categorization of sampling methods.

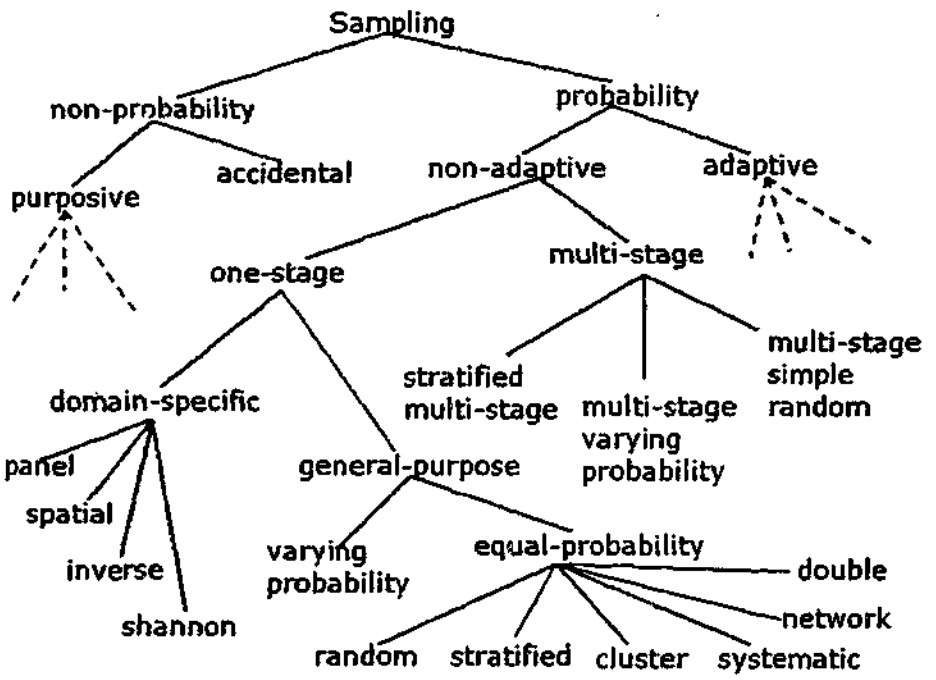

Figure 3.5: Part of categorization tree of sampling methods proposed in Gu (Gu, Hu and Liu, 2001)

A full discussion about the various sampling methods is beyond the scope of this research, however some sampling methods for data mining that were studied more deeply and are related to this research are introduced, for instance: *random sampling, stratified sampling, multistage sampling, incremental sampling, adjusting prevalence and progressive sampling.* These sampling methods are now briefly described. Further and detailed discussion about sampling methods and their categorization can be found in (Gu, Hu and Liu, 2001; 2000; Weiss and Indurkhya, 1998; Pyle, 1999), and (Foreman, 1991).

Random sampling is a method for selecting a number of units from a population in a way that each possible combination of those units has the same probability of selection, and

every element in the population has the same chance of being selected in the sample. It is the most common sampling method in use. Random sampling can be *with replacement* and *without replacement*. Random sampling without replacement is the method in which n distinct elements are selected from a population of N elements, in a way that every possible combination of n elements is equally likely to be selected (Liu and Motoda, 2001). On the other hand, in random sampling with replacement a sample of n elements is selected from a population of N elements in a series of draws; and each element may be selected more than once at any draw. This means that the elements of the population have the same chance of being drawn, irrespective of how many times they already have been drawn.

Stratified sampling consists in dividing a population in non-overlapping sub-populations, called *strata*. Then small samples are selected from different strata independently of each other. At the end, combining the small samples or individually selecting samples from each stratum forms the total sample. Stratified sampling is efficient when class values are not uniformly distributed, and one class dominates strongly in the population. In this situation examples of the smaller class are selected with a greater frequency. Stratification is suitable in cases when classifying the sampling units by location, type, and activity, for example (Provost and Kolluri, 1999).

Multistage sampling combines different sampling strategies in several sampling stages. The population is sampled in stages, where in a first stage primary samples are generated; the primary samples are then selected and divided in smaller secondary samples, which are again sampled. The process continues in this way until ultimate samplings are reached. One distinct characteristic of multistage sampling is that the sampling unit changes in the different stages, whereas in a similar sampling approach known as multi-phase sampling the sampling unit remains the same at each stage. There are several multistage sampling combinations; for example *multistage simple random sampling*, which makes simple random sampling at each stage; *stratified multistage sampling*, in which a population is first divided into strata and then sampling is performed in stages separately within each stratum.

Incremental sampling is an empirical approach to instance selection (case reduction) in which training is performed with increasing larger random subsets of cases. According to

Weiss (Weiss and Indurkhya, 1998) a typical pattern of incremental sampling might be 10%, 20%, 33%, 50%, 67% and 100% of the population. These percentages can be adjusted based on domain knowledge and the amount of information available. The initial percentage should be chosen to provide an initial idea of performance, and can be increased gradually to the complete enumeration. The decision about to select large subsets relies on various parameters, for example the training error rate. In general, when the solution performance does not improve for an increment in the sample size, there is no need to move to a bigger sample. Increasingly bigger samples are expected to give better performance, until a stage is reached where performance does not improve significantly with increase in sample size.

A similar approach of incremental sampling is discussed in (Provost, Jensen and Oates, 2001) as progressive sampling. The difference is that a sampling schema in progressive sampling requires a more formal strategy. Progressive sampling is introduced later in this section.

Adjusting prevalence method adjusts the prevalence of cases in a dataset. Given a low prevalence class, typically where the classes are not homogeneously distributed among the population, prevalence is increased by repeating or weighting cases related to the low prevalence class in the training sample (Weiss and Indurkhya, 1998). As repeating cases increases the sample size, an alternative approach is to keep the low prevalence class intact, while including a random subset of a larger class in the training sample, ideally reaching a balance of 50% of each class, in the case of binary classification. This approach is particularly suitable when the low prevalence class is more important that the larger classes, and, as such, the number of training cases may be balanced equally among the classes, increasing the cost of error for the smaller class. However, again domain knowledge is necessary to use this approach.

Progressive sampling is a strategy that attempts to maximize accuracy. It starts with small samples, instead of using an entire population, and uses progressively larger samples until the model accuracy improves. In this strategy, instance selection is limited to training sets of the minimum sufficient size, and the selection of individual instances is randomly done. The central component of progressive sampling is defined as a sampling schedule $S = \{n_0, n_1,$

n_2, ..., n_k} where n_i is an integer that specifies the size of a particular sample in a dataset with N instances, $n_i <= N$ for all samples.

According to Provost (Provost, Jensen and Oates, 2001) the fundamental issues when performing progressive sampling are determining an efficient sampling schedule, to determine the point at which accuracy no longer improves (known as convergence detection) and also to improve the schedule to achieve better performance.

The concept of convergence condition is also applied in incremental sampling, where large data subsets are selected until the performance reaches a state where it does not improve.

The convergence condition is given by the shape of the learning curve, which graphically represents the relation between model accuracy and the size of the training set for a learning algorithm. This relation is represented by a Cartesian plan where the horizontal axis represents the number n of instances in a given training set, and $0 \leq n \geq N$, where N is the total number of instances. The vertical axis represents the accuracy of a particular inductive algorithm, trained with the training set represented in the horizontal axis. Figure 3.6 shows the general shape of a learning curve.

When the accuracy no longer improves with increments in the training set size, the learning curve reaches a stable level and stops, assuming a linear shape. Therefore, it is assumed that the learning curve has converged and the training set size at this point is recognized as the smallest sufficient training set (Provost, Jensen and Oates, 2001), denoted by n_{min}.

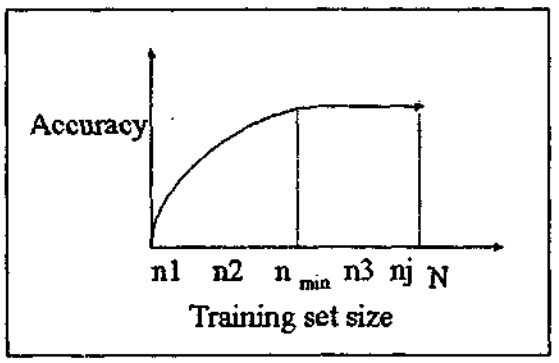

Figure 3.6: Learning curve representing the convergence condition, as in (Provost, Jensen and Oates, 2001)

Training models smaller than n_{min} will have lower accuracy, and training sets bigger than n_{min} will not presents significantly higher accuracy. Further and detailed references on theoretical and empirical studies about learning curve detection for inductive algorithms can be found in (Provost, Jensen and Oates, 2001; Catlett, 1991), and (Haussler et al., 1996).

An interesting and useful survey about suitable sampling techniques relating to specific data mining tasks is presented in (Gu, Hu and Liu, 2000). Where data mining tasks are divided into *mining association rules*, *classification* and *clustering* (a different categorisation than the ones previously presented in section 3.3.1). According to Gu, random sampling of the whole dataset is suggested as an efficient strategy for mining association rules, as it efficiently captures the population variability. For classification tasks, random sampling, windowing (Quinlan, 1987), compacting duplicates, and stratification are suitable sampling approaches.

Each sampling approach has its own particularities that must be observed. For example, stratification performs better when the class values are not uniformly distributed in the training set. For clustering, different sampling strategies can be used based on the nature of data, for example, random samplings to create an initial model to start clustering.

Reinartz (2001) and Gu (Gu, Hu and Liu, 2000) suggested that the study of sampling data for data mining is an open field for research and experimentation. Currently, there is no "cookbook" which indicates a priori which sampling approach better suits a particular data mining task, given a specific dataset, in a specific context.

3.6.2 Designing the Training Dataset

Normally, data mining applications (and most of the situations that apply learning algorithms) require a set of data from which to build the model (the training or mining dataset), and another set of data on which to test the model (the testing dataset).

The issue of how large a training dataset has to be to effectively build descriptive and predictive models remains an open research issue, as there is no definitive answer that fits all cases. According to Weiss (Weiss and Indurkhya, 1998), as a rule of thumb, the whole dataset is randomly divided into about 80% for training purposes and 20% for testing or evaluation, when the original dataset has a size of approximately 1000 instances.

However, most data mining applications deal with datasets that easily overcome the size of 1000 instances. In an attempt to obtain guidelines for building training datasets for data mining, some reported studies and experiments handling data sampling are presented in this section.

In a datamining application to find patterns describing average temperatures in northern France during September (Howard and Rayward-Smith, 1998), a database was randomly split into training and testing sets with approximately a 2:1 ratio, out of a total of 95 records, 65 were selected for training and 30 for testing.

Buchner (Buchner et al., 1998) reported an experiment where the dataset was split accordingly with a time frame in a data mining application for forecasting high-intensity rainfall over the territory of Hong Kong. In this experiment the training data encompassed the years 1990 to 1996, and the testing data encompassed the period from 1987 to 1989.

Siegler (Siegler and Steurer, 1998) reported a data mining application for forecasting of the German stock index DAX. In this application a dataset of 1000 daily observations from 1992 to 1995 was used. The dataset was split in 800 observations for training, 100 observations for validation and 100 observations as a generalization period. This means that 80% of the whole dataset was selected for training and 20% for testing purposes.

Hruschka (Hruschka and Ebecken, 1998) reported an experiment extracting rules from neural networks in a data mining application, using a dataset with 768 cases about diabetes. In this experiment 75% of the dataset was randomly selected for training and 25% for testing. The dataset had two classes; and the proportion of the classes in the original dataset was kept in the training dataset, being 65% of class 1 and 35% of class 2.

During the evaluation of Treplan, an algorithm for rule extraction from neural networks applied in data mining (Ludermir et al., 1998), a dataset with 9 inputs and 959 instances was used. This dataset was randomly divided into a training set of 879 instances and a test set of 80 instances; e.g. about 92% of the whole dataset for training and 8% for testing.

A different method was employed by Chen (Chen et al., 1998) in an experiment modelling financial data, specifically T-bond futures transaction data. In this case, a dataset with three

million records was used. After pre-processing, the dataset was randomly divided into 50% for training and the rest for testing.

In conclusion, Berry (Berry and Linoff, 2000) suggests that a split of 60% for training set, 30% for test set and 10% for evaluation set works well in practice in most cases.

Table 3.2 summarizes the experiments discussed in this section, identifying the specific experiments, the portion of the dataset used for training (mining), for testing and the size of the original database.

Table 3.2: Experiments on building training datasets

Application	Training set size	Testing set size	Dataset size
(Weiss and Indurkhya, 1998)	80%	20%	100%
(Howard and Rayward-Smith, 1998)	65	30	95
(Buchner et al., 1998)	Years 1990 to 1996	Years 1987 to 1989	
(Siegler and Steurer, 1998)	80%	20%	1000 instances
(Hruschka and Ebecken, 1998)	75%	25%	768 instances
Treplan	92%	8%	959 instances
(Chen et al., 1998)	50%	50%	3 million
(Berry and Linoff, 2000)	60%	30%	100% (10% for evaluation)

3.7 Difficulties in Knowledge Discovery Applications

In spite of the research and development that has been done and reported in the KDD field, several problems and challenges still remain. Some of these problems concern:

- developing effective means for data sampling and dimensionality reduction
- problems related to data distribution in the population, for example developing strategies to handle the low prevalence classification in data mining applications
- difficulties in determining the most appropriate set of thresholds when running data mining experiments. For example, given a specific dataset and applying an association rule learning algorithm, which degrees of rule confidence and support thresholds lead to better descriptive models

- providing integration between KDD and other complementary systems
- mining large databases.

One of the most challenging problems facing the KDD community is about scaling up data mining; this means enabling inductive learning algorithms to mine large and very large databases. The problem of mining large databases addresses issues of "how large" a problem can be feasible handled, and "how fast" a problem can be run. In this situation, there normally are two approaches to choose. One is speeding up a slow algorithm (handling the "how fast" issue). And the second is breaking the dataset (data partitioning), which means reducing the search space (handling the "how large" issue).

The number of examples (or amount of cases) has been one of the most common problems in KDD applications, as it introduces potential problems with both time and space complexity (Provost and Kolluri, 1999). In addition to time-complexity drawbacks, as the number of instances grows, problems concerning main memory size and processor speed became critical issues.

Another point to consider is the degradation of accuracy while learning; which implies that any scaling up technique must be evaluated with regard to the level of accuracy.

Choosing the data partitioning approach to mine large datasets highlights the need for developing effective means and strategies for data sampling, and dimensionality reduction.

A further problem in KDD applications relates to the applicability of knowledge discovered, and how to effectively apply such knowledge in decision making.

Once the KDD process has been completed, the knowledge discovered can be incorporated in a complementary system, such as a decision support system, providing support for business, industrial and scientific applications. This concerns the integration of complementary systems in the KDD process.

This research employs two technologies in order to devise a computational model for decision support: knowledge discovery in databases and a hybrid neural network based system. The knowledge discovery process was carefully performed in the chosen application domain, i.e., aviation weather forecasting. The following issues were addressed:

- the activities of data preparation were intensively applied with special care in data partitioning; a specific sampling design was developed and issues of data and dimensionality reduction were carefully considered
- a series of experiments were carried out in order to determine the most appropriate thresholds when applying an association rule learning algorithm, given a specific dataset. The rule confidence and support degrees, and rule order (i.e. maximum number of antecedent items) thresholds were carefully evaluated considering datasets built from different sampling proportions
- the integration of KDD with other systems, through the application of a hybrid neural network based system to build predictive models based on the knowledge discovered. As such, incorporating the knowledge discovered into a decision support model and providing means to effectively apply this knowledge in supporting decision making

The process to perform knowledge discovery in databases is described in the next chapters, together with the proposed hybrid computational model for decision support, its architecture and components. Additionally, the chosen application domain is introduced.

3.8 Chapter Summary

This chapter introduced concepts about Knowledge Discovery in Databases (KDD) and Data Mining (DM). The KDD process was described and a classification of problems suitable for KDD application was presented.

Concepts about data mining were also presented, the specific types of data mining tasks were discussed, and the classification task was introduced. Further, some of the most common data mining algorithms were introduced, and the Apriori algorithm was described in more detail.

Issues about data preparation for KDD were also presented, including dimensionality reduction and data sampling. Several data sampling techniques for data mining were presented. Finally, this chapter discussed some difficulties in developing KDD applications.

Chapter 4

4 The Hybrid DM-NN Model for IDSS

The hybrid DM-NN model was developed for the purpose of investigating the combination of knowledge discovery in database and intelligent computing technologies, in developing a framework for decision support systems. This chapter introduces the proposed hybrid DM-NN model for intelligent decision support system, its architecture, components and their respective functionalities.

4.1 Introduction

This thesis describes research concerned with investigating the combination of knowledge discovery in database and intelligent computing technologies, in particular an association rule generator algorithm for data mining and artificial neural networks, in developing a framework for decision support.

The aim of the proposed framework for intelligent decision support systems (IDSS) is to support decision making by *recalling past information, inducing "chunks" of domain knowledge* from this information and performing *reasoning upon this knowledge* in order to reach conclusions in a given *classificatory situation*. In a general way, the proposed model for IDSS has to be capable of building domain knowledge from data rich domains and applying this knowledge in problem solving.

From that perspective, this research has concentrated on investigating how *data mining* and *neural networks* can cooperate in order to minimize problems related to *knowledge acquisition*, *reasoning*, and *learning* in building decision support systems.

For this purpose, a new approach for intelligent decision support systems (IDSS) combining data mining (DM) and artificial neural networks (NN) in a single framework (DM-NN) has been developed and applied to a industry problem to empirically assess its applicability.

The data mining component is an implementation of the Apriori algorithm for association rules, as previously discussed in section 3.3.2.1. For the neural network system, the CANN (Components for Artificial Neural Networks) simulator environment (Beckenkamp, 2002) has been employed. CANN is a framework[1] that provides an environment to implement and apply neural network models.

This chapter introduces and describes the proposed hybrid architecture for IDSS. Firstly, an overview about the DM-NN model and its functionality is presented. Next, the DM-NN model architecture is described, along with its components, their roles and interactions. Following, the employed knowledge representation schema is presented, and the neural network environment and respective neural network model are also described.

4.2 Overview of the Hybrid DM-NN Model

The hybrid DM-NN model was conceived to be applied in solving complex *classification problems*, according to the definitions presented in section 3.3.1, "Data Mining Tasks." In particular, the DM-NN model is suitable for data rich domains requiring analysis of a significant amount of information, being characterized as NP-complex problems.

The DM-NN model was designed as a predictive tool for classification problems. It aims to predict a particular class in which a given case falls in within a certain degree of confidence. Examples of such situations are medical diagnoses, where the objective is to diagnose a particular disease based on a set of observed symptoms. Or weather forecasting, where the likelihood of the occurrence of a particular weather phenomenon is determined based on a set of weather observations.

[1] Here the term *framework* is used in the software engineering context, meaning "a set of cooperating classes that make up a reusable design for a specific class of software" (Gamma et al., 1995).

There are two main stages in the operation of the DM-NN model. First, descriptive models about the application domain are built. Next, predictive models of this domain are built.

Raw data are extracted from databases or data warehouse, pre-processed, and cases are obtained as a result of this process. Then, the resulting cases are used as input data into a particular descriptive method in order to build descriptive models. Descriptive models are stored in knowledge bases. An algorithm for data mining was chosen as the descriptive method. Figure 4.1 illustrates the process of building descriptive models.

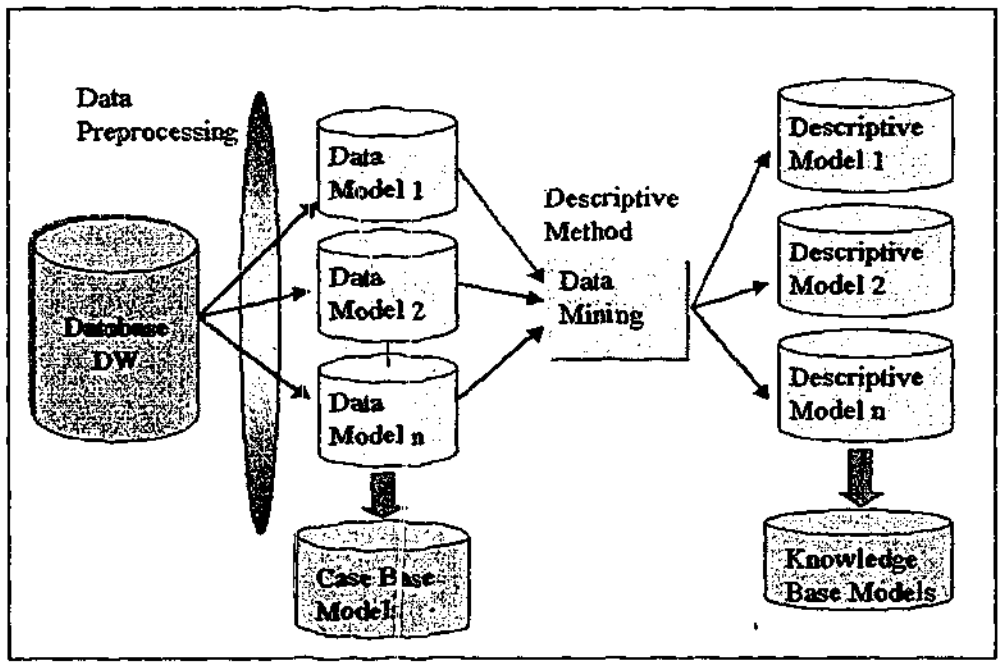

Figure 4.1: Building descriptive models

In the next stage, the descriptive models are used as input data in a predictive method. As the DM-NN model aims to solve classification problems, the predictive method is termed a ***classifier***. For this reason, a neural network model similar to the Backpropagation was selected as the predictive method. The results of the neural network processing (predictive method) are the predictive models. Figure 4.2 illustrates the process of building predictive models.

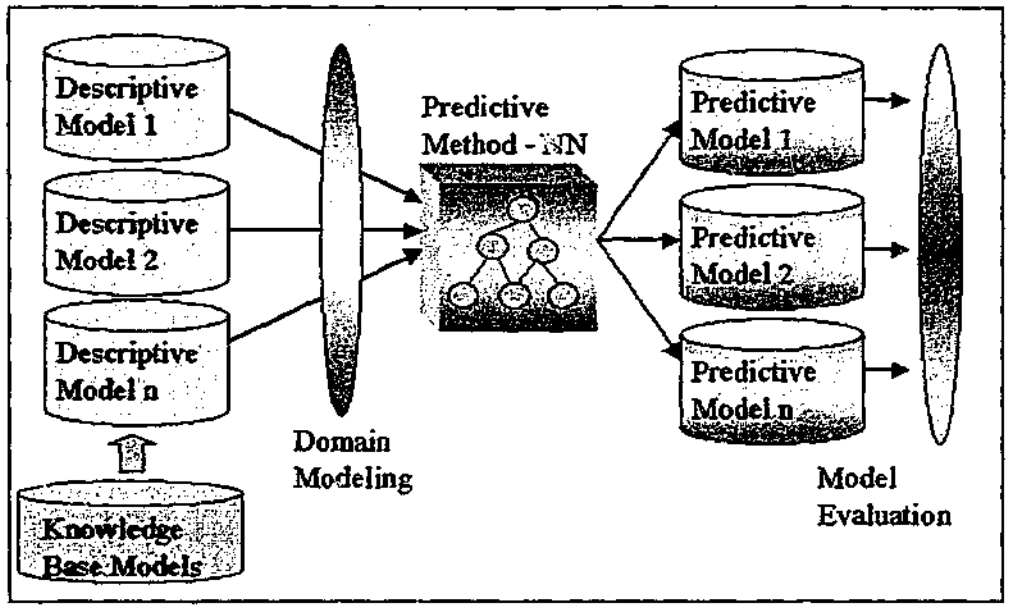

Figure 4.2: Building predictive models

Chapter 5 describes the process of building descriptive models, and Chapter 6 describes the process of building predictive models through an empirical experiment, in which the DM-NN model was applied in aviation weather forecasting to identify fog cases.

4.3 The DM-NN Architecture

This research has concentrated on investigating how *data mining* and *neural networks* can cooperate in order to minimize problems related to *knowledge acquisition*, *reasoning*, and *learning* in building decision support systems.

The DM-NN architecture was designed to capture historical information about a current situation and provide reasoning mechanisms to handle decision problems in a specific context. It is a multilayered architecture that can be divided into two levels (or perspectives): *data* and *process*. At the *process* level it applies data mining for knowledge acquisition and a neural network based system as a core for an advisory system. Specifically, data mining technology was chosen to induce expert domain knowledge from historical databases, hence minimizing the difficulties of acquiring expert domain knowledge, as previously discussed in section 2.2.2, "Intelligent Systems Capabilities" and section 2.6, "Difficulties in Developing Intelligent Systems." A NN based system is employed to implement learning and reasoning with the knowledge obtained through data mining. The NN system also provides explanatory capabilities, and the user interface level. Section 1.2, "Theoretical Background" discussed the

necessity of incorporating those capabilities in DSS models, and sections 2.2.3 and 2.6 discussed the required capabilities in DSS models.

At the *data* level the DM-NN model comprises all the data repositories used during the various stages of decision support; it includes databases (ideally a data warehouse), case bases and knowledge (rule) bases. Figure 4.3 illustrates the main components of the proposed hybrid architecture and the way they interact.

The basic computational elements of the DM-NN architecture are:

- A decision-oriented data repository, such as a data warehouse
- Case bases
- Inductive algorithm for data mining
- Knowledge bases
- An intelligent advisory system

The dotted lines in Figure 4.3 represent processes among the components; for instance, the dotted line from the data mining box indicates that a data mining process happens in order to induce domain knowledge from case bases. The second dotted line indicates the capabilities implemented by the NN based system: learning, reasoning and explanation. The dashed lines in Figure 4.3 represent data flows between the components. For example, historical raw data from databases are preprocessed and fed into the data warehouse; from where cases are selected and extracted, and then stored in case bases.

In the DM-NN model, data warehouses contain historical raw data from the application domain. Cases are built based on this data. Case bases contain selected instances of relevant cases from the specific application domain, they consist of preprocessed sets of raw (historical) data used as input in the data mining component. As such, case bases are also termed *data models (or mining datasets)*. Knowledge rule bases are built based on data mining results; they contain structured generalized knowledge that corresponds to relevant patterns found (mined) in the case bases. The knowledge obtained as a result of data mining trials is termed *knowledge models (training datasets)*, and is stored in knowledge rule bases.

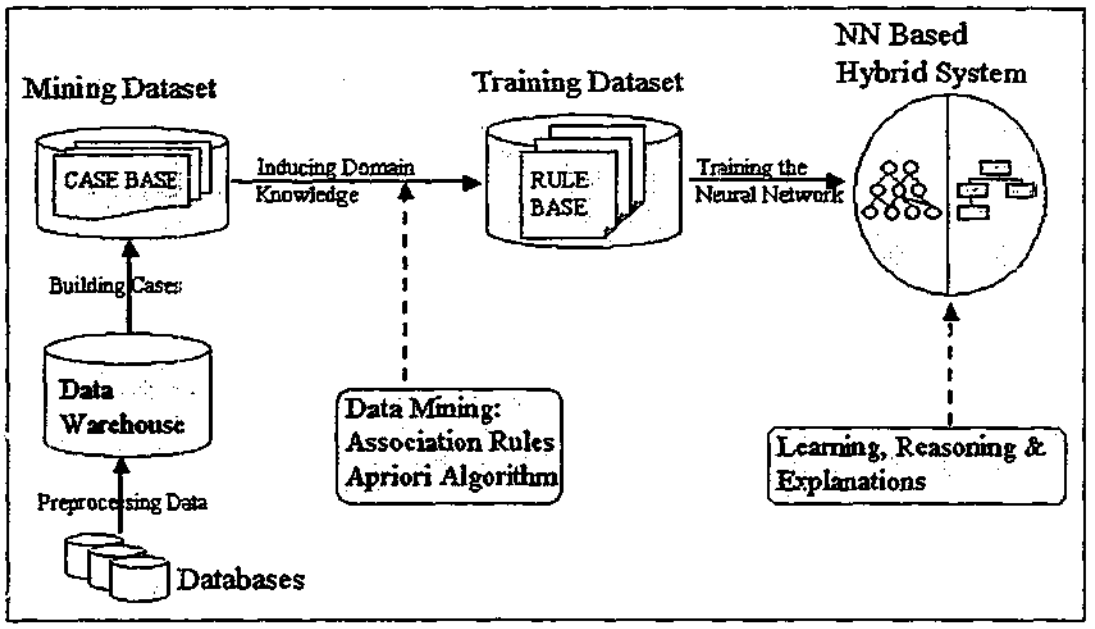

Figure 4.3: The components of the hybrid DM-NN architecture for IDSS

A hybrid (symbolic-connectionist) system is applied to process the obtained knowledge (e.g. knowledge bases), implementing learning, reasoning and explanatory capabilities. The hybrid system also provides the interface for user decision makers to test and validate hypotheses about the specific application.

Following, the components of the proposed DM-NN model for decision support and their roles are described. The next section introduces the adopted knowledge representation schema, including the neural network environment employed (CANN simulator) and respective neural network model (CNM). The functionality of the DM-NN model for decision support is explained later in this chapter.

4.3.1 Decision-Oriented Data Repository

A decision-oriented data repository is introduced in the DM-NN architecture as the primary source of information, and ideally it should be a data warehouse.

Organisational databases normally support transaction processes and contain records from applications and the results of these transactions. Therefore, most of the stored data are transactional data, and normally they are modelled and designed following the logic of the transactional process they support. Regardless of this characteristic, transactional databases are normally the main data source of information requirements for decision support systems.

According to McFadden (McFadden, Hoffer and Prescott, 1999) most organisational systems are developed to support transactional processing, with little attention given to the information needed for decision making. Transactional processing manipulates data to support daily operations, which have different data requirements than informational processing to support decision making. The data warehousing process integrates data from multiple sources into large 'warehouses' to support on-line analytical processing and business decision making (Han, 1998). A data warehouse is a decision support-oriented database that is maintained separately from the organization's operational databases (Han, 1998).

Data warehousing technology was introduced in the DM-NN model to overcome the problems related with transactional data used in high level decision support tasks, i.e. to transform transaction-oriented data to decision support-oriented data.

It should to be noted that the DM-NN model architecture does **not** specify any particular technology, as this depends on the problem being addressed and the suitability of different technologies, although the DM-NN model has been designed to specifically handle classification problems. As such, the database can be of any format, such as relational, flat files, or object oriented. And in the case of a data warehouse, it can be of any representation schema, such as a star schema, snowflake, or fact constellations. What is relevant for the data repository of the proposed DM-NN model is the quality of stored data. This relates to data availability, redundancy, and noise data, as previously discussed in section 3.4, "Data Preparation."

4.3.2 Case Base

A case base contains selected cases about the problem in concern. A case represents a past occurrence and is described by a number of attributes. A case consists of a set of feature/value pairs and a class in which the case falls. Features[1] in a case relate to each other through a logical *AND* operation, plus a feature identifying the respective class that the case belongs to.

The case base is a fundamental component in the proposed DM-NN model. The ability to build relevant cases can lead to the success or failure of a particular application. For that

[1] The terms *features*, *attributes*, and *evidences* are interchangeably used in this thesis, depending on the context. For instance, *attributes* and *features* are normally used when referring to database structures in order to keep the consistency with the jargon in that specific context.

reason it is proposed that cases should be built from the data stored in a data warehouse, as this can ensure consistency of data. In the DM-NN model case bases are also named *mining datasets*, as cases constitute the set of data used for data mining, or even *descriptive sets*, as they are employed to build descriptive models of a particular application domain.

The DM-NN model was applied in aviation weather forecasting; as such, in this research, cases are a series of weather observations. Table 4.1 illustrates an example case of fog phenomenon occurrence; the descriptions of the characteristics represent weather observations on a particular day when a definite occurrence of fog was registered (Auer Jr., 1992).

Table 4.1: Illustrating a case: a reported occurrence of fog phenomena

Description of an observed Fog case
Wind speed 4.1 meters/second
Westerly winds during the day before
Dew Point dropping from 15 C in the morning to 5 C
Easterly wind in the evening
Low cloud in the morning = 1 h
Dew point temperature = 10 C
Sea level pressure = 1022.0 hPa

In the example presented in Table 4.1, all the feature/value pairs relate to each other through a logical *AND* relationship. For instance, the feature/value pairs "Wind speed 4.1 m/s" and "Wind direction Westerly" are logically connected with all others feature/value pairs in the table, representing the weather conditions when a fog occurrence was observed.

4.3.3 Data Mining Component

The DM-NN architecture applies data mining to discover relevant relations out of the case bases in the context of the problem being addressed. In this approach specific knowledge is represented in the form of cases, from where general knowledge are derived in the form of rules.

Sets of cases (mining datasets) stored in case bases are presented to a data mining component to discover "chunks" of knowledge about a particular problem domain. Specifically, this combination of data mining and case bases is suggested to implement knowledge acquisition in the proposed decision support model (Viademonte et al., 2001b),

handling one of the bottlenecks in developing intelligent systems, the knowledge acquisition from human experts. The idea of using cases to perform knowledge acquisition is based on the assumption that a conceptualised part of knowledge about a certain domain is represented as cases (Kolonder, 1993). Consequently, it is possible to induce relevant pieces of knowledge (chunks) from a certain domain from sets of cases about that domain.

Figure 4.4 illustrates the proposed knowledge acquisition process through data mining.

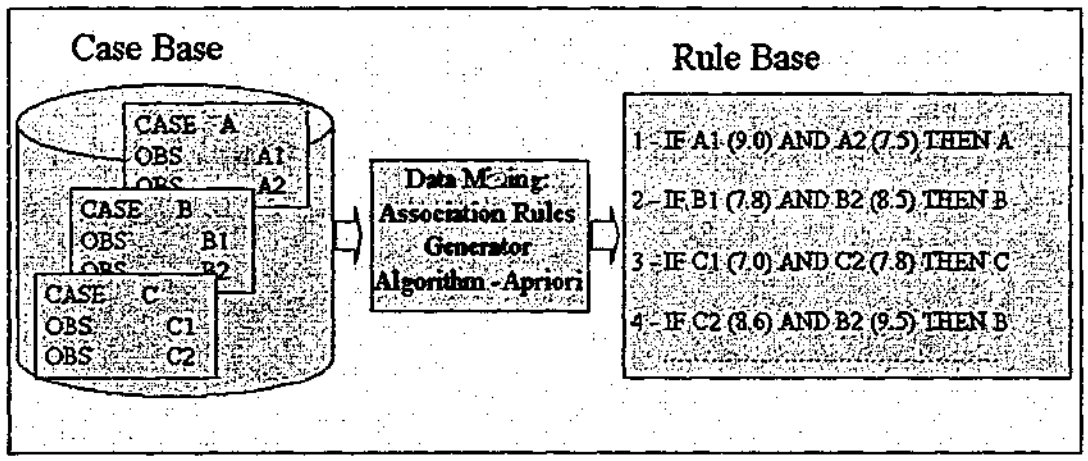

Figure 4.4: Knowledge acquisition through data mining

In the DM-NN model the relations obtained from the cases are represented as association rules and stored in knowledge rule bases. An association rules generator algorithm is employed for data mining purposes. This is an implementation of the Apriori algorithm for association rules (Agrawal, Imielinsk and Swami, 1993), already introduced in section 3.3.2. Briefly, an association rule is an expression $X \rightarrow Y$, where X and Y are sets of predicates; X being the precondition of the rule in disjunctive normal form and Y the target post condition. Association rules have two attributes, a confidence measure and a support measure. The rule confidence is the conditional probability with which predicates in Y are satisfied by a tuple (record) in the database given that predicates in X are satisfied. Such a rule is said to be frequent if its frequency exceeds a predefined threshold, e.g. if all predicates $X \cup Y$ occur together at least a user-specified minimum number of times (Agrawal et al., 1998).

4.3.4 Knowledge Base

A knowledge base contains structured knowledge that corresponds to associations found in the case bases, as a result of the data mining process. The content of knowledge bases are

sets of association rules, according to what is described later in section 4.4.2.1, "Representing Knowledge Through Association Rules."

Expert domain knowledge can be induced from historical cases stored in case bases, without using a traditional knowledge engineering approach. One of the purposes of this research is to build domain knowledge through data mining.

In the context of the DM-NN model, knowledge bases are accessed by neural networks for learning purposes, and as such they constitute the training datasets.

4.3.5 Intelligent Advisory System - IAS

It is the purpose of this research to implement the following capabilities in the DM-NN model for decision support:

- Incorporating specific domain knowledge
- Learning and reasoning
- Issuing recommendations
- Drawing justifications

This is in accordance with what has been discussed in section 1.2, "Theoretical Background" and section 2.2.2, "Intelligent Systems Capabilities."

The IAS component is responsible for the implementation of these capabilities. It is said to be advisory as it offers suggested choices to the user decision maker together with respective justifications. The IAS component was especially designed to address classification problems.

The specific architecture of this advisory system is hybrid as it combines NN within a symbolic mechanism for knowledge representation, according to what is further discussed in section 4.4, "The Knowledge Representation Schema." Furthermore, specific needs and advantages of such a hybrid approach for knowledge representation are introduced in section 2.3, "Hybrid Symbolic-Connectionist Systems."

The IAS component is capable of learning from data, and reasoning about what was learned through its neural network mechanism. And it is able to justify its reasoning through its symbolic knowledge representation mechanism, which cooperates with the NN model. As

such this component is said to be *intelligent*, according to the concepts of intelligent systems discussed in Chapter 2.

The IAS component uses knowledge stored in knowledge bases to learn about a particular problem, this is why this research treats knowledge bases (see previous section) as *training datasets.* After the neural network training process has been completed the system is capable of reasoning about the problem within the boundaries of the knowledge it obtained. It can be assumed that it had learned about the specific problem and it is ready to be used as an advisory decision support system. Figure 4.5 shows an overview of the internal architecture of such a system, its main components and processes.

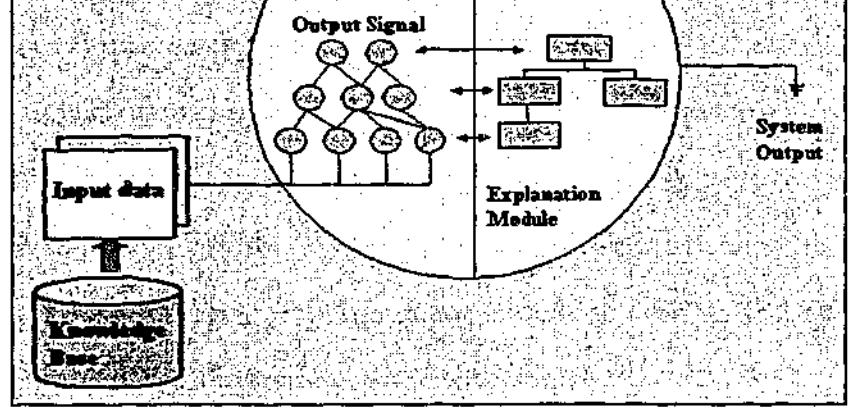

Figure 4.5: Internal architecture of the IAS component

At the end of its operation the IAS component presents an evaluation of the situation it was asked to process in the form of possible solutions, as well as explanations of its output.

The CNM neural network model was employed to implement the IAS component, CNM is later discussed in this chapter, as well as the reasons it was selected in this research.

Briefly speaking, the IAS component implements learning through the CNM neural network model. The CNM implements two learning algorithms, namely the Incremental Reward and Punishment (IRP) algorithm and the Starter Reward and Punishment (SRP) algorithm.

In the CNM, synapses have weights and pairs of accumulators for punishment and reward. During the learning phase (training), as each example is presented and propagated, all links

that led to the right classification have their reward accumulator incremented, otherwise, misclassifications increment the punishment accumulators. At the end of the learning phase, connections with higher punishment values than reward values are pruned. The remaining connections have their weights updated using the accumulators. The IRP and SRP algorithms are introduced in section 4.6, in this chapter.

Once the CNM is trained, it pursues the following strategy to come up with a decision for a specific case. The CNM evaluates the given case and calculates a confidence value for each hypothesis. The inference mechanism finds the winning hypothesis, the one with the highest confidence value, and returns the corresponding result.

Moreover, the IAS component is implemented through a NN environment that is able to access the implicit information stored in the NN structured through a design that combines NN models with a symbolic mechanism for knowledge representation. As a result of this symbolic knowledge representation, the IAS is able to draw justifications about its output. Once a particular output neuron is fired, the IAS recovers the input neurons and the pathway that led to the result, identifying explicitly the information content of those neurons.

The CNM model and its algorithms are introduced in section 4.6 in this chapter. Next, the DM-NN knowledge representation schema is explained, as well as the environment for the IAS component in which the CNM model was implemented.

4.4 The Knowledge Representation Schema

The knowledge representation schema employed in the DM-NN model is grounded on the knowledge representation formalism of knowledge graphs, the main goal of which is to provide means for the representation and combination of knowledge elicited from multiple experts (Leao and Rocha, 1990).

4.4.1 Knowledge Graphs

A Knowledge Graph (KG) is defined as a directed AND/OR acyclic graph used to represent the knowledge of an expert or group of experts for a particular classification hypothesis.

There are three types of nodes in a KG:

- *hypothesis nodes* represent the hypotheses, or classes, considered in the graph
- *evidence nodes* represent input information that support a particular hypothesis. Evidence nodes are placed in the graph in their order of importance, from left to right
- *intermediate nodes* represent different groupings of evidences that leads to a specific hypothesis or class. These groups of evidence represent *chunks* of knowledge applied by an expert when reasoning about a problem. Intermediate nodes represent a logical *AND* operation among the evidence nodes linked to them

Figure 4.6 illustrates the structure of a knowledge graph.

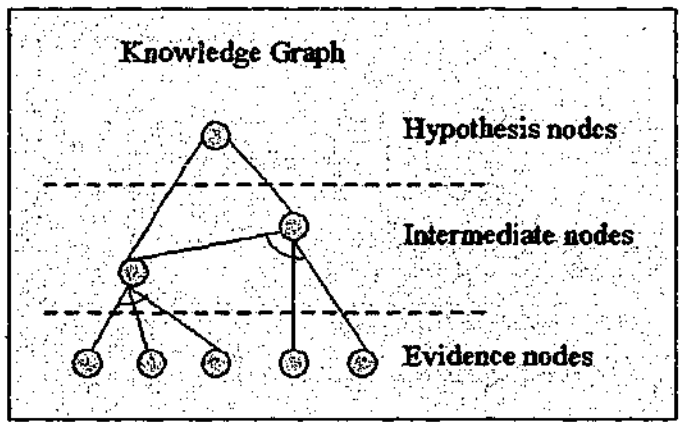

Figure 4.6: The basic structure of a knowledge graph

The knowledge acquisition methodology of knowledge graphs (Leao and Rocha, 1990) was developed based on knowledge graphs formalism, with the aim of providing the means of eliciting and representing knowledge from multiple experts.

Several experiments have been developed using the knowledge graph methodology for knowledge acquisition. For example, Leao (Leao and Rocha, 1990; Leao, 1988) applied this methodology in eliciting and representing expert knowledge in the domain of congenital heart diseases, in a research conducted at the Cardiology Institute (Brazil, RS). Later, this methodology was applied in the development of the **HYCONES** system, a hybrid connectionist expert system developed for the diagnosis of Congenital Heart Diseases (Leao and Reategui, 1993b).

Furthermore, this methodology has also been applied in the development of the **SECOX-HI** system, a hybrid model for expert systems developed for classification of operation states of (reservoir) floodgate manoeuvres, in a hydroelectric power plant (Viademonte, 1994; Viademonte, Leao and Hoppen, 1995), and (Viademonte, Hoppen and Beckenkamp, 1997).

The knowledge representation formalism of knowledge graphs also has been the main motivation and basis for the development of the Combinatorial Neural Model (CNM) (Machado and Rocha, 1989; Machado and Rocha, 1990), previously introduced in Chapter 2.

Figure 4.7 shows an example of the use of a knowledge graph for the representation of the *operation state of emergency* for floodgate manoeuvres, one of the various operation states used to classify floodgate manoeuvres. In this domain modelling the operation states are the hypotheses (or classes) under representation.

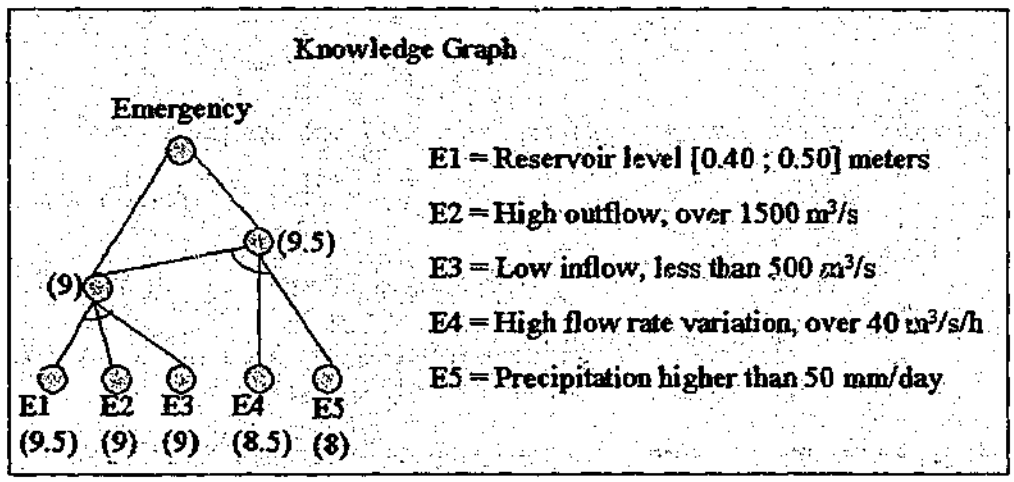

Figure 4.7: Knowledge graph for the operation state of emergency as in (Viademonte, 1994)

Figure 4.7 shows two different pathways that lead to the hypothesis of emergency. The first pathway consists of the three leftmost nodes of the graph (E1, E2, and E3) connected by a logical *AND* operation through an intermediate node. The second pathway consists of the node E5 and E4 connected by a logical *AND* to the first pathway. The two different pathways can be described as two different alternatives to support the hypothesis of emergency, which means they became connected by a logical *OR* operation.

The numbers associated to each input node are the importance degrees of the information represented by the nodes. This number is normally assigned by a domain expert.

The knowledge acquisition methodology for the construction of knowledge graphs starts with a series of interviews with domain expert(s), who determine the set of problems to be handled, as well as the information (or evidences) that influence the identification of each problem. For example, in the weather forecasting context, the expert would have to identify the weather phenomena that the system would try to classify, as well as the weather observations and measurements that contribute to the identification of each selected phenomena. For each of the selected problems, the expert would then have to sort the evidences according to their order of importance, and place them in the evidence layer (bottom layer) in the KG. The hypotheses are placed as hypothesis nodes in the top layer. Next, the evidence nodes that have some degree of importance when grouped together are connected in intermediate nodes, which are then connected to the hypothesis nodes. Finally, in the last phase, an importance degree value in a range between 0 and 1 has to be assigned to each node.

The knowledge acquisition methodology for the construction of knowledge graphs is described in Appendix A.

4.4.2 Representing Domain Knowledge

Domain knowledge is represented in three ways in this research:

- through association rules
- through a neural network model, e.g. implicit in the neural network structure
- through a hierarchy of classes and objects

4.4.2.1 Representing Knowledge Through Association Rules

Although knowledge graphs constitute a powerful approach for knowledge acquisition and representation in classification problems, its construction is time consuming and involves a costly process, requiring the assistance of at least one domain expert. This could be observed during the development of the **SECOX-HI** system and its application in the domain of floodgate manoeuvres, in which the knowledge acquisition methodology for the construction of knowledge graphs was employed to acquire expert domain knowledge (Viademonte, 1994; Viademonte, Leao and Hoppen, 1995).

Consequently, the automatic construction of KGs and similar knowledge representation formalism would represent a potential solution for the problem of knowledge acquisition (when applying those knowledge representation formalisms), particularly in situations in which domain experts are not available.

Considering Figure 4.7 as an example, each pathway in the graph can be translated to a rule representation, associating the respective evidence nodes. For example, the first pathway in Figure 4.7 can be represented as follows:

*If **E1** AND **E2** AND **E3** THEN **EMERGENCY** with 90% degree of confidence.*

Or in a more explicit fashion:

If *Reservoir level is in the range of [0.40 ; 0.50] meters* *AND*
High outflow, over 1500 $m^3/2$ *AND*
Low inflow, less than 500 $m^3/2$
THEN the state is emergency with 90% degree of confidence.

This notation is very similar to the notation of association rules, as described in section 3.3.2.1. An association rule is an expression $X \rightarrow Y$, where X and Y are sets of predicates; X being the precondition of the rule in the disjunctive normal form and Y the target post condition. As such, the association rule representation of the first pathway of Figure 4.4 can be described as:

$X = \{E1, E2, E3\}$, and $Y = \{ EMERGENCY \}$, *rule confidence= 90%*
Where $X \rightarrow Y$

Consequently, knowledge graphs can be represented by sets of association rules, similar to the ones generated by the Apriori algorithm (Agrawal, Imielinsk and Swami, 1993), as discussed in Chapter 3. Furthermore, association rules can be automatically induced from cases, through an association rule generator algorithm. This approach might represent a potential solution to the problem of knowledge acquisition and representation through knowledge graphs.

For that reason, association rules were chosen as the formalism to represent domain knowledge in this research. As a result, an association rule generator algorithm was selected for data mining in this research.

Therefore, association rules were chosen as the knowledge representation formalism in the DM-NN model for their similarity with knowledge graphs, because they represent a clear and natural way of knowledge representation that is easy for people to understand, because they easily represent simple causalities which are suitable for the meteorological domain, because there are efficient algorithms for association rules discovery, and finally because they fit smoothly into the selected neural network model, the CNM.

4.4.2.2 Representing Knowledge Through Neural Networks

Neural networks (NN) are good at implementing lower level reasoning. They excel in recognising complex patterns, learning and generalization from examples and have powerful self-organizing capabilities (Zurada, 1992). Furthermore, NN models like Backpropagation and CNM have been largely applied in classification problems, as discussed in Chapter 2.

Neural networks, particularly the CNM model, were the selected approach to implement learning and reasoning capabilities in this research. The CNM was selected because of its compatibility with KGs and, as a result, association rules. Additionally, the CNM has been successfully employed in several experiments dealing with classification problems, such medical diagnoses (Leao and Reategui, 1993b), credit card scoring (Reategui and Campbell, 1995) and engineering problems (Viademonte, Leao and Hoppen, 1995). Another reason for choosing the CNM model is that this author has experience in applying and working with CNM, and one of the motivations of this research was to continue with previous work this author have been involved in applying NN, particularly the CNM, in building intelligent systems.

The knowledge acquisition methodology of KGs has been the main motivation and the basis for the development of the CNM (Machado and Rocha, 1989). Therefore, the structure of the CNM is very similar to that of the graphs.

The CNM is usually implemented with a three layer topology: an input layer, a hidden layer (also named combinatorial layer), and an output layer. As such, KGs can be directly mapped into the CNM topology.

The *hypothesis nodes* in the KG are mapped into the CNM's *output layer*, so the hypothesis nodes became the output neurons in the neural network. *Evidence nodes* in the KG become the *input neurons* in the neural network, and *intermediate nodes* are mapped on to the CNM's *hidden (combinatorial) layer*.

Regarding its similarity with the KGs, association rules can also be mapped into the CNM topology. In this case, each evidence/attribute value pair corresponding to a rule's antecedent items is mapped on to an input neuron in the CNM topology. The right side of the rule, e.g. the consequent item, is mapped on to an output neuron; and the rules correspond to the strengthened connections among the input nodes, e.g., the CNM combinatorial (hidden) layer. For instance, rules describing relations in the weather forecasting domain (described in Chapter 5) are represented by neurons and synapses. Figure 4.8 illustrates this property.

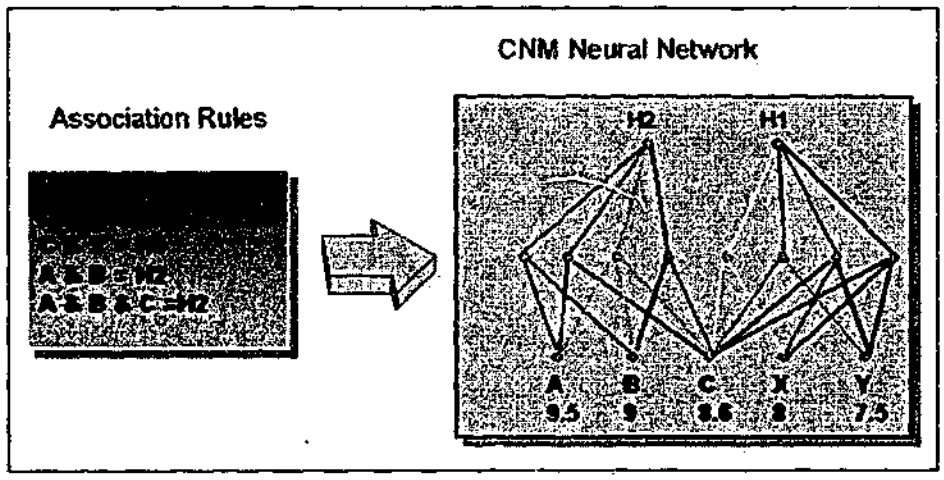

Figure 4.8: Mapping rules into the CNM topology

In Figure 4.8, the antecedent items of the rules, the evidences *A*, *B*, *C*, *X* and *Y*, are represented as input neurons in the CNM structure, and the hypotheses *H1* and *H2* are represented as output neurons. For instance the rule antecedent item *A* is mapped on the first input neuron, item *B* is mapped on the second input neuron, and item *C* is mapped on the third input neuron. Additionally, importance degree values can be assigned to each rule antecedent item, similarly to the importance degree values assigned to the evidence nodes in the KGs. These importance degree values can be then transferred as input neuron weights in the CNM structure. Synaptic weights are calculated by the CNM algorithms, and hidden neurons correspond to combinations of evidences, representing rules. The leftmost hidden

neuron in Figure 4.8 represents the rule *If A and B than H2,* indicating to hypothesis *H2,* represented by the output neuron with the same name.

The CNM was the selected neural network model in this research because of its generalization, learning from examples, and self-organizing capabilities, by its compatibility with the KGs and association rules, and the facility of translating association rules into its topology. Furthermore, the CNM has been successfully employed in several experiments dealing with classification problems.

Section 4.6 describes the CNM model in more detail.

4.4.2.3 Representing Knowledge Through a Hierarchy of Classes and Objects

Accordingly to what was previously discussed in Chapter 2 (see Table 2.1), in neural network models knowledge is implicitly represented as connection weights distributed across the NN topology. In such a knowledge representation schema it is very difficult to explicitly access that knowledge for explanatory purposes. To minimize this problem, the CANN simulator was selected as the NN environment in the DM-NN model.

The Components for Artificial Neural Networks (CANN) environment is a research project that concerns to the design and implementation aspects of a framework architecture for decision support systems that rely on artificial neural network technology (Pree, Beckenkamp and Rosa, 1997; Beckenkamp, 2002). The CANN project concerns the creation of basic NN components to implement different NN models, and also to support problem domain modelling.

The CANN architecture combines NN models with a symbolic mechanism to represent the NN structure. The NN structure including the knowledge stored across that structure is symbolically represented in a hierarchical fashion, through an object-oriented design that reflects common properties of classification problems. For instance, the evidences form the input data (in the same way KGs represent evidence nodes). Experts use evidences to analyse the problem in order to be able to come up with decisions.

Evidences in the case of the aviation weather forecasting would be wind speed and direction, for example. On the other hand, the classification categories, e.g. hypotheses, constitute a further core entity of classification problems. In aviation weather forecasting,

hypotheses would be fog occurrence, thunderstorms and cyclones, for example. In CANN, an instance of class *Domain* represents the problem by managing the corresponding *Evidence* and *Hypothesis* objects. Figure 4.9 illustrates an object-oriented representation of the weather forecasting domain.

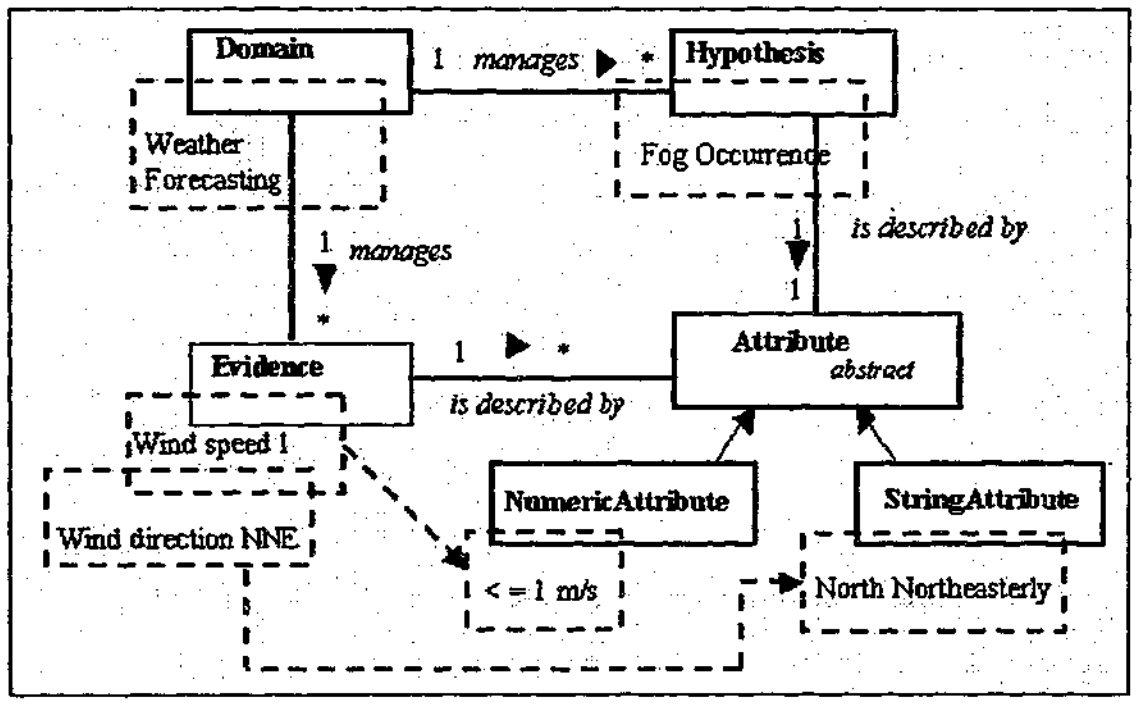

Figure 4.9: Object-Oriented representation of the aviation weather forecasting domain

The domain of *aviation weather forecasting* is modelled as an instance of *Domain* class. The *Domain* class manages hypothesis and evidences associated with a particular domain; and both evidences and hypothesis are described by their attributes, which can be of different types, e.g., numeric, fuzzy and string. Figure 4.9 shows two evidences associated with the hypothesis of fog occurrence, which is an instance of the *Hypothesis* class. In Figure 4.9, evidence instances are *wind speed 1*, and *wind direction NNE*. Evidence instances are described by their respective attribute instances; for example, *wind speed 1* is described by an instance of numeric attribute associated with *wind speed lower or equal to 1 meter/second (< = 1 m/s)*. In a similar way, the evidence instance *wind direction NNE* is described by an instance of a string attribute associated with wind direction *North Northeasterly*.

Besides implementing common properties of classification problems, the object-oriented design allows the representation of abstract concepts such as generalisation, classification and

aggregation; providing the necessary flexibility for knowledge modelling. Further discussion about the representation of abstract concepts and their integration with neural network models is provided in (Leao and Rocha, 1990; Reategui, Campbell and Leao, 1997).

The concepts and ideas discussed in this section, inherent to the CANN simulator, make it a suitable choice to implement the Intelligent Advisory System (IAS) component in this research. Another reason for choosing CANN is because this author has been working in the CANN project development and its application, and is thus very familiar with its concepts and functionalities.

The next section introduces the CANN simulator.

4.5 Components for Artificial Neural Network – CANN

The Components for Artificial Neural Networks (CANN) is a research project that relates to the design and implementation aspects of a framework architecture for decision support systems that rely on artificial neural network technology (Pree, Beckenkamp and Rosa, 1997; Beckenkamp, 2002). The CANN components are object-oriented designed; hence the core parts are done as small frameworks to improve implementation reusability and flexibility.

The CANN project concerns the creation of basic NN components to build different NN models; the creation of NN components that can be reused by third parties; and the construction of a simulation infrastructure that allows to plug several NN components and use/test them (Beckenkamp and Pree, 2000; Pree, Beckenkamp and Rosa, 1997). Additionally, there is a component created to support problem domain modelling and access to data sources for the NN learning and testing process.

At the time this thesis was written, the implemented NN models in the CANN framework were (Beckenkamp, 2002): the Backpropagation (Rumelhart and McClelland, 1986), Combinatorial Neural Model (CNM) (Machado and Rocha, 1990), the Self-Organizing Feature Maps (SOM) (Kohonen, 1982), and the Adaptive Resonance Theory (ART) (Carpenter and Grossberg, 1987). These models were chosen because they cover almost all learning strategies and NN topologies. For example, Backpropagation implements a supervised learning based on the error correction learning algorithm, and its architecture is feedforward multilayer and it is fully connected. The CNM model also implements supervised learning based on a variation of

the error Backpropagation learning algorithm (see section 2.4.5), but the network is feedforward and not fully connected. For more details about the NN models implemented in CANN and their motivations, refer to (Beckenkamp and Pree, 2000) and (Beckenkamp, 2002).

Besides keeping the design open for supporting various NN models, a smooth integration of NN technology into a decision support system forms another important design goal within the CANN project. Thus the main goals of the CANN project are (Beckenkamp, 2002): the *description of flexible and reusable components for core aspects of neural networks implementations*, the *integration of different neural network models in a decision support system*, and *the presentation of a decision support system architecture that can be easily adapted to handle different domain problems.* As such, the CANN simulator is well suited to implement the IAS component in the decision support model proposed in this research.

The next section introduces the CANN architecture and its main components. Full discussions about CANN, including motivation and implementation aspects can be found in (Beckenkamp, 2002).

4.5.1 The CANN Architecture

The CANN project was completely designed based on framework construction principles (Gamma et al., 1995; Pree, 1995), in order to reflect the necessary building blocks for creating different NN architectures. The design takes into consideration the flexibility for reusing the core entities of an NN. Additionally, it is fully implemented in Java programming language (Sun Microsystems, 2004).

There are five frameworks implemented in CANN: the *Simulation framework* (which is why it is called the CANN *simulator*), the *Neural Network*, *Domain*, *Data Converter* and *GUI*. This thesis introduces the Neural Network and Domain frameworks, as they are particular relevant to this research. The GUI framework is introduced in Chapter 6, through the application of CANN in aviation weather forecasting.

The Domain framework implements a generalized way of defining problem domains; as a result, it can be applied to any NN model and domain problem.

A NN framework is defined in order to facilitate the implementation of different NN models. This is achieved by modelling the core entities of the NN (neurons and synapses) as

objects and storing the generated NN topologies as objects via Java's serialization mechanism. As such, CANN is able to reuse these core NN components for implementing new NN models.

This section introduces the CANN architecture and design principles in order to illustrate how CANN was applied in this research. Readers interested in detailed discussion about CANN design should refer to (Beckenkamp, 2002). This thesis also assumes that readers are familiar with concepts such as object orientation, frameworks and design patterns. Readers interested in further discussion in these subjects should refer to (Pree, 1995) and (Gamma et al., 1995).

4.5.1.1 Object-Oriented Modelling of Neural Networks

Neurons and synapses form the basic building blocks of NN models (see section 2.4). CANN provides two classes[1], *Neuron* and *Synapse*, whose objects correspond to these entities. Both classes are abstract and offer properties that are common to different neural network models. The idea is that these classes provide basic behaviour independent of the specific NN model. Subclasses add the specific properties according to the particular model.

An object of the *Neuron* class provides methods to calculate its activation and to manage a collection of *Synapse* objects that process outgoing signals of a neuron. A *Synapse* object represents a directed connection between two neurons, a receptor and source neurons (see Figure 4.10).

The receptor neuron manages a list of incoming synapses (represented by the solid arrows in Figure 4.10) and computes its activation from these synapses. A *Synapse* object has one *Neuron* object connected, which is the source of the incoming sign. The dashed arrows in Figure 4.10 represent the computational flow (the set of incoming signs from all source neurons). The incoming signal from one source neuron is processed by the synapse and forwarded to the receptor neuron on the outgoing side of the synapsis. This *Neuron* object computes its activation from all incoming synapses.

Through this mechanism different neuron network topologies can be built, for example, a multilayer feedforward or recurrent networks. Each NN model is responsible for its

[1] The terms *classes* and *objects* here, are referred in the Object-Orientation context.

topological construction; distinct models use *Neuron* and *Synapse* classes as their basic building blocks for the neural network structure and behaviour construction.

For example, in the case of implementing a multilayer feedforward neural network model, all necessary neurons are generated. Following, the synapses to connect the neurons at different layers are generated and connected to their respective neuron layers.

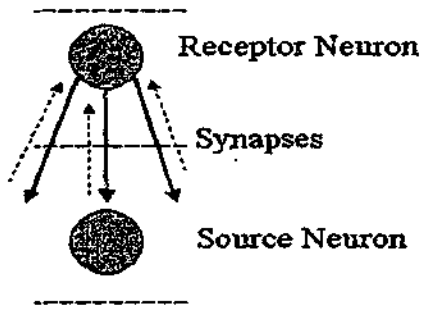

Figure 4.10: The relationship between Neuron and Synapses objects, as in (Beckenkamp, 2002)

The abstract methods defined in *Neuron* class are responsible for generating synapse instances and their appropriate connection to source neurons. Specific methods for different NN models are implemented in the abstract class that defines a particular neural network model, which implements its respective subclasses. Through this mechanism of class inheritance, different neural network models can be implemented. Figure 4.11 shows part of the class hierarchy derived from the *Neuron* class, implemented in the neural network framework.

Figure 4.11 illustrates the instantiation of a Backpropagation and CNM models. Specific behaviour from Backpropagation and CNM neurons are implemented in respective Neuron subclasses, namely, *BPNeuron* and *CNMNeuron* and their respective derived subclasses, according to each neuron functionality, i.e. input, output, or hidden neuron (Beckenkamp, 2002).

The class hierarchy for the *Synapse* class follows a similar fashion to the *Neuron* class. In both class, hierarchy subclasses are created to implement each specific NN model. For example, specific *Synapse* subclasses in the CNM model include punish and reward accumulators and a method to implement them. Such methods are implemented by the CNM synapse subclass.

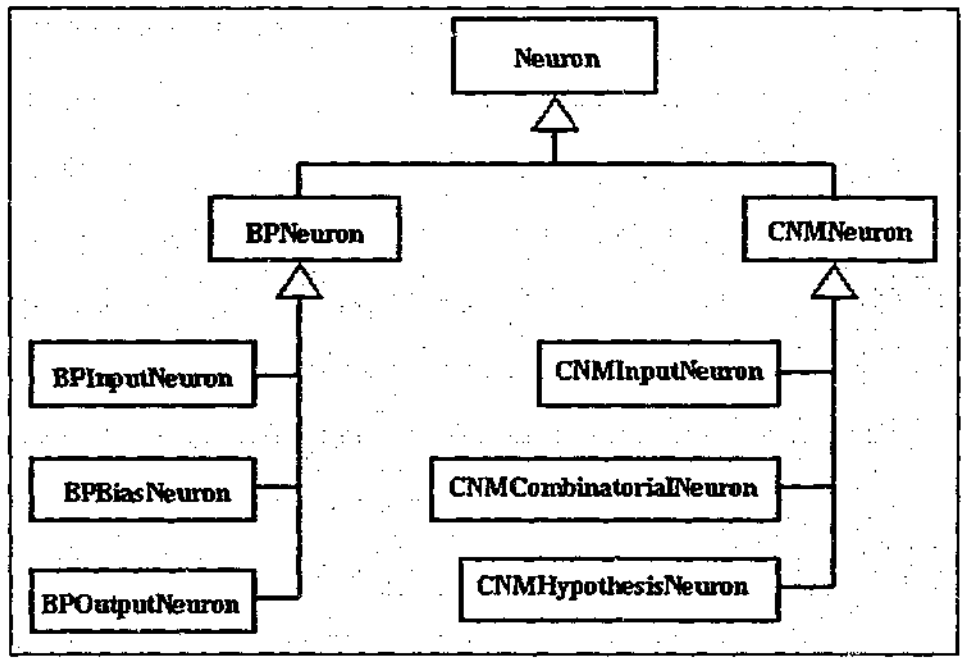

Figure 4.11: Part of the Neuron class hierarchy implemented in CANN, as in (Beckenkamp, 2002)

CANN provides a schema to separate specific NN implementations from the application domain implementation. For that an abstract class named *NetManager* and an interface named *InetImplementation* were defined. *NetManager* defines implementations that do not belong to NN models in particular, such as manipulating the domain knowledge, and fetching test and training datasets. An *INetImplementation* object represents the inference engine (e.g., neural network model) of the running CANN system. Its interface reflects the needs of the decision making process under consideration in a particular CANN application.

Flexibility in implementing NN models is one of the main goals of the CANN project. This means that the design should allow the experiment of different NN models, in a simultaneous fashion. The *NetManager* class, together with the *INetImplementation* interface, implements this feature, and are responsible for controlling a set of instances of different neural network models, or even a set of instances of the same neural model but with different configurations.

A specific NN model is defined by subclassing *INetImplementation* and overriding the corresponding hook methods.

4.5.1.2 Domain Representation

CANN was initially developed on ideas and applications of applying NN in classification problems (Pree, Beckenkamp and Rosa, 1997), thus the chosen object-oriented design of this system aspect reflects common properties of classification problems. In CANN, the domain is represented through four main classes: *Domain*, *Evidence*, *Hypothesis*, and *Attribute*. Figure 4.12 illustrates the class hierarchy for domain representations.

Evidences form the input data, and experts use evidences to analyse the problem in order to arrive at decisions. Evidences in aviation weather forecasting would be the level of rainfall, wind speed and direction, for example. Evidences are described by their respective attributes, so the *Attribute* class was incorporated in the framework. One or more *Attribute* objects describe the value of each *Evidence* object.

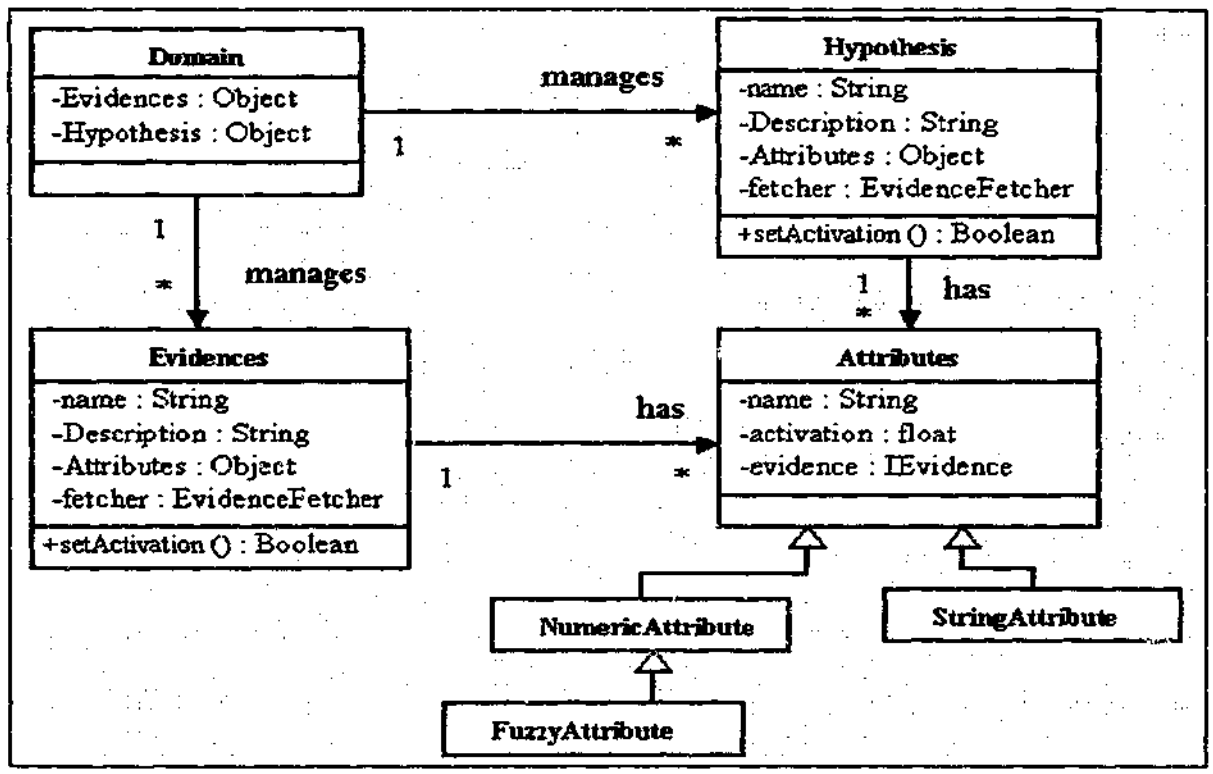

Figure 4.12: Class hierarchy for domain representations, as described in (Beckenkamp, 2002)

For example, in the case of the evidence wind direction, this might be defined as a set of string values (string attributes) such as *North*, *Northwest*, *South*, *Southwest*, etc. The evidence level of rainfall might be defined by a numeric attribute, such as *7* (*7 mm/day*) and the evidence wind speed defined as a fuzzy set (Kosko, 1992; Viademonte, Hoppen and Beckenkamp,

1997) of values: *light, moderate, light to moderate, fresh, strong* and *galeforce*. Consequently, the *Attribute* class is subclassed according to the different data types an attribute might hold, such as numeric, string and fuzzy sets (see Figure 4.12).

Furthermore, the classification categories (or hypotheses) constitute a further core entity of such problems. In aviation weather forecasting, hypotheses would be fog occurrence, thunderstorms, cyclones, etc. The *Hypothesis* class represents the possible classes (or hypotheses) involved in a particular application.

In CANN, an instance of class *Domain* represents the problem by managing the corresponding *Evidence* and *Hypothesis* objects. Figure 4.12 shows the class hierarchy involved in the problem domain representation.

4.5.1.3 Data Conversion

Training and testing of an NN model are important features, and for both tasks data have to be provided. For training an NN to identify a particular weather pattern, data might come from an ASCII file. One line of that file represents a set of weather observations, which are the atmosphere evidences plus the correct pattern classification. After training the NN, weather data should be tested, that is, CANN takes the evidences of a particular weather observation as input data and has to classify it. The data source might in this case be a relational database management system. It should be clear from this scenario that CANN has to provide a flexible data conversion subsystem.

Besides the Neural Network and Domain frameworks, CANN defines a framework for processing problem-specific data. *Fetcher* and *EvidenceFetcher* abstract classes constitute the framework for processing problem-specific data (Beckenkamp, 2002). Class *Fetcher* is abstractly coupled with class *Domain*. A *Fetcher* object is responsible for the preparation/determination operations associated with a data source. For instance, if the data source is a plain ASCII file, the specific fetcher opens and closes the file.

The *Evidence* and *Hypothesis* classes are abstractly coupled with the *EvidenceFetcher* class. Specific subclasses of *EvidenceFetcher* are responsible for accessing the data for a particular evidence. For example, a converter for reading data from an ASCII file stores the position

(from column, to column) of the data in the ASCII file. An *EvidenceFetcher* for reading data from a relational database would know how to access the data by means of SQL statements.

Readers interested in a more comprehensive and detailed discussion about the CANN project and framework should refer to (Beckenkamp, 2002; Pree, Beckenkamp and Rosa, 1997), and (Beckenkamp and Pree, 1999).

In order to evaluate the proposed decision support model in this research, the CNM neural network model was employed (for reasons previously discussed in section 4.4.2) as it is implemented in CANN. The next section introduces the CNM NN model and its main algorithms.

4.6 The Combinatorial Neural Model - CNM

The CNM network (Machado and Rocha, 1990) was designed to provide a computerised method that would use the same reasoning model as knowledge graphs; consequently it is well suited for classification problems (Leao and Reategui, 1993a). According to Reategui (1997) by having a neural network with a similar structure to that of knowledge graphs, machine learning and knowledge engineering technologies could cooperate in the development of intelligent systems.

The CNM neural network model was selected for this research for its generalization, learning from examples, and self-organizing capabilities; by its compatibility with the KGs and association rules, and, consequently, by the facility CNM offers in translating association rules into its topology, as previously discussed in section 4.4.2.2, "Representing Knowledge Through Neural Networks."

In the CNM, domain knowledge is mapped on to the network through evidences and hypotheses, in a similar fashion to KGs. The CNM has a feedforward topology, usually implemented with three layers: an *output layer*, an *input layer*, and a *hidden layer* (also named the combinatorial layer). The output layer represents the hypotheses (or classes) involved in a particular problem, where each neuron (node) represents a single hypothesis. The input layer represents the evidences, used to support a particular hypothesis. Each input neuron represents an evidence/value pair, and an evidence might have many values. The

combinatorial layer represents different combinations of evidences that lead to a specific hypothesis (Machado and Rocha, 1990). The topology of a CNM model is multi-layer feedforward, and it is similar to that illustrated in Figure 2.5.

Evidences and hypotheses are assigned by a domain expert or a knowledge engineer. The combinatorial layer, otherwise, is automatically generated. A combinatorial neuron is created for each possible combination of evidences that lead to a specific class, from order 1 to a maximum order predefined by the user.

Input neurons are formed by fuzzy values in the interval [0,1], indicating the degree of confidence (or measure of relevance) of the information represented by each input neuron.

Neurons are linked by connections, e.g. synapses. CNM implements two types of synapses: *excitatory* and *inhibitory*. Excitatory synapses propagate an input signal using their synaptic weight X as an attenuating factor. Inhibitory synapses implement a fuzzy negation on the arriving signal, transforming it into 1-X. Then signals are propagated by multiplying its value by the synaptic weight.

Combinatorial neurons propagate incoming values according to a fuzzy *AND* operation. The output value Y of a fuzzy *AND* operation corresponds to the minimal arriving value, e.g., the smallest value obtained from the product of input signals with their corresponding synaptic weights (Equation 4.1) (Machado and Rocha, 1992):

$$Y = \min\{W_i \times X_i\}$$

Equation 4.1: Fuzzy AND operation

In Equation 4.1, $i \subseteq \{1,..., n\}$, indicates input neurons, Xi is the input signal of i_{th} input neuron, and Wi its corresponding synaptic weight.

The output layer implements a competitive mechanism among the pathways that connect to each output neuron, propagating the maximum of their incoming values through a fuzzy *OR* operation. The output value Y of a fuzzy *OR* operation corresponds to the maximum arriving value from the lower layer, e.g., the highest value obtained from the product of input signals with their corresponding synaptic weights (Equation 4.2) (Machado and Rocha, 1992).

$$Y = \max\{W_i \times X_i\}$$

Equation 4.2: Fuzzy OR operation

In Equation 4.2, $i \subseteq \{1,..., n\}$, indicates the output neurons, Xi is the incoming signal of i_{th} output neuron, and Wi its corresponding synaptic weight.

CNM modifies weight values through a supervised learning algorithm (section 2.4.5) that aims to minimize the mean square error (Kosko, 1992) of the network. The mean square error is propagated through the network during the learning stage, when the neural network is presented with a set of examples (training dataset).

The learning approach implemented by CNM is based on the concept of rewards and punishments, analogous to that of the Backpropagation model, to identify successful and unsuccessful pathways. Synapses defined in CNM topology have weights and a pair of accumulators for rewards (R_{ACC}) and punishments (P_{ACC}) (Machado and Rocha, 1992). To begin with, all weights are set to one and all accumulators to zero. During the training phase, as each example in the training set is propagated along the network, pathways that lead to correct classifications have their reward accumulators incremented. Similarly, misclassifications increment the punishment accumulators. Weights remain unchanged during the training process. At the end of training phase, which is done with a single scan over the training set, pathways with more punishments than rewards, e.g., $R_{ACC} < P_{ACC}$ are pruned, and the remaining connections have their weights calculated.

The CNM learning method is performed through the Incremental Reward and Punishment (IRP) algorithm, and the Starter Reward and Punishment (SRP) algorithm. The SRP algorithm is used to initialise the network and calculate the punishment and reward accumulators according to the training set. The accumulators are set to zero and the weights are set to one. All data examples are applied to the network in a single scan, to update the rewards and punishment accumulators. After scanning the training set and calculating the accumulator values, no-rewarded pathways are pruned and the remaining pathways have their weights updated.

The IRP algorithm adjusts the knowledge (accumulator and weight values) of the network. The IRP algorithm updates the punishment and reward accumulators, considering the values

previously calculated by the SRP algorithm, removing all negative or weak connections. Weak connections are those with weights smaller than a predefined pruning threshold that is empirically determined by the user. Furthermore, the IRP algorithm identifies and keeps all pathognomonic pathways (those with no punishment and a positive reward value), setting their respective weights with values higher than the pruning threshold. After the learning stage, the final CNM neural network keeps only the pathognomonic pathways.

Beckenkamp (2002) enumerates several characteristics of the CNM NN model. Some of the characteristics that are particularly relevant to this research and its application in aviation weather forecasting are:

- Input neurons only bypass the information, which has to be normalized in the range [0,1]. This value is normally assigned by a domain expert or knowledge engineer during a data preparation stage. It is called the relevance degree of an evidence/value pair.
- Input and output neurons are defined by a domain expert or knowledge engineer. Combinatorial neurons are automatically generated, based on the possible combinations among input neurons. As a result, CNM can easily fall in a combinatorial explosion situation. The CNM implementation in CANN provides optimisation of the original CNM algorithm in order to minimize this problem.
- CNM synapses are provided with reward and punishment accumulators that are used to decide whether or not to prune the synapses, and also to update synaptic weights.

A more detailed explanation of both the IRP and the SRP algorithms can be found in (Denis and Machado, 1991; Machado, Barbosa and Neves, 1998), and (Beckenkamp, 2002). The CNM has been successfully employed in several experiments dealing with classification problems, and these experiments are reported in the literature (Machado and Rocha, 1992; Leao and Reategui, 1993a; Reategui, Campbell and Borghetti, 1995), and (Viademonte, Leao and Hoppen, 1995). The successful applications of the CNM model in practice constitute one of the main reasons for choosing the CNM as the NN model in this research.

Chapter 6 describes the CANN and CNM implementation in aviation weather forecasting.

4.7 The Functionality of the DM-NN Model

The purpose of the proposed DM-NN model is to support decision making by recalling past facts and decisions, inducing "chunks" of domain knowledge from this information and performing reasoning upon this knowledge in order to verify hypotheses and reach conclusions in a given situation. The DM-NN model for IDSS is task-oriented. This is a typical feature of an intelligent decision support, as the knowledge it contains should be specific to the problem at hand. However, the general principles behind the proposed system can be applied to any domain where large volumes of historical data are available.

The DM-NN model for decision support can be seen from two perspectives: as an iterative and interactive decision support *process* and a *computational architecture*. Firstly, it defines a decision process, and at the same time it provides a computational architecture for linking various technological components in a single decision support cycle (see Figure 4.13).

From the process perspective it proposes a line of actions that can be taken to support a particular decision situation. In that sense it is a normative process, as one activity relies on the previous activity linked by some algorithmic relationship. Despite this normative aspect, it is not necessary for the decision maker to follow all the proposed steps until the final recommendation is reached. If a decision maker is satisfied with intermediary results, the process can be stopped at that level.

At the same time, the proposed model for decision support involves a computational architecture, as it suggests the combined use of different computational components and technologies. As a result, it defines an interactive computational environment that uses data mining technology to automatically induce domain knowledge from case bases, and a NN based system as a core for an advisory system, which provides the user interface.

The decision support model comprises a decision-oriented data repository (data warehouse or databases), case bases and knowledge rule bases. The data repository contains historical raw data from the problem domain under consideration. Case bases contain selected cases from the specific problem at hand, which are built from the historical raw data stored in databases. Knowledge rule bases are built based on the data mining results; they contain structured generalized knowledge that corresponds to relevant relations discovered (mined) in case bases.

Figure 4.13 illustrates the decision support cycle of the DM-NN model.

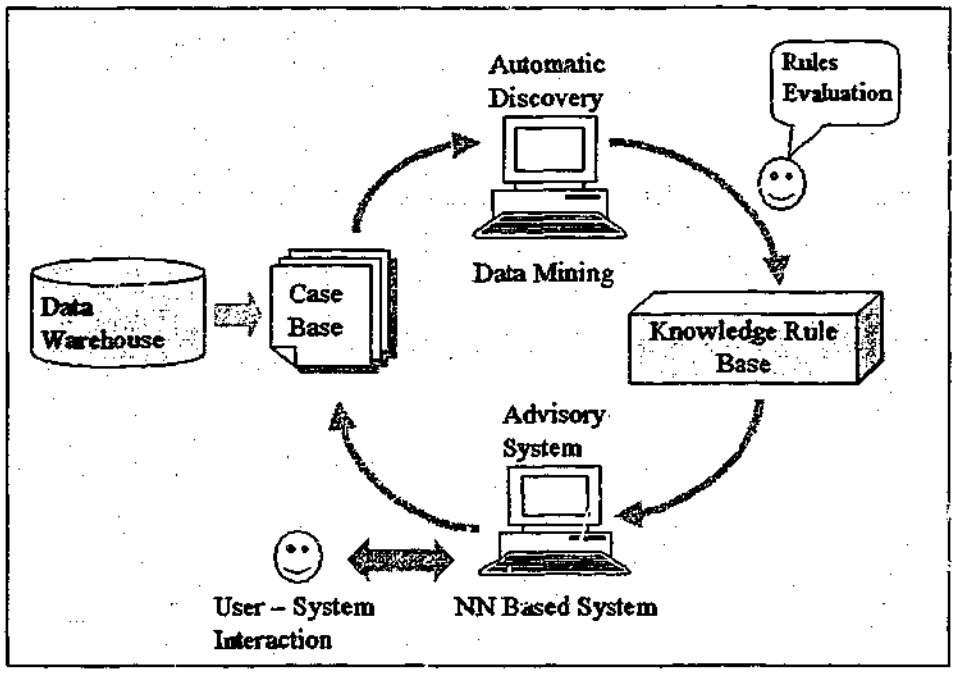

Figure 4.13: The decision support cycle of the DM-NN model

The system provides three levels of decision support: *rules generation*, *case consult* and *case base consult*. The rule generation corresponds to the set of association rules generated in a data mining session, which may be evaluated by the decision maker. If the generated set of association rules provides enough information to the user decision maker to arrive at a decision, the situation is resolved and the generated rules can be stored in the knowledge base for further use. At this point the process can be considered finished.

Otherwise, the rules can be presented to the NN for learning. After the NN-based learning procedure has been executed, the advisory component provides a case consult and a case base consult facility through its consult mode for the user to test and validate hypotheses about the current decision situation (see section 4.3.5, "Intelligent Advisory System.")

A case consult presents a selection of evidences, and their respective evaluation by the IAS component. The IAS component evaluates the selected evidences and calculates a confidence degree for each hypothesis. The inference mechanism implemented by the CNM neural model indicates the hypothesis with the highest confidence degree as the candidate solution (or class) to the problem. A case base consult is similar to a case consult, except that instead of presenting a single case (or one set of evidences) each time, several cases are presented to the

IAS component for evaluation. The IAS component evaluates a set of cases in the same way as a single case.

Furthermore, if the user decision maker believes that the outcome of the IAS (after a consult has been performed) represents novel and potentially useful information; it can be stored, either in the case base as a new case, or in the knowledge base as a new rule. Normally, it is expected that IAS outcomes will be stored in the knowledge base, but this decision depends on the specific characteristics of each application, and what the user believes is more appropriate.

The proposed architecture aims to serve as a model for a KDD-based intelligent decision support that can be used more widely in decision contexts, in data rich domains.

4.7.1 Applying the DM-NN Model

In order to verify the DM-NN model applicability and assess its performance it was implemented in the context of aviation weather forecasting, to identify severe and rare weather phenomena at airport terminals, particularly fog phenomena. Severe and rare weather events are intrinsically problematic to predict because weather forecasters normally do not have extensive experience in forecasting such events; as such, false alarms or incorrect forecasts are more likely to occur. Furthermore, severe weather events are hazardous events that can potentially cause serious damage and unsafe conditions.

Chapter 5 and Chapter 6 describe this implementation in the context of the decision support cycle illustrated in Figure 4.13. Chapter 5 describes the knowledge discovery stage of this implementation, where descriptive models are built (refer to section 4.2), and Chapter 6 describes the intelligent advisory stage where predictive models are built.

Specifically, Chapter 5 describes the process of building cases from a meteorological data repository (Australian Data Archive for Meteorology, ADAM), including the activities of data preprocessing for data mining to populate the case bases with built cases. Chapter 5 also describes the data mining process, and the process of building knowledge bases.

Chapter 6 describes the intelligent advisory system stage, where the NN based system is employed. The stages of domain modelling in the CANN simulator, NN training and consultation are described.

Figure 4.14 illustrates this approach. It depicts the decision support cycle implemented through the DM-NN model and how that cycle is developed through the DM-NN model implementation in aviation weather forecasting.

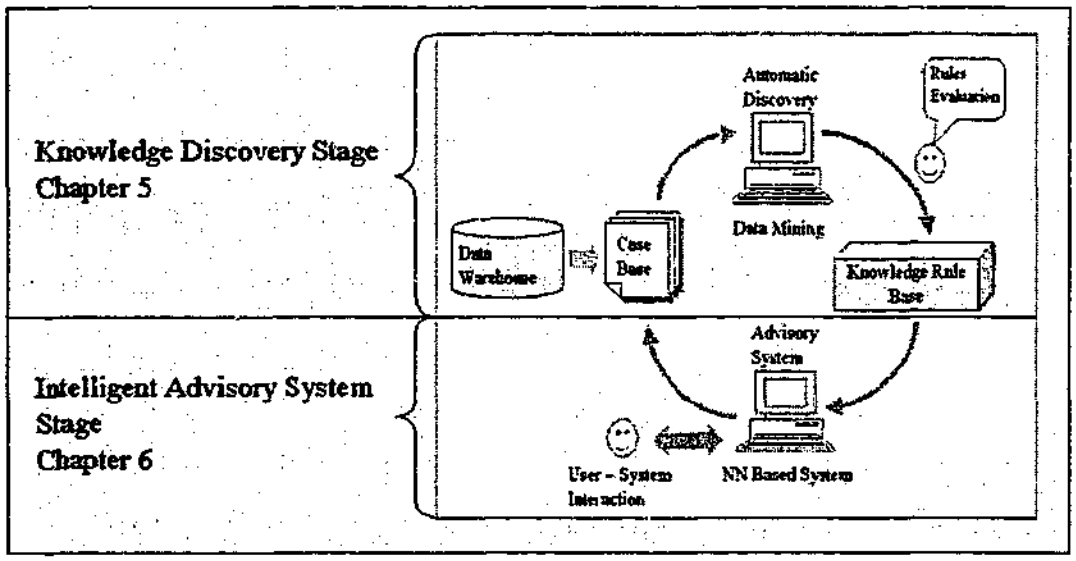

Figure 4.14: The decision support cycle in the context of aviation weather forecasting

A quantitative approach was used to assess the DM-NN model performance, where the holdout method was employed using a database provided by the Australian Bureau of Meteorology (BOM), with 49901 weather observations from July 1970 until June 2000, taken at Tullamarine.

Following, Chapter 5 and Chapter 6 describe the implementation of the DM-NN model in aviation weather forecasting, and Chapter 7 discusses the performance of the DM-NN model in the context of this implementation.

4.8 Chapter Summary

This chapter introduced the proposed hybrid DM-NN model for intelligent decision support system. It first gave an overview of the operation of the DM-NN model, and then described the proposed framework architecture, its components, their interactions and functionalities. The employed knowledge representation schema was explained, the neural network environment CANN simulator and CNM neural network model were introduced, and the specific category of problems the DM-NN model was designed to handle, as well as its operational mode were explained.

Chapter 5

5 Applying the DM-NN Model in Aviation Weather Forecasting: The Knowledge Discovery Stage

This chapter describes the stage of knowledge discovery in the implementation of the Hybrid DM-NN model for IDSS in the context of aviation weather forecasting at Tullamarine.

First, the problem of aviation weather forecasting is introduced. Next, the stages of discovering knowledge from meteorological databases and building knowledge bases are described.

5.1 Introduction

The implementation of the DM-NN model in aviation weather forecasting forms the second major theme of the framework for the IDSS model proposed in this research. Chapter 4 introduced the framework technological components and their respective roles and interactions. Chapter 5, Chapter 6 and Chapter 7 are dedicated to the implementation of the IDSS model in aviation weather forecasting. Although these three chapters cover distinct stages of that development, together they form the whole theme of that application.

This chapter covers the stage of knowledge discovery developed in this research that relates to the activities of data gathering and preprocessing, data selection, modelling, sampling and data mining.

Figure 5.1 illustrates the stage of knowledge discovery in the perspective of the decision support cycle implemented through the DM-NN model, previously discussed in Chapter 4. The activities depicted in Figure 5.1 were developed and discussed thoroughly this chapter in the context of aviation weather forecasting.

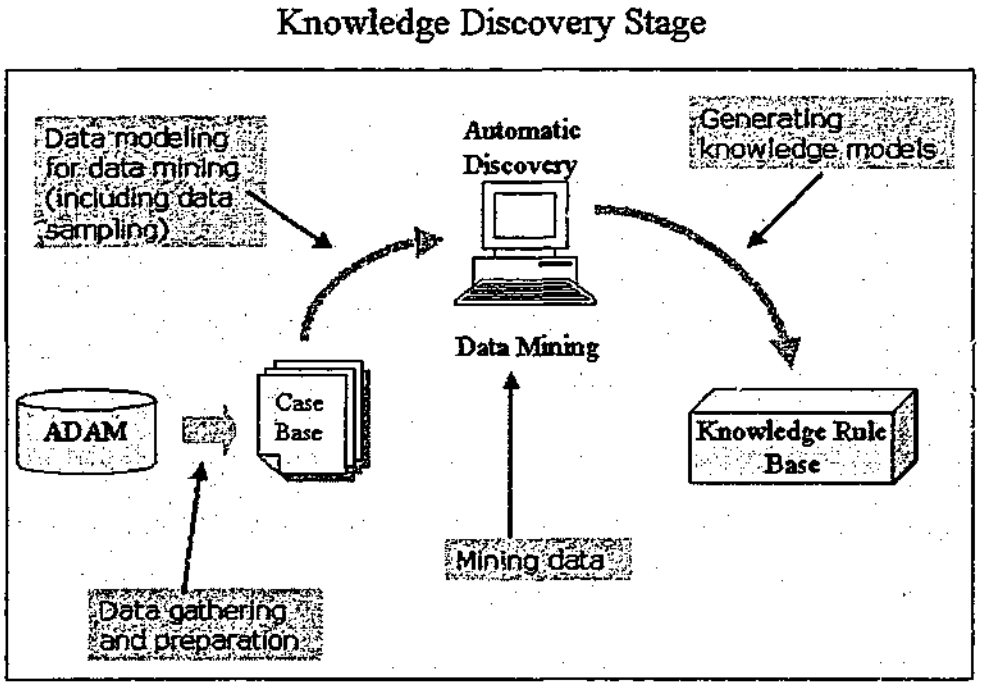

Figure 5.1: The knowledge discovery stage

Section 5.2 introduces the issues of aviation weather forecasting, definitions and concepts.

Section 5.3 addresses the issues of data gathering and preparation. It includes the activities of features selection and construction, and also data analysis. The activities of data modelling for data mining are also described, including data sampling and building datasets for data mining.

Section 5.4 addresses the activities of data mining, including selecting database attributes, discretization of numerical attributes (features), choosing data mining threshold values and finally generating the knowledge models.

5.2 Issues in Aviation Weather Forecasting

The proposed DM-NN model for IDSS was applied in aviation weather forecasting, in particular, identifying fog phenomenon at airport terminals. It allowed a validation of the DM-NN model and an assessment of its performance. This application was developed at Tullamarine.

The Australian Bureau of Meteorology (Meteorology, 2003a), Victorian Regional Forecasting Centre (VRFC) is the organization responsible for providing weather forecast reports for metropolitan Melbourne and regional Victoria. These reports include public weather forecasts, nautical information and aviation weather forecasts.

The design, development and application of the DM-NN model in aviation weather forecasting were done in the context of the Mandala Project (Linger et al., 2000). The Mandala Project is the base for the Bureau's Forecast Streamling and Enhancement Project (FSEP), which concerns the development of relevant and integrated support systems for operational forecasters. The FSEP project aims to provide on-line services that include access to a wide variety of data, data visualisation and graphical editors, historical forecast databases and automated forecast guidance, amongst other services (Meteorology, 2003a; Linger et al., 2000). It includes testing of intelligent alerting techniques based on 'agent' technology, and the implementation of a robust objective guidance technique at some 600 sites across Australia, combining direct model output and statistical predictions of weather variables (Meteorology, 2003a).

Weather forecasts are based on a collection of weather observations describing the current state of the atmosphere, such as precipitation levels, information on the ground about air and wind-related measures, such as direction and velocity, temperature, dew point depression, etc (Meteorology, 2003b). As described by the Bureau of Meteorology (Meteorology, 2003a) these data come from a wide array of sources and include:

- weather reports from human observers, automatic weather stations, ships at sea and buoys
- measurements of the upper atmosphere from instrument packs carried aloft by weather balloons and from aircraft reports
- data received from weather satellites, including satellite images and vertical temperature cross sections of the atmosphere, detected from satellite based sounders
- data from other sources such as weather radar and surface based profilers

There are two types of weather phenomena that are particular relevant for aviation weather forecast, *rare* and *severe* events. Rare events are intrinsically problematic to predict because forecasters normally do not have extensive experience in forecasting such events; as such, false alarms or incorrect forecasts are more likely to occur. Severe weather events are hazardous

events that potentially can cause serious damage and unsafe conditions. Unsafe conditions are named *sub-minimal* conditions.

In the case of aviation weather forecast, severe weather conditions potentially prevent aircraft from landing and taking off. For example, if the visibility at the airport is below a particular critical value, aircraft may not be allowed to take off and land.

Some of the most hazardous weather events are severe thunderstorm activity, cyclones, low cloud and fog. In particular, dense fog and low cloud represent some of the greatest hazards to aviation and to nearly all forms of surface transport. Aircraft are generally not allowed to take off or land if the visibility is less than a critical minimum value.

5.2.1 Weather Forecast at Tullamarine Airport

Two different types of aviation weather forecasts are made for Tullamarine: TAF (Terminal Aerodrome Forecast) and TTF (Trend Type Forecast). Routine TAFs are issued by the Victorian Regional Forecasting Centre (VRFC) every 6 hours for a forecast period of 24 hours. TAFs are the primary information that is supplied to the airlines by the VRFC. TAF provides hour-by-hour forecasts of weather conditions critical to aviation operations such as cloud amount and height, visibility, turbulence, precipitation, wind speed and direction, temperature and pressure (Meteorology, 2003a).

Every half an hour a TTF is appended to an observation of the current weather, which is called the Metar. The forecast period of the TTF is 3 hours and for this period it overrides TAF. These forecasts are amendable at any time. Special observations (named Speci) are issued whenever horizontal visibility, cloud base or other significant weather varies across certain minimum values critical to aircraft operations (Keith, 1991). Measurements below these minimum values are known as *sub-minimal* conditions. Sub-minimal conditions are particular significant because if they are predicted by forecasters for a certain period, the Civil Aviation Authority (CAV) will assign a requirement that any aircraft allocated to arrive at the airport during that period must carry extra fuel. This is to enable the aircraft to keep a holding pattern over the airport, or to diverge to an alternative airport. Not properly forecasting sub-minimal conditions can potentially cause a situation in which an aircraft does not have enough fuel to travel to an alternate airport, as such, having to proceed with a risky landing operation.

On the other hand, carrying more fuel than necessary has a significant economic impact on the airlines. Most sub-minimum weather conditions at Tullamarine are caused by low cloud and fog (Keith, 1991).

TAFs are verified by the Aviation and Defence Office in the Services Policy Branch of the Bureau of Meteorology's Head Office in Melbourne. Most planning of fuel requirements to aircraft of international flights is done using TAF, and for national flights is done using TTF (this is because most flights coming into Tullamarine arrive via Adelaide, Sydney, Hobart, Canberra and provincial centres. All of these are less than 2 hours flying time from Melbourne).

According to Keith (1991) forecasts for Tullamarine demonstrate poor performance, particularly of short-term forecasts of *low stratus* and *fog*. Short-term forecast is characterized by forecasting an event from a progressively smaller lead of time.

Attempts to formulate objective forecasting schemes using conventional synoptic data have not significantly improved very short-term forecasting. Keith (1991) demonstrated that forecast skill decreases sharply when lead times become less than the interval between synoptic observations. Love (1985) comments about the difficulty of very short-term forecasts based on a synoptic network. According to Keith (1991) short term airport forecasts are based on a variety of information, the most relevant of which are *synoptic observations* and *aircraft reports within about 150 km of the airport*, *satellite pictures* and *radar*.

In the case of Tullamarine, where the vast majority of sub-minimal weather conditions involve low cloud and fog, radar is of value for only a relatively small percentage of sub-minimum conditions. Most of the poor weather at Tullamarine occurs overnight, thus assistance from satellite pictures is limited to 3-hourly infrared images. Keith (1991) observed that despite several attempts using infrared satellite pictures for detection of low cloud and fog, these have not proved helpful for the forecast problem at Tullamarine. For instance, Bond (1981) aimed to detect moisture fields, which are a precursor of fog formation, using infrared images; and Bedson and Canterford (1983) formulated a low cloud enhancement for infrared satellite pictures. According to Keith (1991) the problem with using satellite pictures for detection of *fog* and *stratus* is related to temperature. Fog and low stratus are invisible on

these pictures due to the small differences between the ground and cloud-top temperature, which are almost the same in most cases. In addition, the low cloud and fog does not simply advect over the airport but forms over a broader area, so the satellite pictures do not identify low cloud and fog in small areas, such as the airport (Keith, 1991; Auer Jr., 1992).

Additionally, one of the major difficulties of fog forecasting is that it is a very short-term forecasting. As one attempts to forecast an event from a progressively smaller lead of time, the utility of guidance that is derived from synoptic-scale data progressively decreases. When it occurs in a location, such as Tullamarine, where single station data gives no measurable assistance, the forecast task become virtually impossible.

5.2.2 Fog Formation and Definitions

According to international convention (Auer Jr., 1992), fog can be defined as restricting visibility to equal or less than one kilometre, or visibility between one and two kilometres.

Fog forms when air cools without losing water, or when air is moistened (slightly or moderately wet) without warming. Fog has several characteristics: restricts visibility, it has aerosol composition, cooling and moistening of near surface air, it presents a coin effect (it comes and goes successively), and it seems that the earlier it starts, the later it clears.

Fog is classified in different types, according to the origin of its formation. The types of fog are: *rain*, *radiation*, *freezing*, *post-front* and *hill fog*.

It is necessary for the terminal forecast to indicate that fog is expected if the probability of fog occurrence is higher than or equal to 50% *(Fog (P) >= 50%)*. If the probability is higher than or equal to 30% but less than 50% *(Fog (P) >=30% and Fog (P) <50%)*, this needs to be indicated in the terminal forecast.

If *Fog (P)>=1% and Fog (P) <30%*, this needs to be indicated in a special advice, referred to as "Code Grey." The Code Grey advice stipulates a small but reasonable probability of a sub-minimal weather condition occurrence.

There are two categories of Code Grey advice relating to fog phenomena occurrence: when the probability of fog occurrence is higher than or equal to 15% and less than 30% (*Fog (P)>=15% and Fog (P) <30%*), or when the probability of fog occurrence is higher than or equal to 1% and less than 15% (*Fog (P)>=1% and Fog (P) <15%*)

5.2.3 The Intelligent Decision Support for Aviation Weather Forecasting

The hybrid DM-NN model for IDSS was applied to identify the occurrence of fog phenomena at Tullamarine, supporting the task of fog forecasting in small intervals of time (short-term forecast). The DM-NN model is expected to issue a confidence measure related to fog occurrence, taking into account the weather pattern at the airport at a particular time.

Fog phenomenon was selected in the application developed in this research because it is a significant event for aviation weather forecast, and it is difficult to forecast. There is also available historical data that makes it suitable for the concepts developed in the DM-NN model.

The work of a forecaster is complex, and is categorised by uncertainty, incomplete information, multiple sources and a great variety of information and strict timelines all overlaid by a legal regime. Forecasters are required to exercise judgement because science is often inadequate at the level of detail required by specific forecasts (Linger et al., 2000). Much of the skill of a forecaster is dependent on their experience in terms of the type of forecast and the locality of the forecast. In such an environment, work activity assumes not only task performance (construction of a forecast), but also the review and re-assessment of the work done in order to understand and learn from the experience. Such work requires an enhanced decision support approach, which is called the *Intelligent Decision Support* (IDS) (Linger and Burstein, 1997). In such IDS approach, the forecaster is engaged in a cognitive process of problem exploration, with the system providing necessary intelligent assistance to cope with uncertainty (Burstein et al., 1998). This type of decision support requires the system to have an extended functionality, including reasoning, memory aids, explanation facilities and learning capability (Linger and Burstein, 1997). These requirements are in accordance with the capabilities of intelligent systems as discussed in Chapter 2.

Local area forecasting (such as airport forecasting) relies on extensive knowledge of the local weather patterns, the geography of the region and the development of weather at that location in the context of broad meteorological features. Airport terminal forecasting is further complicated by the time scale, which can be up to 18 hours, compared to 3 hours for general forecasts. Rare event forecasts, such as fog forecast, are intrinsically problematic because, by

definition, forecasters do not have extensive experience forecasting such events. Yet it is these events that have the most significant impact on airport operations and financial (and legal) implications for airlines. This is an area where forecasters need to share past experiences and, importantly, to learn from those experiences (Linger and Burstein, 1997).

Consequently, access to past decision situations, and knowledge derived from them, can provide a valuable source of improvement in forecasting rare weather events. The complexity and diversity of weather observations and the large variation in the patterns of weather phenomena implies serious problems for forecasters trying to devise correlation models. Consequently, the area is a potential candidate for *intelligent systems* purpose (Viademonte et al., 2001b).

Additionally, as previously discussed in section 5.2.1, data available for predicting weather phenomena is usually highly diverse, and comes in large volumes and from different sources and formats. It must be organized into a comprehensible form before it is useful for predicting the evolution or prognosis of future weather patterns. As such, there is an overload of available data that makes *KDD* and *data mining* technologies potentially useful in this domain.

5.3 Discovering Knowledge from Meteorological Databases

This research applied data mining and performed the activities of data selection and modelling, cleaning, pre-processing and sampling, this stage was named *knowledge discovery stage.*

Data mining was applied to generate sets of rules from sets of cases stored in case bases, as such case bases constitute the sets of minable data. Generated rules are stored in knowledge rule bases, and are further processed by the neural network component. As such, the knowledge discovery stage comprises two main phases: case bases generation and knowledge bases generation.

The case bases generation includes all data preparation activities done since the application domain was identified, until a comprehensible set of minable data was achieved.

As previously discussed in section 5.2, the DM-NN model was implemented to identify fog occurrence at Tullamarine. A database with weather observations was categorized into two groups (classes), one representing a fog occurrence, identified as fog cases (or fog class), and one representing a non fog occurrence, identified as not fog cases (or not fog class).

The first step in knowledge discovery is to identify and understand the application domain, gather and prepare data. This normally includes the activities of data cleaning, dimensionality reduction, features selection and transformation, and data sampling.

In this research, after data preparation activities were performed, the resulting dataset was split in two subsets of data: one dataset with fog cases only and the other with not fog cases only. This division was performed for data analysis and sampling purposes. A random sampling approach was used to generate the datasets for mining and testing. The mining sets were used as input data into a data mining algorithm, and the testing sets were used to assess the performance accuracy of the DM-NN model.

Figure 5.2 illustrates the adopted schema for generating *case bases.*

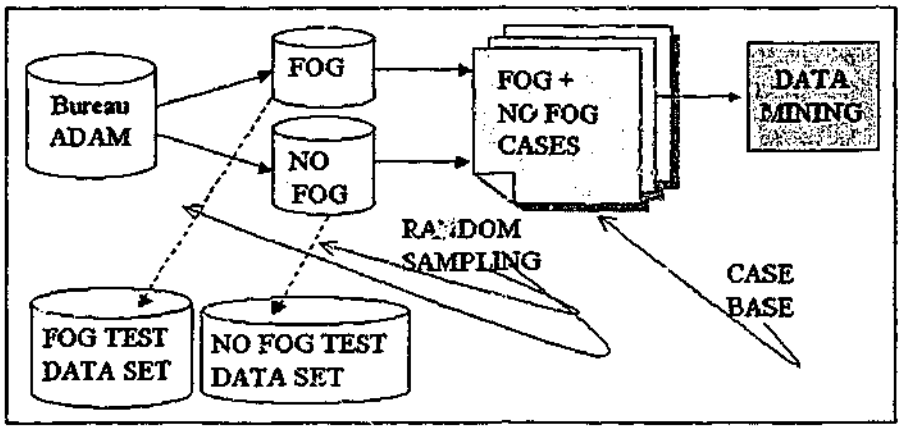

Figure 5.2: Schema for generating case bases

Different subsets of data were generated for data mining, according to different sampling proportions, and several rules describing fog and not fog were obtained as a result of the data mining processing. The criterion adopted in this research to select interesting (or representative) rules was based on data mining thresholds: setting different levels of rule confidence degree, support degree and rule order. This is an interactive process, as normally several data mining sessions are required to reach a satisfactory final set of rules. This interactive process implies selecting different features (weather observations in the case of this

research), modifications in the attribute discretization ranges and trying diverse settings of data mining thresholds. As a result of this process, several sets of association rules were obtained in the experiment conducted in this research.

The outcome of the data mining process are sets of association rules, as described in section 4.4.2 "Representing the Domain Knowledge," which populated the knowledge bases.

Figure 5.3 shows the process adopted to generate *knowledge bases*:

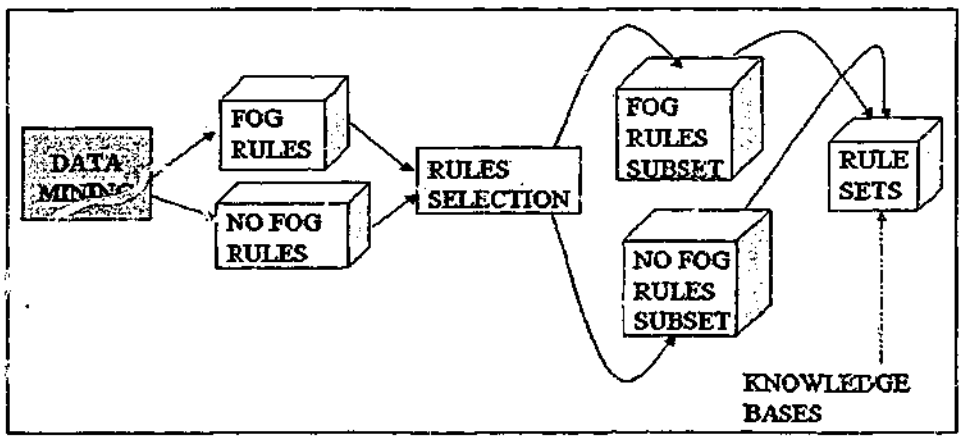

Figure 5.3: Schema for generating knowledge bases

The knowledge bases were used as the input data, e.g. training datasets, to the hybrid neural network based system, called the IAS component in the DM-NN model (refer to section 4.3.5, "Intelligent Advisory System.")

After the training process is finalized, the DM-NN is ready to be used as an advisory system within the application domain it was applied. The predictive performance of the DM-NN model in aviation weather forecasting was assessed through the testing datasets. The holdout method for estimation of accuracy (Weiss and Indurkhya, 1998) was applied for this. Chapter 7 describes the performance assessment process developed in this research.

The activities of data preparation and data modelling are discussed next in this chapter, including the stages of data cleaning, features selection and transformation, and data sampling. The mining sets, training and testings sets obtained are also described in this chapter, as well as the process developed in generating these sets of data.

5.3.1 The Domain of Aviation Weather Forecasting

The database for the study of aviation weather forecasting was provided by the Bureau, and obtained from ADAM (Australian Data Archive for Meteorology) data repository. It is a database that stores daily weather observations, and is built and maintained by the Bureau.

Three sets of data were provided by the Bureau. The first dataset had 75 attributes (features) and 2917 records of weather observations. Many of the attributes in this dataset were related to data quality control and codes describing weather observations. For example, the dew point weather observation had two associated attributes, *DWPT* and *DWPT_QUAL*. This last attribute indicates data quality information, which was not relevant for the data mining purposes of this research. Many other observations (attributes in the database), like *wind speed*, *wind direction* and *visibility* showed the same problem. The attribute identifying fog occurrence was described through a Boolean type attribute (*FOG_OCC*) together with an attribute describing the visibility observation, represented by codes.

This led to a situation that identified fog types instead of identifying fog occurrence; and this research was concerned with identifying fog occurrence regardless its type. Clearly, this situation was not desirable. The preliminary experiments showed that the problem of fog forecasting would not be properly addressed without intensive data preparation work.

A second dataset was provided by the Bureau, with 47461 records and 17 attributes representing weather observations, from July 1970 until June 1999. Later, a third dataset was provided with weather observations from 1999 until 2000. This third dataset had 2440 records, and the same 17 attributes.

The final database had weather observations from July 1970 until June 2000, with 49901 records (rows of data), 17 attributes, and a size of nearly 250 MB.

The attributes represent the weather observations used when issuing an aviation weather prognosis bulletin, and were selected and provided by forecasters from the Bureau.

The attribute definitions and relevance for fog forecasting were obtained during meetings with forecasters from the Bureau, from bibliographic references and specific documents provided by the Bureau, among them: (Auer Jr., 1992; Bedson and Canterford, 1983; Bond, 1981; Colquhoun, 1987; Keith, 1991; Love, 1985); including online information maintained by the Bureau (Meteorology, 2003a; 2003b).

Table 5.1 lists the weather observations attributes with their respective descriptions:

Table 5.1: Weather observations

Attributes	Description
1 Year	Year of the weather observation
2 Month	Month of the weather observation
3 Day	Day of the weather observation
4 Hour	Period of 3 hours weather observation
5 Dry-bulb temperature	Temperature in Celsius degrees
6 Dew point temperature	Temperature in Celsius degrees
7 Total cloud amount	Height
8 Total low cloud amount	Height
9 Mean sea level pressure	Pressure measured in HpA
10 Past weather	Weather code, weather since last observation
11 Rainfall	Precipitation measured in millimetres (mm)
12 Present weather	Weather code, weather when the observation was done
13 Visibility	Kilometres, at the airport area
14 Wind direction	Degrees, direction from where the wind is coming
15 Wind speed	Meters/second
16 Fog type	F = "Fog"; LF = "Local Fog"
17 Fog occurrence	FD = "Fog Day"; LFD = "Local Fog Day"

5.3.2 Understanding Meteorological Data

A series of meetings with weather forecasters were conducted in order to understand the meaning and relevance of meteorological data. As a result of these meetings and the studied literature, meteorological data was conceptualised according to the context of aviation weather forecasting. This section explains the meaning of the selected meteorological data in that context.

Seasonal factor

The attributes *Month*, *Year*, *Day* are date stamps and monotonic variables. The seasonal factor was modelled monthly; as such, the year and day attributes were not necessary for data mining purposes. However they were kept in the dataset, as they are necessary to calculate the *Previous afternoon dew point* attribute, which is a new calculated field (which will be discussed further in this section).

Hour observation

The hour observation refers to a three hour period observation. The weather observations are recorded by the Bureau at a granularity of three hours. This granularity must be considered when making a prognostic.

Fog identification

This research addresses the problem of fog forecasting; consequently fog is the event under consideration. There were two attributes in the original database that together identified a specific occurrence of fog, *Fog type* and *Fog occurrence*. When *Fog type* has no value it means no fog phenomenon occurrence at that time. *Fog occurrence* indicates whether it is a Fog Day (FD), or a Local Fog Day (LFD), or no fog day at all. The distinction between local fog and fog day does not matter in terms of aviation weather forecasting; it does matter in terms of fog phenomena classification. Therefore, the attribute *Fog occurrence* is not necessary for the purposes of this research, as this research focus on fog phenomenon occurrence, regardless of its classification. *Fog occurrence* was removed from the database.

Temperatures

The *Dew point* and *Dry bulb* are the local temperatures at the airport in degrees Celsius, taken at the time of the observation.

Wind

This includes information about wind *direction* and *speed*. Both are taken on the surface at the airport, around 200 feet altitude. The wind direction is expressed in degrees of compass points, from 0° to 360°, and the speed in meters/second (m/s).

Weather codes (past and present weather)

Weather codes are provided by the Bureau, and a particular weather phenomena (or pattern) has a code assigned to it. For example, code number '41' relates to fog in patches at the time of the observation, with visibility less than 1000 metres, but with the sky visible. Code number '40' identifies occurrence of fog at a distance but no fog at the station during the past hour, with visibility greater than 1000 metre. Code number '99' identifies heavy thunderstorms with hail, at the time of the observation.

The *Past weather* attribute (code) relates to what was the weather pattern since the last 3 hours observation, and the *Present weather* relates to the weather pattern at the time of the observation.

All other weather observations, such as the *Amount of clouds*, *Sea level pressure*, *Rainfall* and *Visibility* are taken locally at the airport.

The inclusion of new information not presented in ADAM (at the time this research was being developed) was suggested by the forecasters. Among them, the *Previous dew point* observation was suggested as potential useful information for fog prognosis. The *Previous dew point* can be calculated through the *Date*, *Hour* and *Dew point* attributes. The *Previous dew point* was inserted in the database as a new calculated field; it is discussed further in this chapter.

In addition, *dew point depression, synoptic types*, and *geostrophic wind,* which were not present in the database provided by the Bureau were considered useful for a better predictive performance, according to the forecasters. Because of time constraints, this information was not included in the research developed in this thesis. It is important to observe that, although a good predictive performance in fog identification is important for this research; the main goal is to assess and evaluate the proposed DM-NN model for decision support.

5.3.3 Data Preparation

Normally, weather observations are collected and stored without necessary care for KDD purposes. As a result, lack of consistency and quality problems are common, what makes data preparation often necessary in the meteorological domain (Buchner et al., 1998). The importance of the data preparation (or pre-processing) stage in the KDD process should not be underestimated. If the quality of the data is low or the problem is incorrectly formalized, the algorithms used for data mining will behave inefficiently or produce incorrect results.

An intensive data preparation stage, addressing issues of data quality was performed in this research, and was discussed in section 3.4, "Data Preparation."

5.3.3.1 Feature Selection and Construction

The features (attributes) in the dataset were mostly indicated by the Bureau. However, some analyses were still performed. The *Year* and *Day* attributes were not necessary for data

mining purposes. The year attribute does not specify any pattern, as a year never repeats itself. The forecasters specified that the seasonal factor should be modelled monthly; consequently the day attribute was not relevant.

The attribute *Fog occurrence* was removed because it is not necessary for the purposes of this project. As previously discussed in section 5.3.1, this attribute identifies fog days or local fog days, and such classification does not matter in terms of aviation weather forecasting.

A new attribute was inserted in the database, the *ID* attribute. It is an auto number type attribute, and the primary key attribute.

A calculated attribute was built and inserted in the database. The weather observation *Previous afternoon dew point* was calculated based on the *Year, Month, Day, Hour* and the *Dew point* attributes. This calculated attribute was recommended by the forecasters as important information for fog prognosis. For that reason, *Year* and *Day* attributes were not removed from the database despite not being used in any data mining session.

For the purpose of inserting previous afternoon dewpoint information in the database a new attribute was created, named *Previous_DewPoint*. A program module in Visual Basic was written to calculate and update the previous afternoon dewpoint attribute. Because the weather observation database was stored in MS Access, Visual Basic was the chosen language to implement the algorithm in order to keep the same environment and to avoid unnecessary data import/export procedures from MS Access and other environments. In addition, Visual Basic offers a set of object classes in its DAO (Data Access Object) object library that facilitates the manipulation of databases and tables.

The program scans the database with weather observations updating the *Previous_DewPoint* attribute. The Australian Bureau of Meteorology, Computing Services Division, informed the algorithm. The program module called the *"Function_Update_PreviousDewPoint"* implements the algorithm described in Figure 5.4.

The algorithm to calculate the previous dew point attribute is described as follows:

```
1 Starting sequentially from the beginning of the table
2 Read the current date (day/month/year) and hour attributes values
3 Calculate the previous date, related to the current date read in
step 2, i.e., the day before
4 Search for the previous date and the same hour read in step 2
5 If the search is successful go to 6, otherwise go to 7
6 Read the Dew Point attribute value
7 Move back to the record position with the current date, indicated in
step 2
8 Update the Previous Dew Point attribute value with the Dew Point
attribute value read in step 6, or leave it blank otherwise
9 Move to the next record in the database
10 Repeat steps 2 to 10 until reaching the end of the file
```

Figure 5.4: Algorithm to calculate the previous afternoon dew point

The generation of a new attribute identified as *Dew point depression* was also suggested by the forecasters. The *Dew point depression* is calculated as the difference between *Dry bulb* and *Dew point* temperatures.

Information related to *synoptic types* and *geostrophic winds* (neither of which were in the database provided by the Bureau) were also suggested to be included. For reasons of time constraints, this information was not included in the weather observation database. The information already available was considered enough for the research purposes of this thesis, e.g. to verify the performance and feasibility of the proposed DM-NN model for decision support.

5.3.3.2 Analysis of Null Values

Handling null values is one of the most common problems in data preparation for data mining (Catlett, 1991; Fayyad, Haussler and Stolorz, 1996; Pyle, 1999). A null value in an attribute or variable might be an empty or a missing value. An empty value is a value that has no corresponding real world value, and a missing value has an underlying value that was not captured, this means the value has not been stored in the dataset but it may exist in the domain (Pyle, 1999).

Determining if any particular null value in a variable is empty, rather than missing, requires domain knowledge and is extremely difficult to be automatically detected. Missing and empty values can be removed or replaced; but this decision relies on the amount of those values, as

well as domain knowledge. Section 3.4, "Data Preparation" discussed the issue of empty and missing values.

Another common data quality problem that is closely related with the occurrence of null values is low information density, known as *sparsity*. In this case, variables are sparsely populated with instances values. Normally, sparse variables can be removed. However, there are cases when sparse values hold significant information. To remove or not a sparse variable is an arbitrary decision based on confidence levels (Pyle, 1999).

According to the analysis of null values (see Table 5.2) the *Rainfall* attribute showed a significant ratio of null instances, 30.42%. This ratio of null values may be enough to discard the attribute for data mining purposes; however, according to the experts (forecasters from the Bureau), the rainfall observation consists of significant information and needs to be kept in the database.

Table 5.2 shows the number of null values, which could be empty or missing, for each attribute in the database.

Table 5.2: Analyses about null values

Attributes	Amount of nulls values
1 Year	No nulls values
2 Month	No nulls values
3 Day	No nulls values
4 Hour	No nulls values
5 Dry bulb temperature	16 nulls (0.03 %)
6 Dew point temperature	31 nulls (0.06 %)
7 Previous dew point temperature	572 nulls (1.14 %)
8 Total cloud amount	25 nulls (0.05 %)
9 Total low cloud amount	314 nulls (0.63 %)
10 Mean sea level pressure	25 nulls (0.05 %)
11 Past weather	93 nulls (0.19 %)
12 Rainfall	15180 nulls (30.42 %)
13 Present weather	81 nulls (0.16 %)
14 Visibility	14 nulls (0.03 %)
15 Wind direction	104 nulls (0.21 %)
16 Wind speed	29 nulls (0.06 %)
17 Fog type	48963 nulls (98.12%)

The *Previous dew point* temperature has 572 (1.14 %) null instances. Doing a specific analysis with just *fog* cases, this attribute shows a smaller frequency of null values (see Table 5.3). As

this research is particularly concerned with fog forecasting the attribute was kept in the database without further modifications.

Fog type showed a high number of null values: 48963 (98.12%) instances. However, as explained by the forecasters, in this case a null value is neither missing nor empty information, but indicates that a particular weather observation is not a fog occurrence. *Fog type* was transformed to properly represent this information; section 5.3.3.4 in this chapter described the corresponding data transformation operation.

Analyses of empty and missing values were also individually made for both, *fog* and *not fog* classes, in order to verify the occurrence of empty/missing values in each class. The main reason for this division is that the number of not fog cases (48963) is significantly higher than fog cases (938). An attribute may have a small number of null values in the database, but with the majority of its null values allotted to the *fog* class, this potentially represents a problem for the purposes of this research and needs to be verified.

According to this analysis, the *Rainfall* attribute shows a significant number of null values in *fog* class, as already verified in a previous analysis with the whole population. The other attributes did not have enough null values to be considered sparse. For example, the *Total cloud amount* and *Total low cloud amount* have 1.28 % and 2.24 % of null values respectively, which is certainly not enough to consider these attributes as sparse.

Attributes without null values (either empty or missing) were not considered in this analysis. Table 5.3 shows the results for *fog* class.

Table 5.3: Analysis of sparse attributes in fog class

Atributes	Amount of nulls values
Previous dew point temperature	0.32 %
Total cloud amount	1.28 %
Total low cloud amount	2.24 %
Mean sea level pressure	0.11 %
Rainfall	21.11 %
Wind direction	0.32 %
Wind speed	0.21 %

A better understanding of the *Rainfall* attribute is necessary to take a decision concerning its null values, and this is discussed in section 5.3.3.4, "Data Transformation."

The analysis performed in the *fog* class has also been done for *not fog* class.

The same conclusions obtained in the analysis for *fog* class can be extended for *not fog* class. This means that just the *Rainfall* attribute, again, can be considered sparse with a significant amount of null values.

Table 5.4 shows the results of this analysis.

Table 5.4: Analysis of sparse attributes in not fog class

Attributes	Amount of nulls values
Dry-bulb temperature	0.03 %
Dew point temperature	0.06 %
Previous dew point temperature	1.16 %
Total cloud amount	0.03 %
Total low cloud amount	0.59 %
Mean sea level pressure	0.05 %
Past weather	0.20 %
Rainfall	30.60 %
Present weather	0.16 %
Visibility	0.03 %
Wind direction	0.21 %
Wind speed	0.06 %

5.3.3.3 Analysis of Variability

Another important issue to consider in KDD applications is the data distribution or variability. The possible patterns enfolded in a variable are distributed across a variable's range in some particular way. These patterns are fundamental to discover the relationships among variables (Pyle, 1999). The analysis of variability is important in order to verify the occurrence of *constants*, *outliers* and *monotonic* variables, as previously discussed in section 3.4.

In this research the standard deviation was used to calculate the variability of numeric attributes and frequency analysis was used for categorical attributes.

Analysis of variability did not include *Year*, *Day* and *Fog type* attributes. *Year* and *Day* attributes were not used for mining purposes, as previously discussed. The year is a monotonic variable, which never repeats itself (when a year finishes it does not happen again). *Fog type* was discretised in two possible values, *F* and *NF*, indicating a fog case or a not fog case respectively. Consequently its variability is limited to these two possible values.

The analysis of variability was primary done in the fog population. This was because the classes' distribution in the population is significantly heterogeneous, with a much higher

predominance of *not fog* class. Consequently, the variability in *fog* class was more important to capture.

As part of the variability analysis some statistics were generated through SPPS, including the number of valid and null values (identified as "missing"), the mean, and standard deviation. Table 5.5 shows the statistical measures taken from the fog population.

Table 5.5: Fog class statistics

Attributes	Valid	Missing	Mean	Standard Deviation
Dry bulb	938	0	8.3932	3.7175
Previous dewpoint	935	3	6.9829	2.9539
Dewpoint	938	0	7.1631	3.2665
Total cloud	926	12	5.1728	2.9574
Total low cloud	917	21	4.4809	3.2236
Past weather	938	0		
Sea level pressure	937	1	1023.8779	7.7426
Rainfall	740	198	8.041E-02	0.3830
Present weather	938	0		
Visibility	938	0	13.0951	12.7766
Wind direction	935	3	162.2652	153.5125
Wind speed	936	2	1.5654	1.8022

The analysis of variability in *not fog* class was done as part of the sampling procedures, and conducted in the different *not fog* samples. Additionally, as this research aims to identify fog phenomena, it is primary concerned with building descriptive models of fog class.

Histogram graphs were generated to analyse the variability of the attributes using the SPSS[1] Statistical Package. The histograms show the frequency distribution of each attribute, its standard deviation, its mean and the number of valid values (not considering missing values).

The histograms for the fog population are as follows:

[1] SPSS is a trademark of SPSS Inc.

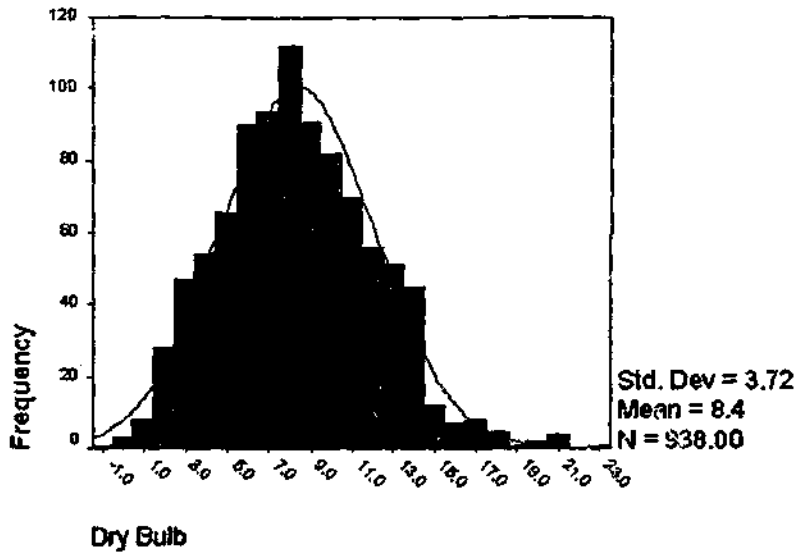

Figure 5.5: Frequency distribution for dry-bulb attribute

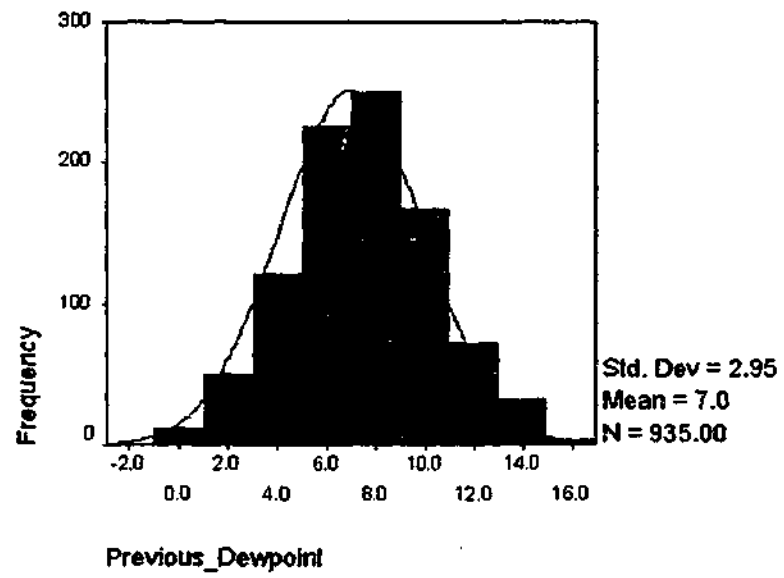

Figure 5.6: Frequency distribution for previous dewpoint attribute

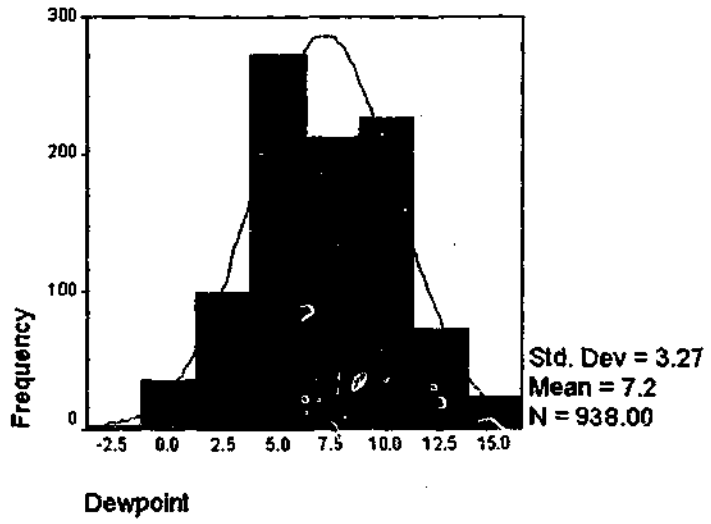

Figure 5.7: Frequency distribution for dewpoint attribute

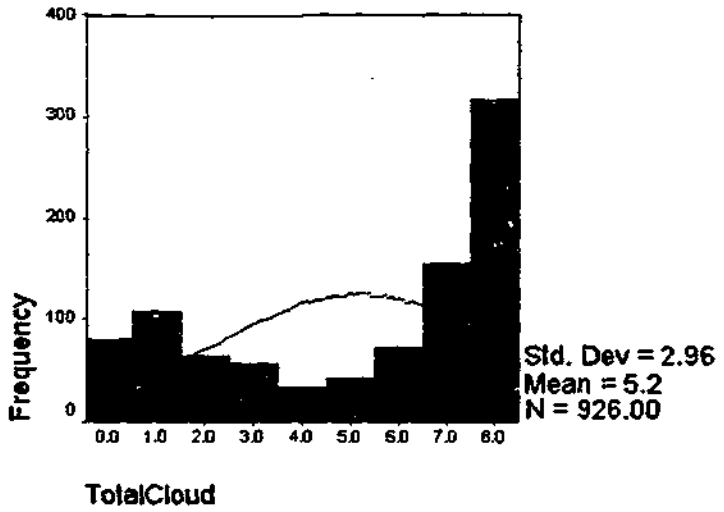

Figure 5.8: Frequency distribution for total cloud attribute

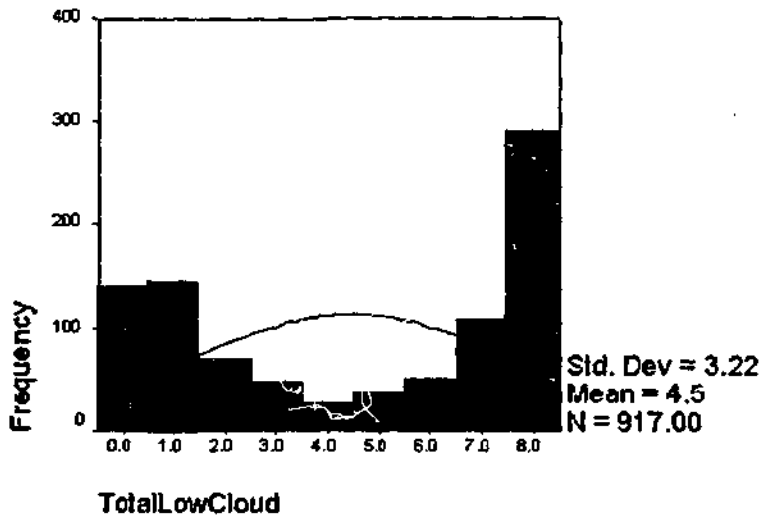

Figure 5.9: Frequency distribution for total low cloud attribute

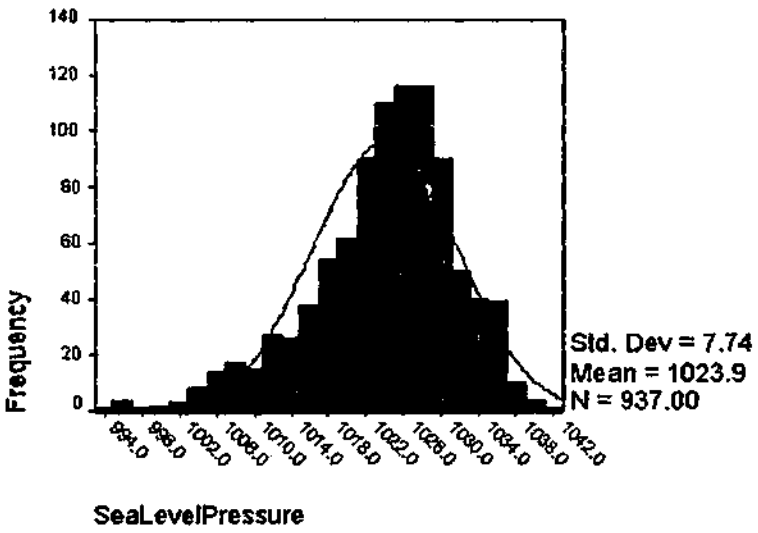

Figure 5.10: Frequency distribution for sea level pressure attribute

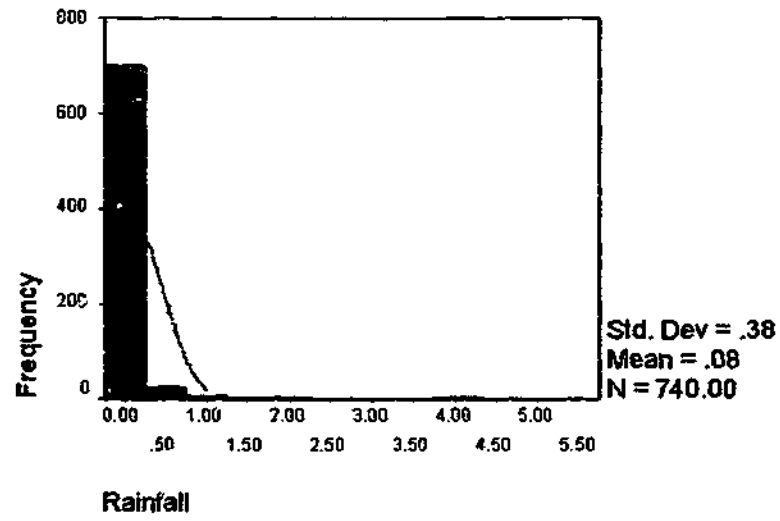

Figure 5.11: Frequency distribution for rainfall attribute

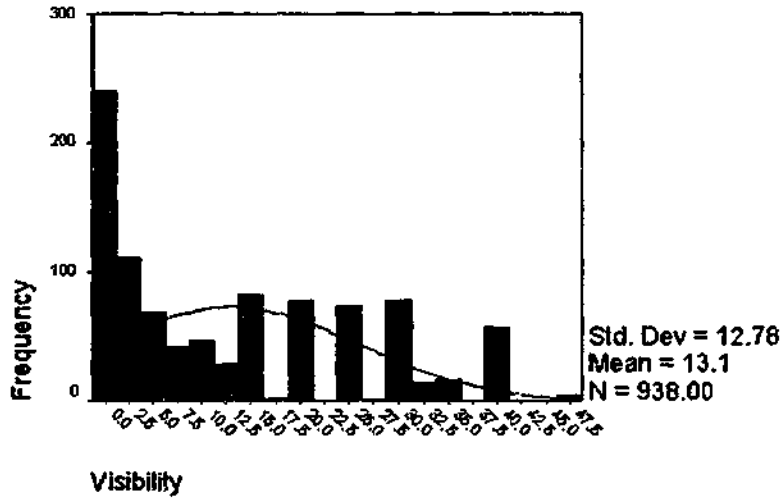

Figure 5.12: Frequency distribution for visibility attribute

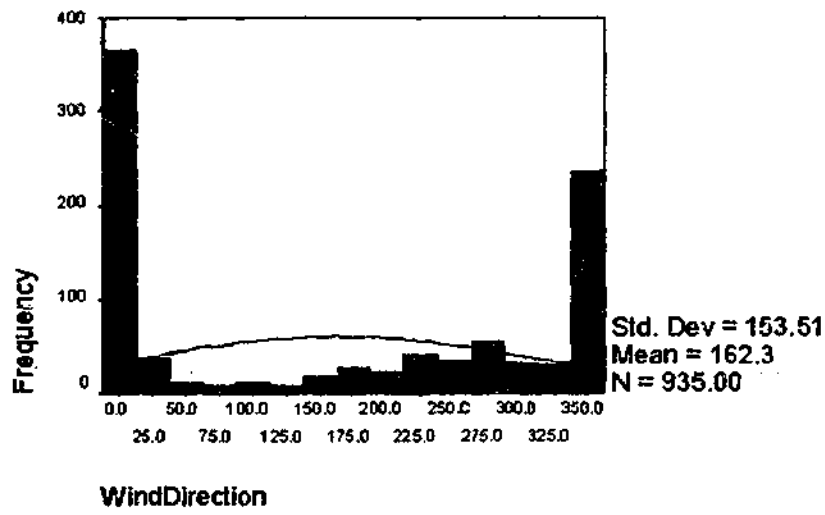

Figure 5.13: Frequency distribution for wind direction attribute

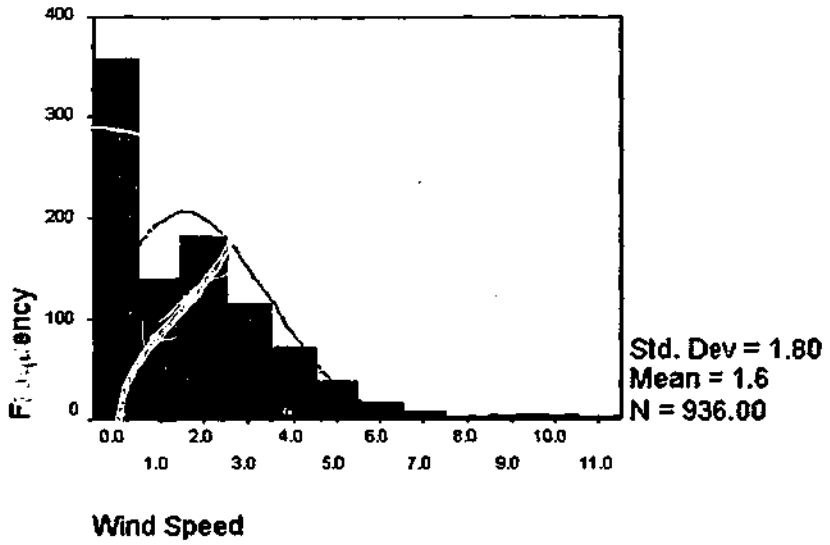

Figure 5.14: Frequency distribution for wind speed attribute

The analyses of variability revealed several problems. For instance, fog observations with high *Visibility* values, and *Wind direction* with many instance values concentrated in the 0° and 360° degree points.

The *Rainfall* attribute shows two problematic behaviours for data mining: sparsity and lack of variability. It has 198 null values, which is 21.11% of the total fog population, and 14982 null values, which is 30.60% of the total not fog population (refer to Table 5.4). For data mining purposes this number of null values potentially represents a problem. There are various approaches to handle this situation, such as removing the attribute, removing just the rows with null values, replacing the null values or transforming the attribute. The selection of any of these approaches requires domain knowledge, as the very basic question is finding out what a null value really mean in this case.

According to weather forecasters from the Bureau, a null value in the *Rainfall* attribute should be considered as zero, i.e. no rainfall at all, for forecasting purposes. According to weather forecasters, and factual knowledge in the domain, fog is unlikely to occur during rainfall. However, transforming all the null values to zero leads to the problem of lack of variability, as the null values will represent 812 instances out of 938, or 86.57% of the fog population. In this particular case the *Rainfall* attribute could be considered as a constant, and could be removed because the lack of variation in content implies no information for modelling purposes.

A second problem that was observed was fog observations with high visibility values, near or over 30 kilometres. The visibility axis in the histogram of frequency distribution for *Visibility* (Figure 5.12) shows a cluster of high values at the right end of the axis, indicating a possible case of outlier. Fog is mainly characterized by low visibility, normally less than 1 kilometre, so small values are expected. Apparently, the occurrence of fog with visibilities around 30 kilometres is a contradiction, or even an impossible situation.

Again, domain knowledge is necessary to interpret this situation and make a decision. According to weather forecasters, these particular cases of fog observations with high visibility values refer to *fog patches*, which means that there are patches of fog over the airport runaway area, but the overall visibility is mostly fine. Therefore, the high visibility values are not incorrect, and were kept in the dataset without further modifications.

The third attribute that showed a potential problem was *Wind direction*. This is specified relative to true geographic north, and is the direction from which the wind was blowing (at the time the observation was recorded). Wind direction can be specified either as the number of degrees clockwise from true geographic north, or as one of eight or sixteen compass points (Meteorology, 2003a).

Wind direction is numerically represented in the Bureau database in degrees of compass points. The histogram of frequency distribution for *Wind direction*, Figure 5.13, shows a high frequency of zero and its neighbourhood values, and also a high frequency of 360 degree values. As these values represent compass points in degrees, zero and 360 represent the same compass direction. The problem here is to understand exactly the meaning of zero degree compass points, and whether a zero degree can be considered the same information as 360 degrees.

A second problem related to wind direction observation refers to the ranges of degrees, e.g. how each range can be discretised and where the proper boundaries are between each range. This problem is addressed at Chapter 9 as issues of future research.

The analysis of variability did not show other problems related to data distribution, such as existence of outliers, monotonic variables or constants in fog population.

Following, the approaches used to handle the problems discussed in this section are described.

5.3.3.4 Data Transformation

For reasons of data integrity and compatibility, some of the attributes had their types and sizes modified, and some integrity constraints updated. The attributes *Year*, *Month*, *Day*, *Hour* are the database index and multi attribute secondary key (the *ID* attribute is the primary key). They were transformed into the same data type, "TEXT" and had their size adjusted to their possible range of values; for example, the *Day* attribute was originally defined with a size of 6 characters, and was modified to size of 2 characters, as it holds values between "01" and "31." The following modifications were performed:

- *Year* attribute was modified to Text type, size 4. Integrity constraints: required, zero length not allowed, validation rule: Is Not Null
- *Month* attribute modified to Text type, size 2. Integrity constraints: required, zero length not allowed, validation rule: Is Not Null
- *Day* attribute modified to Text type, size 2. Integrity constraints: required, zero length not allowed, validation rule: Is Not Null
- *Hour* attribute modified to Text type, size 2. Integrity constraints: required, zero length not allowed, validation rule: Is Not Null
- *Fog_Type* attribute modified to Text type, size 2
- *Fog_Occurrence* attribute was removed from the database
- A new identifier attribute was created named *ID*, autonumber type. This attribute was necessary to identify specific records in the database, mainly during operations of data import and export, such as when sampling data
- A new attribute was created, called the *Rainfall_Range*, text type, size 1. This stores the rainfall categorical ranges (further discussed in this section)
- A new attribute was created, called the *Wind_Compass*, text type, size 8. This stores the values of wind direction represented as compass points (North, South, etc) instead of degrees (this transformation is further discussed in this section).

As already mentioned, the *Year* and *Day* attributes were necessary to calculate the previous afternoon dewpoint.

In the original database *Fog type* could assume three possible values: *F* when it was indicating a fog observation, *LF* when indicating local fog and null when it was not a fog observation. This study is concerned with the occurrence of fog, regardless of whether it is a local fog or not. For this reason all the *Fog type* instances with a value of *LF* were transformed into *F*, meaning a fog case. All instances of *Fog type* with a null value were assigned a *NF* value, meaning not fog. As a result, the *Fog type* attribute holds two possible values, *F* when it refers to an observation of fog, i.e. a fog case, or *NF* otherwise, i.e. a not fog case.

The *Past weather* and *Present weather* attributes were transformed from numeric type (their original data types) to non-numeric, type text, size 4 characters. These attributes are qualitative (categorical) attributes, indicating weather codes.

A transformation of the *Rainfall* attribute was required. Rainfall was initially measured in millimetres, but forecasters express rainfall in codes representing ranges of millimetres, rather than discrete values of millimetres. This procedure makes sense according to the nature of the forecast task, as it is almost impossible to differentiate precise measurements of rainfall, like 0.3 millimetres and 0.2 millimetres. For example, the small difference of 0.1 millimetres of rainfall does not have any practical effect in aviation weather forecasting. The forecast task, in general, has a tolerance for imprecision in order to conform best to reality.

As a result, the numeric values of rainfall were transformed into categorical codes, representing ranges of rainfall. Table 5.6 shows the rainfall code ranges (Meteorology, 2003a). Null values are classified as code zero (0), meaning no rainfall was observed.

Table 5.6: Categorical codes of rainfall attribute

Rainfall Code	Description
0	No Rain
1	0.1 to 2.4 mm
2	2.41 to 4 mm
3	4.1 to 9 mm
4	9.1 to 19 mm
5	19.1 to 39 mm
6	39.1 to 79 mm
7	Above 79 mm

To implement this operation a new attribute called the *Rainfall_Range* was inserted into the dataset, text type, with size 1. A Visual Basic procedure named *Transform_Rainfall* was implemented to update the *Rainfall_Range* categorical attribute according to the corresponding *Rainfall* numeric value, as described in Table 5.6.

The categorical representation of rainfall observation conforms better to the weather forecast task, but showed little impact in minimizing the problem of sparsity. Table 5.7 shows the frequency distribution of *Rainfall_Range* attribute in the whole population.

Table 5.7: Frequency distribution of Rainfall_Range attribute

Values	Frequency	Percent
0	41617	85.0%
1	6231	12.7%
2	572	1.2%
3	415	0.8%
4	79	0.2%
5	10	0%
6	2	0%
7	7	0%
Total	48933*	100%

*This analysis shows the results after the data preparation procedures were finalized, when some *not fog* records with high number of null values were removed from the database.

Observations with no rainfall are still highly predominant in both classes, although observations of code 1 have a better distribution for data mining purposes, followed by code 2. Figure 5.15 shows the frequency distribution histogram for *Rainfall_Range* attribute in the whole population.

Despite the sparse distribution along the range of its values, *Rainfall_Range* attribute was kept in the database.

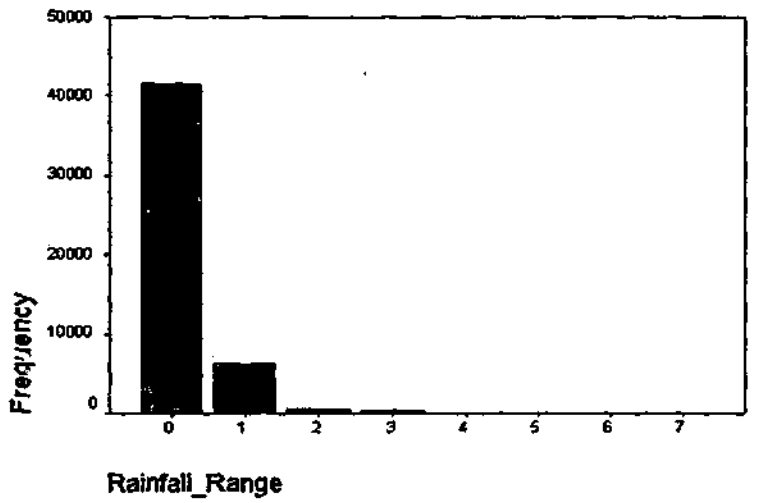

Figure 5.15: Frequency distribution for Rainfall_Range attribute in the whole population

Another problem was related to the wind direction observation. Wind direction is a measure taken by instruments and it is numerically represented in degrees of compass points, from 0° to 360°.

A high number of cases distributed in the compass points of 0° and 360° were identified in the fog population (refer to Figure 5.13). The problem here is to understand the meaning of 0° compass point, and whether a 0° could be considered the same information as 360°.

According to weather forecasters, a zero value assigned to a wind direction observation does not represent a compass point direction. It indicates no wind observed or even a variable weak wind blowing, with a speed equal to or approximately zero meters per second (0 m/s). The wind direction attribute had to be adjusted to properly represent this information.

Additionally, weather forecasters do not use detailed numerical measurements when reporting a forecast bulletin, but a categorical description of compass points, such as N for North, S for South and successively. Figure 5.16 shows an example of part of a meteorological observation bulletin for Melbourne (Meteorology, 2003a), which shows the wind direction representation (columns named WIND indicate the wind direction information).

Wind direction is represented in this research by the 16 compass points representation, from 0° degrees to 360° degrees. This representation is named VRB, and is used by the Australian Bureau of Meteorology (Meteorology, 2003a).

```
IDA33V00
BUREAU OF METEOROLOGY
VICTORIAN REGIONAL OFFICE
P.O. Box 1636M Melbourne Vic 3001    http://www.bom.gov.au

Three Hourly Meteorological Observations for Melbourne
Issued at 1406 on Thursday the 19th of October 2000

MELBOURNE                LAVERTON                MOORABBIN
HOURS   TEMP (C)  WIND (KNOTS)  TEMP (C)  WIND (KNOTS)  TEMP (C)  WIND (KNOTS)

3pm       19.6      E 6           18.5      N 18          18.2      NNE 15
6pm       17.9      E 7           17.1      NNE 16        16.2      NNE 12
9pm       18.2      E 6           16.8      N 9           17.2      NNE 18
Midnight  16.5      SW 3          14.0      NNW 26        16.3      N 9
3am       13.4      E 2           12.6      NW 6          13.4      NNE 7
6am       13.0      E 2           10.7      NNE 5         13.2      ENE 5
9am       16.6      ENE 5         16.3      N 8           16.3      N 12
Noon      19.0      E 5           18.1      WSW 9         19.0      NNE 11

Max to 2pm           20.3

Overnight Minimum    12.4
Yesterday's Maximum  19.7
Noon MSL Pressure    1009.0

Melbourne Ap Sunshine yesterday was 0 hrs
```

Figure 5.16: Three hourly meteorological observation bulletin for Melbourne

According to the Australian Bureau of Meteorology, each value in degrees belongs to the closest compass point. Therefore the middle value between each two compass points was chosen as the boundary value, with the middle value itself belonging to the next upper (clockwise) compass point. For example, between N (360° degrees) and NNE (22.5° degrees), the middle point is 11.25° degrees, which belongs to NNE, with 11.24° degrees belonging to N compass point (obviously, this procedure is a simplification of a more complex situation, which is to properly establish numerical boundaries among qualitative compass points).

To implement the transformation of wind direction from numerical degrees to qualitative description of compass points a new attribute was inserted into the dataset, a text type attribute named *WindCompass,* of size 8 characters. A Visual Basic procedure named *Convert_WindCompass* was implemented. This updates the *WindCompass* categorical attribute according to the values of *WindDirection* numerical attribute, using the compass points classification as showed in Table 5.8.

Table 5.8 shows compass points in degrees, their respective value range in degrees, their classification and descriptions.

Table 5.8: Compass points classification

Compass points	Value ranges in degrees	Compass points classification	Compass points description
0	0	CALM, wind speed = 0	Calm
	0	VARIABLE, wind speed > 0	Variable
360	[0.1 ; 11.24] and [348.75 ; 360]	N	Northerly
22.5	[11.25 ; 33.74]	NNE	North to North Easterly
45	[33.75 ; 56.24]	NE	North Easterly
67.5	[56.25 ; 78.74]	ENE	East to North Easterly
90	[78.75 ; 101.24]	E	Easterly
112.5	[101.25 ; 123.74]	ESE	East to South Easterly
135	[123.75 ; 146.24]	SE	South Easterly
157.5	[146.25 ; 168.74]	SSE	South to South Easterly
180	[168.75 ; 191.24]	S	Southerly
202.5	[191.25 ; 213.74]	SSW	South to South Westerly
225	[213.75 ; 236.24]	SW	South Westerly
247.5	[236.25 ; 258.74]	WSW	West to South Westerly
270	[258.75 ; 281.24]	W	Westerly
292.5	[281.25 ; 303.74]	WNW	West to North Westerly
315	[303.75 ; 326.24]	NW	North Westerly
337.5	[326.25 ; 348.74]	NNW	North to North Westerly

A final analysis was undertaken to verify the amount of null and missing values in the whole population, in particular *not fog*. As a result, some instances of *not fog* with null values assigned to most of their attributes were removed. As the amount of cases allotted to *not fog* class were sufficiently large, this operation would not cause any problem in terms of data availability.

Initially, the not fog population had 48963 instances (records in the database). After all data transformation and removal of instances with a significant amount of null values, not fog population had 47995 instances. The fog population remained with 938 instances, and the overall population had 48933 records. This database was used in selecting cases for data mining.

With the operations described in this section, the resulting database was considered satisfactory in terms of data quality and integrity to move to the next phase of this research, which is the data mining stage.

5.3.4 Data Modelling for Data Mining

After data preparation, the next stage is to select a set of data for mining and testing. For this it is necessary to verify data dimensionality, and how the classes are represented and distributed in the database.

Dimensionality concerns the amount of attributes and instances (cases) for data mining. Class distribution concerns the amount of instances presented for each class in the population. The weather observation database used in this research showed a very heterogeneous class distribution for *fog* and *not fog*, consequently a homogeneous analysis of both classes was very difficult. The weather observations database showed a low prevalence classification, being almost all cases allotted to *not fog*, and far fewer cases to *fog*. The dataset has 938 instances of *fog* and 47995 instances of *not fog*. Figure 5.17 illustrates the distribution of classes in the weather observations database.

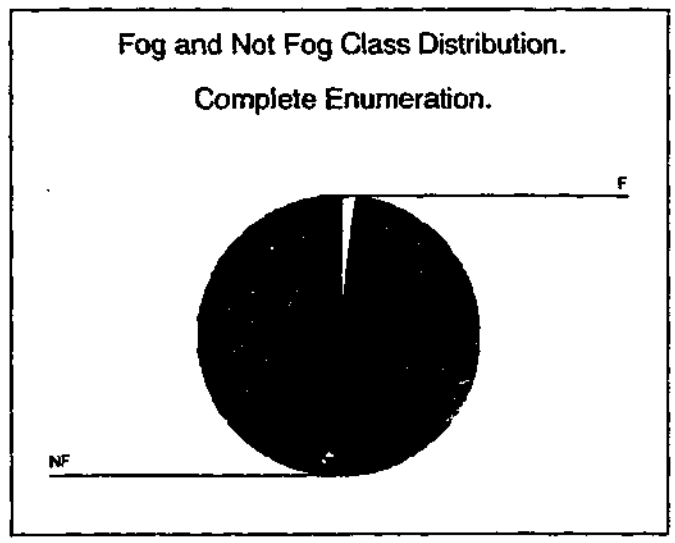

Figure 5.17: Distribution of classes in the population

Fog represents 1.92% of the population, and *not fog* represents 98.08% of the population. This difference in class distribution relates to the nature of fog occurrence, fog is a rare phenomenon. Consequently, a low number of fog cases are expected. The problem is that most learning algorithms do not behave well in such situations (Catlett, 1991; Provost and Buchanan, 1995). Furthermore, the dataset for data mining must capture sufficient information about each individual class, attributes and their respective interactions (Provost and Kolluri, 1999; Pyle, 1999).

Because of this significant difference in class distribution direct sampling is usually inefficient. Using a single random sampling approach might bring few problems, for instance the risk of no fog cases or too few fog cases being selected. If this happens the data mining algorithm will not generate good descriptive models about fog, and may not even generate a single model.

On the other hand, taking all fog population and an equivalent number of not fog cases, brings the risk of throwing away information in the unused not fog examples. As a consequence, most patterns of *not fog* class might be missed, as a very small subset of not fog population would be selected (approximately 1.95% of the not fog population).

The problem of imbalanced class distribution (or low prevalence classification) is not new. Many data mining and machine learning researches have studied and proposed solutions for this problem, for example (Fawcett and Provost, 1996; Lewis and Catlett, 1997). For instance, one possible solution is to apply a learning algorithm that can rank examples. Ling (Ling and Li, 1998), for example, applied a learning algorithm that ranked testing examples and used lift analysis as the evaluation criterion in a data mining application for direct marketing. Another possible solution is to implement a specific sampling design (Catlett, 1991; Provost, Jensen and Oates, 2001); for example, oversampling with replacement the positive (fog) cases a few times, while keeping the negative (not fog) cases unchanged, or simply sampling the negative class in higher proportions than the positive class.

It is almost impossible to know in advance which technique works better in a given circumstance, and time constraints make it unrealistic to experiment with many approaches in a single situation. As such, a decision has to be made.

A specific sampling design was implemented in this research to deal with the low prevalence classification problem. It combines **stratified** sampling with **incremental** sampling.

Next, the developed sampling design to generate the datasets for *mining* and *testing* is discussed. A third dataset was also generated, called the *evaluation* dataset. This dataset was generated to handle the problem of overfitting (if necessary).

5.3.4.1 Sampling Data

The sampling approach applied to generate the data models for mining and testing in this research project can be classified as *stratified multi-stage sampling* (refer to Chapter 3, section 3.6 for a discussion about data sampling). The primary purpose of this sampling approach is to achieve a more equal class distribution.

The original population was divided into two strata, and then sampling was separately conducted in stages within the stratum with the highest number of instances (the *major* stratum).

In the first stage, *stratified sampling* was used to separate the survey variable *Fog type*. Stratified sampling consists of dividing a population into sub-populations, called strata. Then small samples are selected from these different strata independently of each other. At the end, the total sample is formed by combining (some of) the small samples (Gu, Hu and Liu, 2001).

In the second stage, *incremental random sampling*[1] was used to sample the major *not fog* class. Incremental sampling is a method in which the sampling size is updated and the data are randomly sampled, starting with a small sample and using incrementally larger samples until the model accuracy does not significantly improve (Weiss and Indurkhya, 1998) (refer to section 3.6). As a result of this approach, three datasets from *not fog* class were generated.

In the third stage, *random sampling without replacement* was applied to each sample in each class to generate the datasets for mining and testing (and evaluation).

In the last stage, subsets of the initial population were reconstructed by joining each *not fog* dataset with their respective *fog* dataset (i.e. mining and testing datasets from each class were joined), obtaining as a result the final datasets for mining, testing and evaluation.

Following, the developed sampling design is described.

Firstly, stratified sampling (Gu, Hu and Liu, 2001) was used to individually sample *fog* and *not fog* classes, as the *Fog type* attribute showed a very heterogeneous class distribution (see Figure 5.17). Figure 5.18 illustrates this approach.

[1] A similar approach is introduced by Provost (Provost, Jensen and Oates, 2001) as *progressive sampling*.

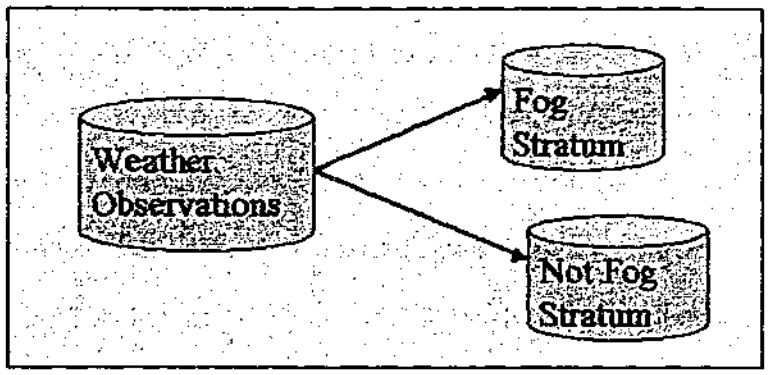

Figure 5.18: Stratifying the weather observations database

As a result, two strata were obtained, one with *fog* cases, called fog stratum, and another with *not fog* cases, called not fog stratum, as illustrated in Figure 5.18.

Sampling the Fog Class

The *fog* stratum has 938 instances in its population. It corresponds to the most interesting class and it is also a low prevalence class in comparison with the whole population.

Therefore a complete enumeration of fog stratum was used to generate the datasets for *mining* and *testing*. The fog stratum was *randomly sampled without replacement* in 85% for mining, with 807 rows of data, and 7% for testing with 63 rows of data.

These percentages were arbitrarily chosen based on the experiments and literature previously mentioned in sections 3.5 and 3.6, about dimensionality reduction and data sampling.

Table 5.9 shows the selected datasets from fog stratum:

Table 5.9: Fog class datasets

Dataset	Number of instances
Population size (N)	938
Sample size (n)	938 (100% of N)
Mining set	807 (85% of n)
Testing set	63 (7% of n)

Sampling the Not Fog Class

The *not fog* stratum has 47995 instances. It was sampled in a different fashion than fog stratum, as increased sizes samples were selected from the whole stratum in 10%, 20% and 100% proportions.

Incremental sampling with replacement was used to generate the samples. The sample with 10% out of the whole stratum was termed *Model1*, and it has 4763 instances. The second sample, *Model2*, is 20% of the stratum, and has 9572 instances. For comparison purposes the whole stratum was also considered, and this was termed *Model10*, meaning 100% of the stratum.

The 10% and 20% percentages were arbitrarily selected based on the size of not fog and fog strata; the aim here was to build data models to minimize the significant difference between the numbers of instances from each class. Therefore small percentages were chosen from the not fog stratum. In addition, the literature provides useful insights in incremental sampling; according to Weiss (Weiss and Indurkhya, 1998) typical subset percentages for incremental sampling might be 10%, 20%, 33%, 50%, 67% and 100%. Using 50% and higher percentages would keep the difference between not fog cases and fog cases too large, therefore small percentages were chosen. The 100% subset was selected to verify the data mining algorithm performance when using a significantly different class distribution. The assumption here was that this subset would produce none or very few fog cases rules.

The sample models obtained are termed *data models* in the context of this research project, e.g., *Model1*, *Model2* and *Model10* are classified as data models.

Table 5.10 shows the sample models generated from the not fog stratum, sample size indicates the number of instances.

Table 5.10: Sample models from not fog stratum

Sample Model	Sample size	Percentage of the stratum
Model1	4763	10%
Model2	9572	20%
Model10	47995	100%

The next step is to generate, from each data model, datasets for mining and testing.

The new problem to be solved here concerns the size of the mining and testing data subsets; and this is one of the most important issues in data modelling for data mining.

Normally the whole dataset is randomly divided into about 80% for training purposes and 20% for testing, when the original dataset has a size of approximately 1000 instances (Weiss and Indurkhya, 1998). Several exploratory studies and experiments handling data sampling are reported in the literature about knowledge discovery, knowledge engineering and machine

learning. Refer to section 3.6.2, "Designing the Training Dataset," for examples illustrating data sampling (Table 3.2 summarizes various experiments) and discussions about generating datasets for data mining. Generally, choosing a suitable mining dataset is an empirical activity rather than an exact science.

According to Weiss (Weiss and Indurkhya, 1998) and based on the experiments described in section 3.6.2, it was decided to select proportions of 60% and 80% from each sample (data model) to generate the mining datasets. And proportions of 10% of the remaining amount in each data model were selected from testing datasets.

From *Model1* two sets of data for mining and testing were obtained.

These were a mining dataset of 60% of *Model1*, with 2836 instances, called *Model1-60*, and a testing dataset of 10% of *Model1*, with 500 instances. The mining and testing samples were *randomly selected without replacement*, having different instances from each other.

A second mining dataset was generated from *Model1*, with 80% of the sample, having 3869 instances. A testing dataset was again generated, with 10% proportion *randomly selected without replacement* from the *Model1* sample, having 456 instances. This model is named *Model1-80*; which means 10% from the not fog stratum was used in the sample, with 80% of the sample selected for the data mining set. Figure 5.19 illustrates the incremental sampling approach of *Model1*.

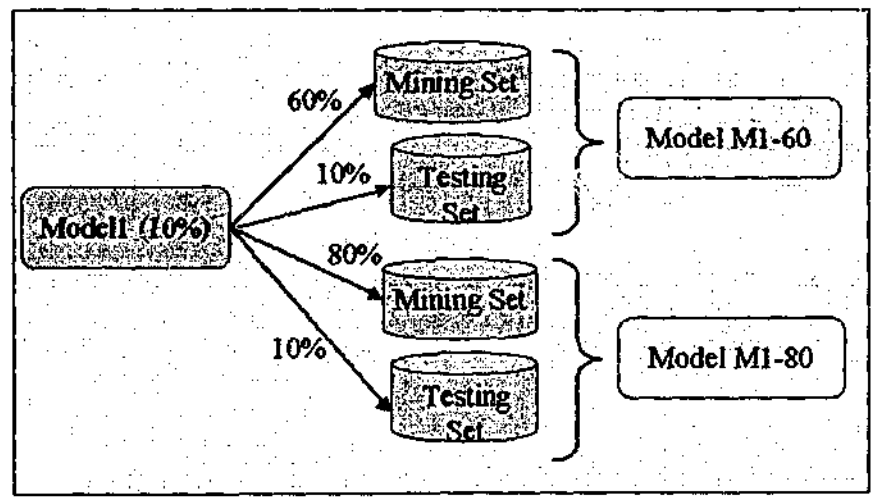

Figure 5.19: Generating data mining and testing sets from not fog stratum

A similar approach was applied to *Model2*; however only a mining set in a 60% proportion was generated, as an 80% proportion would represent a significant difference between the

amount of not fog and fog cases. A mining dataset of 60% of *Model2* with 5699 instances was *randomly* (without replacement) selected. Following, a testing dataset was *randomly selected without replacement*, being 10% of the remaining proportion of *Model2*, having 992 instances. This data model is named *Model2-60*.

Lastly, a data mining set was generated from the whole not fog stratum, using 60% of the population having 28707 instances. A testing dataset was also generated from the whole not fog stratum being 10% of the remaining population (excluding the instances already selected for mining) with 4802 instances. This data model is named *Model10-60*, meaning 100% of not fog stratum was used, with a 60% proportion selected for the data mining set.

Table 5.11 shows the generated data models from the not fog stratum.

Table 5.11: Data models from not fog stratum

Data Model	Model1-60	Model1-80	Model2-60	Model10_60
Population size (N)	47995	47995	47995	47995
Sample size (n)	4763 (10% of N)	4763 (10% of N)	9572 (20% of N)	47995 (100% of N)
Mining set	2836 (60% of n)	3869 (80% of n)	5699 (60% of n)	28797 (60% of n)
Testing set	500 (10% of n)	456 (10% of n)	992 (10% of n)	4802 (10% of n)

As previously mentioned in this chapter, from each data model an additional dataset was obtained, in 10% proportions. The purpose of these datasets was to handle the problem of overfitting when training the neural network model, if necessary. These datasets were termed *evaluation datasets*, and they were *randomly sampled without replacement* after the data mining and testing sets were generated from each data model. The evaluation dataset from Model1-60 has 490 instances, Model1-80 has 438 instances, Model2-60 has 949 instances, and Model10-60 has 4908 instances.

The obtained mining sets were considered sufficient to cover various proportions of the not fog stratum, without requiring an excessive amount of time for running the data mining experiments.

The data models, including the testing datasets, obtained through the employed sampling approach are completely disjoint sets of data. Individual experiments were conducted to assess which data model produces more accurate results, this is discussed in Chapter 7.

The next stage in this research is to effectively mine the data, i.e. to generate the data mining models.

5.3.4.2 Generating Data Mining Models

The final datasets for data mining, called *mining models*, were obtained by combining the fog mining dataset with each not fog mining dataset. Therefore four mining models were obtained, identified by their corresponding sampling proportions: *Mining Model1-60*, *Mining Model1-80*, *Mining Model2-60*, and *Mining Model10-60*.

For instance, *Mining Model1-60* identifies a dataset obtained by a sample of 10% of the not fog stratum, and a proportion of 60% of this sample was allocated for the data mining set. This data mining set was then joined with the data mining set generated from the fog stratum, resulting in the final mining set, including fog and not fog instances. The other data mining models follow the same structure.

Table 5.12 describes the obtained models for data mining, identifying how each data mining model is compounded and their respective amount of cases (rows of data) from each class, as well as the total amount of cases. All samples were generated using the SPSS statistical package.

The testing datasets were not used in the data mining stage, only the mining datasets. Testing datasets were used at a later stage, when the generated rule sets were applied for training the neural network based system, and testing datasets were used to assess the performance of the DM-NN model. And the evaluation datasets would be used to handle to problem of overfitting in the neural network, if necessary.

Table 5.12: Data mining models

Mining Model	Mining Model 1-60	Mining Model 1-80	Mining Model 2-60	Mining Model 10-60
Mining sets	Fog mining + NotFog mining Model 1-60	Fog mining + NotFog mining Model 1-80	Fog mining + NotFog mining Model 2-60	Fog mining + NotFog mining Model 10_60
Rows in fog class	807	807	807	807
Rows in not fog class	2836	3869	5699	28797
Total number of rows	3643	4676	6506	29514

With the data models obtained for data mining, the next step is applying the data mining component to generate descriptive models of fog occurrence, and then populate the knowledge rule bases with the sets of association rules obtained.

5.4 Building Knowledge Bases: Mining Data

The next stage in applying the DM-NN model is to effectively mine the data mining models obtained (see Table 5.12), and as a result to generate the knowledge rule bases. The knowledge rule bases are the learning (training) datasets used by the artificial neural network system CANN (refer to section 4.5).

Knowledge discovery in databases constitutes an interactive and iterative process, having many steps. This research project distinguishes the activities of *domain modelling, data modelling* and *knowledge modelling.*

In the scope of this research domain modelling is considered in the same way as it has been widely used by decision support, expert systems and the artificial intelligence community in general. Basically, it is concerned with building a model of a particular domain under investigation for any particular purpose. In this thesis section 5.2 and section 5.3, including subsection 5.3.2, "Understanding Meteorological Data," are concerned with domain modelling.

Data modelling in the context of this research relates to all the activities that transform raw data into the data used for data mining. Such data modelling includes data pre-processing, features selection, reduction and transformation, and data sampling. Section 5.3.3, "Data Preparation," and section 5.3.4, "Data Modelling for Data Mining," are concerned with data modelling.

Knowledge modelling in this context includes the activities related to extracting knowledge from data. This includes the interactive process of mining data, testing and tuning different data mining parameters and data models, e.g., adding or eliminating data features, and even cases. It is effectively an interactive and iterative process, where the purpose is to build descriptive models the most comprehensible as possible about the application domain under

study. The ultimate goal of such a knowledge modelling process is to achieve a good predictive performance of the decision support model.

The above definitions are important for a better understanding of how the knowledge models were generated and what they are in this research. The approach used here to generate knowledge is based on the data models, a data mining algorithm (the descriptive method) and the choice of data mining parameters (rule confidence degree, rule support and maximum rule order). For each mining data model, and combinations of mining parameters, distinct sets of association rules were obtained. Each of these distinct sets of association rules is identified as a *knowledge model*, and populates a particular *knowledge rule base*.

The next step in the research described in this thesis is knowledge modelling. This includes the interactive process of mining data, testing and tuning different data mining parameters and data models, e.g., adding or eliminating data features, and even cases. The way these processes were conducted in this research project are described next.

5.4.1 Applying Data Mining

There are various procedures that must be done when applying data mining, such as features selection, selection of feature values, selection of a target attribute (also known as survey variable), discretization of numeric attributes, and the selection of mining parameters. Almost all data mining projects require the execution of these procedures at a certain level.

5.4.1.1 Selecting the Target Attribute

Selecting the target attribute concerns identifying the attribute in the database that represents the target of the data mining process. This means the attribute that discriminates the class under study, or the subject that has to be described. This attribute forms the consequent part of the rules. As this research project aims to describe situations where fog is most likely to occur, the attribute in the database that identifies fog occurrence is the selected survey variable, i.e. the *FogType* attribute. *FogType* attribute was discretised in two possible values, *F* and *NF*, representing whether a particular weather observation (case) refers to a fog case (*F*) or not (*NF*), respectively.

5.4.1.2 Features Selection

Features selection relates to choosing the attributes that form the antecedent part of the rules. In this research almost all attributes were selected for data mining, except for the *Year* and *Day* attributes, as already mentioned. The *Year* attribute was not used because a particular year only happens once and is never repeated, as such a value of a particular year does not categorize any situation. The *Day* attribute was not used because the forecasters classified this information as not relevant for forecasting purposes.

The *Visibility* attribute represents the visibility over the airport runaway. This attribute might be considered synonymous with the attribute that identifies fog occurrence, and consequently could be discarded. For instance, low visibility values would indicate a definitive occurrence of fog, and high visibility values would indicate a not fog case. Data mining experiments selecting and not selecting *Visibility* attribute were conducted, with the aim of verifying the impact of this attribute in identifying fog.

The objective of these experiments was to verify the generated rules in both classes (*fog* and *not fog*) and the prevalence of *Visibility* attribute, and ultimately, the predictive performance of the DM-NN model when trained with and without *Visibility* in the training set. Section 7.5 discusses these experiments.

5.4.1.3 Selection of Attribute Values

Selection of attribute values is a procedure that addresses dimensionality reduction, together with feature and case selection. It is possible that some attribute values are not relevant to the survey variable, or have a small frequency of occurrence in the database, or have a high frequency for either *fog* or *not fog*. In all these cases, such attribute values can be discarded for data mining purposes.

Attribute values with a small frequency of occurrence in the database do not specify any pattern and are usually ignored by most data mining (and machine learning) algorithms. On the other hand, attribute values with a high frequency of occurrence in all classes do not discretize any class; and as such have no value for classification purposes.

Some values of *Hour*, *Month* and *Rainfall* attributes were excluded from the data mining experiments because they had either a small frequency in the database, or because they had a high frequency for either classes, *fog* and *not fog*.

Figure 5.20 illustrates the frequency distribution for the *Hour* attribute in the whole population.

Figure 5.20: Frequency distribution for the Hour attribute

In this attribute, the values 11, 14, 17, 2, 20, 23, 5 and 8 were discarded, as they have a very low frequency of occurrence, below 0.5%. It can be observed in Figure 5.20 the small number of occurrences for some values, such as 2 and 20, among others. This small frequency implies that these values are likely to be ignored by the data mining algorithm; consequently there is no need to select these values. It should be noted that the Apriori algorithm counts the sets of the most frequently occurring groups of items, named large itemsets, as discussed in section 3.3.2.1

Considering the rainfall observation, only the categories 0, 1, 2, and 3 were selected in the *Rainfall_Range* attribute. The remaining categories have a frequency below 0.5%, considered very small for data mining purposes, as shown in Table 5.7 and Figure 5.15 in this chapter.

Considering the month observation, only the values of 4 (April), 5 (May), 6 (June), 7 (July), 8 (August), 9 (September) and 10 (October) were selected, as there is no occurrence of other values for the month attribute in the database, as shown in Figure 5.21.

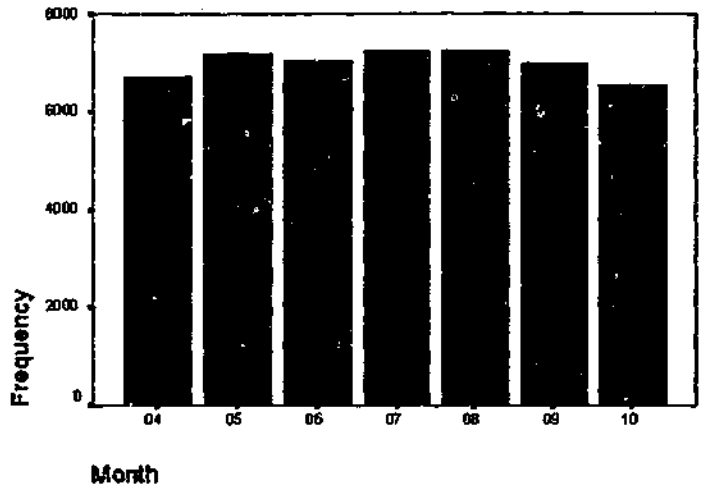

Figure 5.21: Frequency distribution for Month attribute

The remaining attribute values were selected in all data mining experiments.

5.4.1.4 Discretization of Numerical Attributes

Discretization of numerical attributes is used to determine the granularity of a certain variable and can be used to simplify the data mining problem. Most data mining tools and algorithms, mainly those used to generate association rules, require numerical continuous variables to be discretised in categorical ranges, or **bins**. This involves dividing the range of values into subranges and using the subranges as substitutes for continuous values. This process is called **binning** in some literature (Fayyad and Irani, 1993). The rules generated in data mining processing describe ranges of values represented by a discrete value. Categorical attributes already express a discrete value, consequently there is no need to discretise them. However numerical attributes have to be discretised in ranges.

For example, for some decision purposes, air temperature may be discretised into the categories *hot*, *warm*, *cold*, and *freezing*. Each of these four alpha labels (categories) represents part of the air temperature range. A rule representing some trivial relation could be: "If it is *cold* than use a coat when going out," instead of: "If the temperature is *between minus 10 Celsius degrees and 5 Celsius degrees* than use a coat when going out." In this example, the temperature range of *minus 10 Celsius to 5 Celsius degrees* is replaced by the category of *cold*, which is a more natural way of expressing such a situation.

In this research project, numerical attributes were discretised into categorical bins to generate association rules. For example, the wind speed was discretised into four categories[1]: *light*, *light moderate*, *moderate*, and *fresh moderate*. Figure 5.22 illustrates how binning works for the wind speed attribute.

Domain knowledge facilitates setting the bin boundaries, identifying where meaningful boundaries fall. When domain knowledge is not available, some rationale has to be applied. For example, to assign bin boundaries so that each bin contains approximately the same range size. Normally, the discretization of a particular attribute depends on the amount of cases in the database and the frequency of each value in the attribute value range (Fayyad and Irani, 1993), i.e. the discretization is a measure proportional to the total amount of cases in the mine set and the frequency of occurrence of each attribute value.

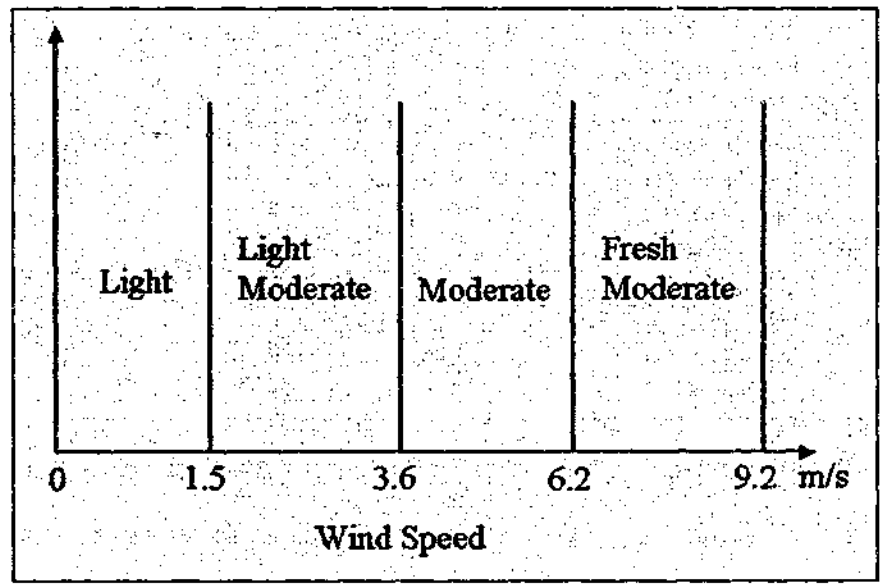

Figure 5.22: Discretizing continuous values of wind speed

The data mining algorithm employed in this research includes a method that automatically discretizes numerical attributes during pre-processing, based on the attribute frequency distribution in the mine set and the number of their categories, aiming to achieve a uniform grouping.

Because the number of cases varies in each data mining model, each numerical attribute was discretised, keeping constant the number of cluster (bins) in each attribute for all data

[1] The categories for wind speed used by the Bureau include *strong* and *galeforce*. Those categories were not included in this research as there was no occurrence of these wind categories in the mine sets.

mining models. Table 5.13 illustrates the clusters, their boundaries and labels for all numeric attributes in *Model1-60*.

Table 5.13: Discretization of numerical attributes for Model1-60

Attribute	Numerical ranges	Labels
Dry Bulb (Celsius degrees)	< = 8.5 > 8.5 and <= 12 > 12	Low Med (medium) High
Dew Point and Previous Dew Point (Celsius degrees)	< = 4 > 4 and < = 6 > 6 and < = 9 > 9	Low Med (medium) High Max (maximum)
Total Cloud Amount (Eighths)	< = 4 > 4 and < = 7 > 7	Min (minimum) Med (medium) Max (maximum)
Total Low Cloud Amount (Eighths)	< = 1 > 1 and < = 6 > 6	Min (minimum) Med (medium) Max (maximum)
Sea Level Pressure (HpA)	< = 1014.1 >1014.1 and <= 1020.2 >1020.2 and <= 1025.6 > 1025.6	Low Med High VHigh (very high)
Wind Speed (meters/second)	< = 1.5 > 1.5 and < = 3.6 > 3.6 and < = 6.2 > 6.2	Light LMode (light moderate) Mode (moderate) FMode (fresh moderate)
Visibility (Kilometers)	< = 25 > 25 and < = 35 > 35 and < = 40 > 40	Lev1 (level 1) Lev2 (level 2) Lev3 (level 3) Lev4 (level 4)

The binning approach follows in the same fashion for all data mining sets used in this research. The discretization values and ranges for *Model1-80*, *Model2-60* and *Model10-60* are included in Appendix B.

5.4.1.5 Selecting Mining Parameters

Mining parameters (thresholds) orient the processing of rules, specifically the pruning of uninteresting rules, and ranking rules according to some criteria. As a result of the pattern discovery process, many rules are likely to be generated. Depending on the size of the database, thousands of rules or even more, could be obtained. Some discovered rules might not be interesting for several reasons. For example, a rule can correspond to already known information, as such uninteresting, a rule can refer to uninteresting attribute combinations,

and rules can be redundant (Weiss and Indurkhya, 1998). For those reasons, pruning and sorting procedures are often necessary.

Generally, when evaluating the quality (or interestingness) of a rule, the common factors that are taken into account are the coverage, the completeness, and the confidence of the rule. Pruning and sorting criteria for association rules are *rule confidence*, *frequency*, and *support degree* (Klemettinen, Mannila and Toivonen, 1997). Rules are pruned and ranked by setting thresholds on these parameters.

The parameters for rule filtering used in this research are: the minimum level of *rule confidence* and *support degree*, the minimum *number of records* in the database that support a particular rule, e.g., the minimum number of cases for which a rule is verified. And the maximum *rule order*, e.g. the maximum number of rule antecedent itemsets.

Confidence and support constraints are important because they work as filters for relevant rules; in general, relevant rules have support and confidence degrees above some minimum threshold. If the support and confidence are not large enough, the rule is not relevant enough and can be discarded. In brief, the support degree represents the ratio of the number of the records in the database for which the rule is true to the total number of records in the database. The confidence degree expresses the belief in the consequent being true for a rule once the antecedent is known to be true. Refer to section 3.3.2.1, "The Apriori Algorithm for Association Rule Learning," for a formal definition of support and confidence degrees.

The problem is to effectively assign the best-fit set of such parameters in a given situation. Thresholds that are too restrictive are likely to miss important information, but on the other hand, overly flexible thresholds might produce too much uninteresting information.

The weather observation database used in this research presented an unbalanced class distribution. Although this problem was minimized through the applied sampling design, class distribution remains a concern. For instance, the data mining model *Model1-60* has 807 cases of fog and 2836 cases of not fog (refer to Table 5.12). The problem is that it is generally easy to discover rules predicting the major class, but difficult to discover rules predicting the minor class (Kononenko, Simec and Robnik-Sikonja, 1997). This problem is worth avoiding because it might have an impact later on the neural network training performance.

Furthermore, there are computational costs associated to threshold levels; in general, restrictive thresholds help reduce the search space, consequently reducing computational costs. For instance, (Klemettinen, Mannila and Toivonen, 1997) shows the effect of setting different rule confidence and frequency thresholds in a data mining application using an alarm database.

In this research rules with 70%, 80% and 90% confidence degrees were generated. As it was impossible to know beforehand the amount of rules that could be obtained according to a specific confidence degree, it was decided to use the most frequent thresholds applied in data mining applications (Weiss and Indurkhya, 1998; Piatetsky-Shapiro and Frawley, 1991; Klemettinen, Mannila and Toivonen, 1997). The goal was to verify if there was a significant difference in performance according to different combinations of parameters (confidence degree, minimum support degree and maximum rule order). And, if so, which combination(s) of these parameters leads to a better classificatory performance.

Rules with 50% confidence degree were generated specifically for *Model10-60*. This model contains the whole population, and an extremely unbalanced class distribution. As such, a more flexible confidence degree was allowed in order to obtain a higher number of rules for the *fog* class (it should be noted that in this data mining model the difference between fog and not fog cases is very significant, as shown in Table 5.12).

Two sets of data mining experiments were conducted. In the first set of experiments, 50% (for Model10-60 only), 70%, 80%, and 90% of rule confidence degrees were used. The minimum rule support was set to 8% and the maximum rule order to 7, with the minimum number of cases as 50 cases. These parameters were arbitrarily selected. This first set of experiments is identified as **V2**.

In particular, the maximum rule order of seven antecedent itemsets was selected because it was verified that forecasters do not use more than seven pieces of information when issuing a forecast report, and normally it is less than seven.

A second set of experiments was conducted, using a minimum rule support of 6% and a maximum rule order relaxed to 10 itemsets. The levels of confidence degree remained the

same in both set of experiments, as well the minimum amount of cases for each rule, i.e. 50 cases. This second set of experiments is identified as **V3**.

Table 5.14 shows the data mining parameters selected in both sets of experiments.

Table 5.14: Mining parameters and thresholds

Mining Parameter	Thresholds	
Confidence Degree	50%, 70%, 80%, 90%	
Minimum Number of Cases	50	
Minimum Support Degree	8%	6%
Maximum Rule Order	7	10

The sets of rules obtained are identified as knowledge models and populate a particular knowledge rule base.

The reason for conducting these two experiments is that the first experiment resulted in very restrictive models, with a small number of rules describing fog class. Consequently, it was decided to repeat the data mining experiments with more flexible thresholds, aiming to achieve a higher amount of rules describing fog class. As such, the minimum rule support was relaxed to 6% and a higher amount of rule itemsets were allowed, setting the maximum rule order threshold to 10.

5.4.2 Generating Knowledge Models

The approach used in this research to generate knowledge was based on data mining models (see Table 5.12), a data mining algorithm (the descriptive method, see section 4.2) and a set of data mining parameters and respective thresholds (rule confidence degree, rule support and maximum rule order). For each data mining model, and combinations of mining parameters, a distinct set of association rules was obtained. Each of these distinct sets of association rules is identified as a *knowledge model,* and populates a particular *knowledge rule base.*

For instance, Model1-60, with 70% minimum confidence degree, minimum rule support of 8%, and maximum rule order of 7, populates a particular knowledge rule base **as a result of data mining processing.**

Similarly, Model1-60, with 80% minimum confidence degree, minimum rule support of 6%, and maximum rule order of 10, populates a distinct knowledge base as a result of data

mining processing. The process follows in this fashion until all data mining models are processed with the selected data mining parameters and thresholds (refer to Table 5.14).

As already mentioned, two sets of mining experiments were conducted with all data mining models, according to different combinations of parameters (refer to Section 5.4.1.5, "Selecting Mining Parameters.") These two sets of experiments, identified as **V2** and **V3**, are described next.

5.4.2.1 Knowledge Models V2

The first data mining experiments resulted in association rules with 50% (for Model10-60 only), 70%, 80%, and 90% of confidence degrees[1], a minimum rule support of 8% and a maximum rule order of seven, with a minimum number of cases as 50. This group of experiments is identified as **V2**.

Table 5.15 shows the amount of rules obtained from each data mining model, according to their confidence degrees. In Table 5.15 *F* identifies *fog* class and *NF* identifies *not fog* class.

Table 5.15: Rule sets by data mining models in experiment V2

Data Mining Model	Generated Rules by Rule Confidence Degree 'Rule Support 8%', Maximum Rule Order 7											
	50%			70%			80%			90%		
	F	NF	Total	F	NF	Total	F	NF	Total	F	NF	Total
Mining Model1-60				54	186	240	22	180	202	10	122	132
Mining Model1-80				35	180	215	16	180	196	8	128	136
Mining Model2-60				32	177	209	18	177	195	10	169	179
Mining Model10-60	17	170	187	8	170	178	7	170	177	7	170	177

As can be observed from Table 5.15, the results obtained in these first experiments were considered very restrictive, as the numbers of rules obtained, specifically for fog class, were considered too small for a good descriptive capability. The highest number of association rules describing the fog class was 54, obtained through Model1-60, using a 70% confidence degree.

[1]The confidence degrees indicate the minimum level of confidence accepted, rather than an absolute value. For example, a data mining experiment with a 70% confidence degree indicates that 70% is the minimum level; rules with higher levels than 70% are also included.

In the same experiment, the number of association rules describing the not fog class was 186, a difference of 132 rules, as shown in Table 5.15.

The experiment using Model10-60 resulted in very poor models, with few association rules allotted to the fog class and a significant difference between the amount of rules for fog and not fog classes. Even relaxing the confidence degree to 50%, only 17 association rules were obtained for the fog class, contrasting with 170 association rules for not fog class, as indicated in Table 5.15. This difference might prevent a good descriptive capability for fog class from this data mining model.

A similar interpretation can be extended to the experiments using a 90% confidence degree. The resulting association rule sets have 132, 136, 179 and 177 rules, with 10, 8, 10, and 7 rules describing fog class, respectively. The maximum number of 10 rules allotted to the fog class was obtained with Model1-60 and Model2-60. Again, this number of rules, 10, is unlikely to be enough for a satisfactory description of fog phenomenon.

Using a confidence level of 80%, an increase in the number of rules for the fog class was obtained, but an increase in the number of rules for the not fog class was also obtained. Meanwhile the higher amount of rules assigned to fog class helps improve the descriptive capability for this class, the proportional increase of rules for the not fog class might not help achieve a better predictive performance, as the rules distribution between both class remains highly heterogeneous.

Figure 5.23 illustrates the rules distribution in each class, fog and not fog.

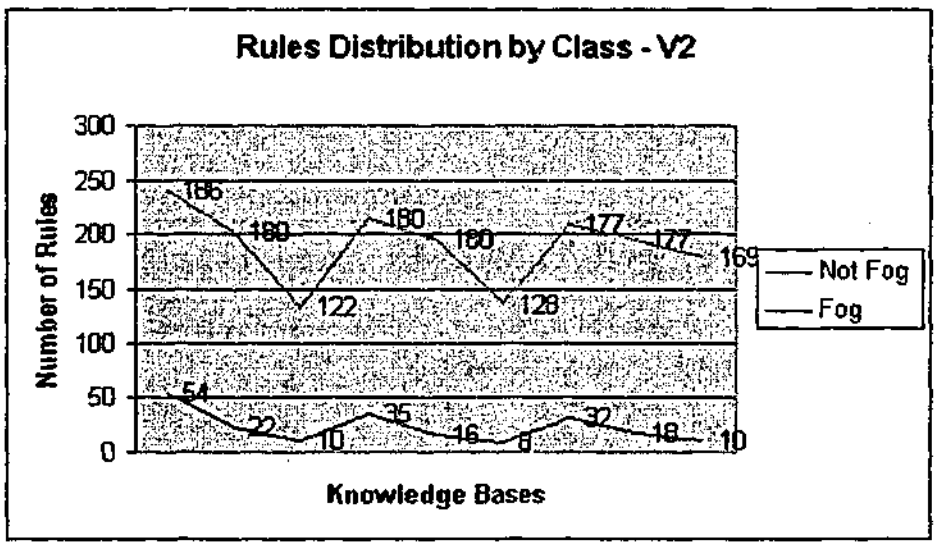

Figure 5.23: Rules distribution for Fog and Not Fog Class, in experiment V2

The better models were obtained with a confidence level of 70%, with 54 association rules assigned to fog class using Model1-60, 35 association rules assigned to fog class using Model1-80, and 32 using Model2-60 (refer to Table 5.15).

It is very difficult to assess in advance the classificatory performance of the DM-NN model, when training the neural network model with the knowledge rule bases obtained. However, as already mentioned, is generally easy to discover rules predicting the major class, but difficult to discover rules predicting the minor class (Kononenko, Simec and Robnik-Sikonja, 1997).

Intuitively, it is easy to recognize that assumption, specifically in this research, as the used neural network learning algorithm implements a supervised learning approach based on the error correction algorithm. This means that the number of examples in each class under study affects the algorithm learning rate. Refer to section 2.4.4, "Neural Networks Learning Approaches," for discussions on this subject.

Consequently, it seemed necessary to achieve a more equal distribution between the amount of rules describing either classes, or at least, to increase the number of rules describing the fog class. This is because the difference in number of rules, or the small number of rules assigned to the minor class, fog, might have a negative impact on the neural network training performance. As such it was decided to repeat the experiments using a more relaxed set of mining parameters. The next section discusses these experiments.

For illustration purposes, Figure 5.24 shows a set of association rules discovered through the data mining processing of Model1-60, using a 70% minimum confidence degree.

For instance, the first association rule in Figure 5.24 indicates a fog case, attribute *FOGTYPE=F*, with 91.82% rule confidence, 12.52% rule support, and found in 101 cases.

The itemsets of this first association rule are:

Dry Bulb temperature less or equal to 8.5 degrees Celsius, *Total Cloud Amount* higher than 7 heights, *Total Low Cloud Amount* higher than 6 heights, and no *Rainfall* observed.

The numeric attribute values were later transformed into their respective labels, as described in Table 5.13, before the rules were presented to the neural network system as part of the integration of the knowledge rule bases within the CANN system.

ASSOCIATION RULES - MODEL1-60V270-NV ITEMSETS	CONSEQUENCE	CONFIDENCE %	N. OF CASES	SUPPORT %
IF DRYBULB <= 8.5 & TOTALCLO > 7 & TOTALLOW > 6 & RAINFALL = 0	THEN FOGTYPE = F	91.82	101	12.52
IF DRYBULB <= 8.5 & TOTALCLO > 7 & SEALEVEL > 1025.6 & RAINFALL = 0	THEN FOGTYPE = F	91.55	65	8.5
IF DRYBULB <= 8.5 & TOTALCLO > 7 & TOTALLOW > 6 & SEALEVEL > 1025.6	THEN FOGTYPE = F	91.36	74	9.17
IF TOTALCLO > 7 & TOTALLOW > 6 & RAINFALL = 0 & WINDCOMP = CALM	THEN FOGTYPE = F	86.46	83	10.29
IF MONTH = 06 & DRYBULB <= 8.5 & WINDSPEE <= 1.5	THEN FOGTYPE = F	74.2	94	11.65
IF DRYBULB <= 8.5 & DEWPOINT > 6 & DEWPOINT <= 9 & WINDSPEE <= 1.5	THEN FOGTYPE = F	73.64	81	10.4
IF DRYBULB <= 8.5 & TOTALLOW > 6 & WINDSPEE <= 1.5	THEN FOGTYPE = F	73.57	103	12.76
IF MONTH = 07 & DRYBULB <= 8.5 & WINDSPEE <= 1.5	THEN FOGTYPE = F	72.82	75	9.29
IF HOUR = 6 & TOTALCLO > 7 & TOTALLOW > 6	THEN FOGTYPE = F	72.22	65	8.5
IF TOTALCLO > 7 & WINDCOMP = CALM	THEN FOGTYPE = F	72.11	106	13.14
IF TOTALCLO > 7 & TOTALLOW > 6 & RAINFALL = 0	THEN FOGTYPE = F	71.17	195	24.16
IF DEWPOINT > 6 & DEWPOINT <= 9 & TOTALCLO > 7 & WINDSPEE <= 1.5	THEN FOGTYPE = F	70.71	70	8.67
IF PREVIOUS > 6 & PREVIOUS <= 9 & TOTALCLO > 4 & TOTALCLO <= 7 & PASTWEAT = 2 & RAINFALL = 0	THEN FOGTYPE = NF	99.58	239	8.43
IF TOTALCLO > 4 & TOTALCLO <= 7 & PASTWEAT = 2 & WINDCOMP = N	THEN FOGTYPE = NF	99.58	238	8.39
IF TOTALCLO > 4 & TOTALCLO <= 7 & TOTALLOW > 1 & TOTALLOW <= 6 & WINDSPEE > 6.2	THEN FOGTYPE = NF	99.57	233	8.22
IF TOTALCLO > 4 & TOTALCLO <= 7 & SEALEVEL <= 1014.1 & WINDSPEE > 6.2	THEN FOGTYPE = NF	99.56	227	8
IF PREVIOUS <= 4 & PASTWEAT = 2	THEN FOGTYPE = NF	99.41	339	11.95
IF DRYBULB > 8.5 & DRYBULB <= 12 & TOTALCLO > 4 & TOTALCLO <= 7 & PASTWEAT = 2	THEN FOGTYPE = NF	99.34	301	10.61
IF DRYBULB > 8.5 & DRYBULB <= 12 & WINDSPEE > 6.2	THEN FOGTYPE = NF	99.29	279	9.84
IF TOTALLOW <= 1& PASTWEAT = 2 & WINDCOMP = N	THEN FOGTYPE = NF	99.28	274	9.66
IF DEWPOINT > 6 & DEWPOINT <= 9 & TOTALCLO > 4 & TOTALCLO <= 7 & PASTWEAT = 2	THEN FOGTYPE = NF	99.26	270	9.52
IF SEALEVEL <= 1014.1 & WINDSPEE > 6.2	THEN FOGTYPE = NF	99.22	381	13.43
IF WINDSPEE > 6.2	THEN FOGTYPE = NF	98.15	797	28.1
IF HOUR = 18 & TOTALCLO > 4 & TOTALCLO <= 7	THEN FOGTYPE = NF	97.94	238	8.39

Figure 5.24: Association rules from Model1-60, experiment V2

The data mining experiments were replicated using more relaxed levels of rule support and rule order, the next section describes these experiments.

5.4.2.2 Knowledge Models V3

The first data mining experiments, V2, resulted in very restrictive models considering the amount of association rules obtained, specifically describing fog class.

As such, it was decided to repeat the data mining experiments using more flexible mining parameters, aiming to achieve a more equal distribution between the number of rules describing either classes, or at least, to increase the number of rules describing the fog class.

This second set of experiments was conducted using the same mining models used in the previous experiments, V2, keeping the same levels of rule confidence degrees, and relaxing the minimum rule support from 8% to 6%, and allowing a higher number of rule antecedent itemsets, from 7 to 10 itemsets. The minimum amount of cases for each rule remained 50 in all experiments, because this was considered a satisfactory amount, not very restrictive but large enough for a good coverage.

This second set of experiments is identified as **V3.** Table 5.16 shows the number of rules obtained from each data mining model, according to their confidence degrees in this second set of experiments.

Table 5.16: Rule sets by data mining models in experiment V3

Data Mining Model	Generated Rules by Rule Confidence Degree, Rule Support 6%, Maximum Rule Order 10											
	50%			70%			80%			90%		
	F	NF	Total	F	NF	Total	F	NF	Total	F	NF	Total
Mining Model1-60				104	301	405	37	291	328	16	204	220
Mining Model1-80				67	291	358	23	291	314	12	228	240
Mining Model2-60				45	283	328	20	283	303	12	274	286
Mining Model10-60	19	279	298	10	279	289	9	279	288	9	279	288

The best results were obtained with Model1-60 and Model1-80, with 70% rule confidence degree. There were 405 association rules, with 104 allotted to fog class for Model1-60, and 358 association rules, with 67 allotted to fog class for Model1-80. An increase of 50 rules can be observed for fog class in Model1-60 and 32 rules for fog class in Model1-80.

Even with an increase for the amount of rules allotted to the not fog class, these results represent an improvement in terms of descriptive capability for fog phenomenon, compared to the results achieved in the first set of data mining experiment (V2).

Significant changes were not observed in the other experiments. An increase of 15 rules was verified in Model1-60 with an 80% confidence degree, an increase of 13 rules was verified in Model2-60 with a 70% confidence degree, both for fog class. This is certainly an improvement, but its significance is difficult to assess at this stage.

The experiments with a 90% confidence degree did not show significant improvements; an increase of 6 association rules was verified in Model1-60, 4 rules in Model1-80, and only 2 rules in Model2-60 and Model10-60, all for fog class. Model10-60 did not show any improvement at all, with an increase of 2 association rules for fog class, but a much higher number for not fog class. Figure 5.25 compares the number of association rules for **fog class** obtained in both sets of data mining experiments, V2 and V3.

An increase in the number of association rules allotted to the fog class can be verified in this second set of experiments (V3), as a result of applying more flexible thresholds for the mining parameters. The level of 70% rule confidence degree showed the best results, with 104,

67 and 45 association rules obtained for each data mining model, with the exception of Model10-60.

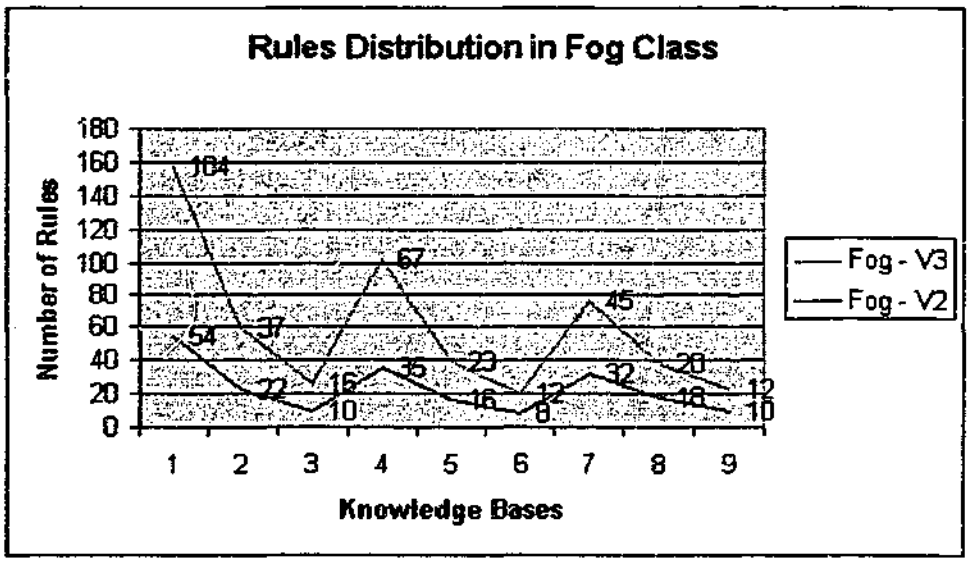

Figure 5.25: Rules distribution in Fog Class, in experiments V2 and V3

Different threshold settings for mining parameters could be used to conduct new experiments, however this approach would lead to a large number of new experiments, demanding extra time. The results obtained at this stage were considered sufficient to assess the performance and feasibility of the DM-NN model proposed in this research. Specifically, the numbers of association rules in fog class that were obtained as a result of the new experiments (V3) were considered satisfactory to populate the knowledge bases.

5.4.2.3 The Knowledge Bases

As a result of the data mining experiments, several distinct sets of association rules (similar to that shown in Figure 5.24) were obtained. Each of these distinct sets of association rules populates a particular *knowledge rule base*, and Figure 4.1 illustrates this process.

The knowledge bases were further applied as training datasets for the neural network model, within the CANN system. As such they had to be properly identified. For this a specific naming schema was developed in order to identify the knowledge bases according to the strategy used to generate them, i.e., original data mining models, levels of rule confidence and support degrees, and maximum rule order.

The knowledge bases naming schema is as follows:

Train_ModelX-XVmnn

where:

ModelX-X: identifies the original data mining model from where this knowledge base was generated, e.g., Model1-60, Model1-80, Model2-60, or Model10-60.

Vm: identifies data mining experiment, 2 or 3, according to the configuration of rule support and rule order, as follows:

V2: identifies a minimum rule support of 8% and a maximum rule order of 7

V3: identifies a minimum rule support of 6% and a maximum rule order of 10

nn: identifies the data mining rule confidence degree (50%, 70%, 80% or 90%).

For example; *Train_Model1-60V270*

identifies a knowledge base originating from data mining Model1-60, in an experiment using a rule minimum support of 8%, a maximum rule order of 7 itemsets (V2 configuration), and a rule confidence degree of 70%.

All the knowledge bases obtained are now listed, identifying their original data mining model, respective name, amount of rules, and the amount of rules in each class. The knowledge bases were further used as the training datasets in the Intelligent Advisory System (refer to section 4.3.5) implemented through the CANN simulator (refer to section 4.5).

Table 5.17 to Table 5.20 list the knowledge bases, organized by their respective data mining models. It should be noted that not all sets of association rules were used to populate the knowledge bases. In particular the association rules obtained through Model10-60, with 70%, 80% and 90% rule confidence degrees, were not selected, because of the small number of rules describing fog class. The significant difference between the rules allotted to both classes, together with the small number of rules allotted to the fog class indicates a low descriptive capability in these models.

Table 5.17: Knowledge bases from Model1-60

	Knowledge Base	Number of Rules	Number of Fog Rules	Number of Not Fog Rules
1	Train_Model1-60V270	240	54	186
2	Train_Model1-60V280	202	22	180
3	Train_Model1-60V290	132	10	122
4	Train_Model1-60V370	405	104	301
5	Train_Model1-60V380	328	37	291
6	Train_Model1-60V390	220	16	204

Table 5.18: Knowledge bases from Model1-80

	Knowledge Base	Number of Rules	Number of Fog Rules	Number of Not Fog Rules
7	Train_Model1-80V270	215	35	180
8	Train_Model1-80V280	196	16	180
9	Train_Model1-80V290	136	8	128
10	Train_Model1-80V370	358	67	291
11	Train_Model1-80V380	314	23	291
12	Train_Model1-80V390	240	12	228

Table 5.19: Knowledge bases from Model2-60

	Knowledge Base	Number of Rules	Number of Fog Rules	Number of Not Fog Rules
13	Train_Model2-60V270	209	32	177
14	Train_Model2-60V280	195	18	177
15	Train_Model2-60V290	179	10	169
16	Train_Model2-60V370	328	45	283
17	Train_Model2-60V380	303	20	283
18	Train_Model2-60V390	286	12	274

Table 5.20: Knowledge bases from Model10-60

	Knowledge Base	Number of Rules	Number of Fog Rules	Number of Not Fog Rules
19	Train_Model10-60V250	187	17	170
20	Train_Model10-60V350	298	18	298

These final sets of rules, identified from Table 5.17 to Table 5.20, were applied as training sets in the neural network model CNM, implemented through the CANN simulator.

Although not specifically relevant to the purposes of performance of the DM-NN model, but relevant to the construction of the computational environment developed in this research, it has to be mentioned that a complex set of procedures were required to transform the files generated through the data mining process into the final files used by CANN.

The knowledge bases shown in Tables 5.17 to 5.20 were generated as MS Access tables, and then exported as ASCII plain files, in order to be read by CANN.

This process demanded a set of operations, such as, to properly transform the numerical attributes into their corresponding categorical values, according to what is presented in Table 5.13. It can be observed in Figure 5.26 below, for example, that the *wind speed* observation in rule 23 (the first rule at the top of the list) is given as *light* instead of its original value, in this case a numerical measure less or equal 1.5 meters/second.

In order to be interpreted by CANN, the weather observations also had to be adjusted according to their sizes and positions in the files. CANN reads data in training and testing datasets according to their respective positions in these files (this is discussed in Chapter 6).

T	HR	MT	DRYB	DEWP	PRDW	CLD	LCL	SEAPR	WINDS	COM	R	PW	TW	VISI	RULE
F						MAX		VHIGH	LIGHT						23
F			LOW			MAX			LIGHT						24
F			LOW				MAX	VHIGH	LIGHT						25
F			LOW	HIGH					LIGHT		0				26
F						MAX	MAX		LIGHT						27
F						MAX				CLM	0				28
F						MAX	MAX	VHIGH							29
F			LOW				MAX	VHIGH			0				30
F	9		LOW					VHIGH							31
F						MAX	MAX		LMODE						32
F	9							VHIGH	LIGHT						33
F						MAX		VHIGH							34
N					HIGH	MED					0		2		58
N						MED				N			2		59
N						MED	MED		FMODE						60
N						MED		LOW	FMODE						61

Figure 5.26: Association rules in Train_Model1-60V270

Figure 5.26 illustrates various rules from the training set Train_Model1-60V270. The first row (header) of the table identifies the weather observation, and is presented here to help

understand the rules displayed in the figure, but it is not included in the training and testing ASCII files.

For instance, the header in Figure 5.26 identifies the first column as "T", which indicates a fog case (F) or a not fog case (N). This column does not exist in the test datasets. Then "HR" identifies the hour observation, "MT" identifies the month, "DRYB" identifies the dry bulb, "DEWP" the dewpoint and "PRDW" the previous dewpoint, and so on.

For example, the top rule, number 23, indicates a fog case and can be read as:

If *CLD = MAX* *And*
SEAPR = VHIGH *And*
WINDS = LIGHT *Then*
Class = FOG.

Where: CLD = cloud amount
SEAPR = sea level pressure
WINDS = wind speed

Comparing the rules in Figure 5.26 with the rules in Figure 5.24 a significant difference in their formats can be noticed. Also the confidence and support measures are disregarded in the CANN simulator, these measures were used to filter and select rules during the data mining process and were not necessary afterwards. The CANN simulator employs neural network weights and morbidity values to assign value measures to data. This is discussed next in Chapter 6.

Chapter 6 describes the application of the CANN simulator to aviation weather forecasting, which comprises the last part of the application of the DM-NN model for decision support developed in this research. The CANN simulator is identified as the Intelligent Advisory System component, as it is responsible for the implementation of learning, reasoning and explanatory capabilities.

5.5 Chapter Summary

This chapter described the stage of knowledge discovery in the application of the Hybrid DM-NN model in aviation weather forecasting at Tullamarine. First, the problem of aviation weather forecasting was introduced and its suitability for this research was discussed. Next, the stage of discovering knowledge from a meteorological database and building knowledge bases

were described. This included the tasks of problem domain understanding, data gathering, selection, and data preparation for data mining. The tasks of features selection, analysis of missing values and variability, and data transformation were also described. Furthermore, this chapter presented the data sampling approach that was adopted, and the data mining process that was developed.

Finally, this chapter presented the process applied in generating knowledge bases and described the knowledge bases obtained.

Chapter 6

6 Applying the DM-NN Model in Aviation Weather Forecasting: The Intelligent Advisory System Stage

The Intelligent Advisory System (IAS) is the component responsible for implementing learning, reasoning, and issuing recommendations in the DM-NN model for decision support. The IAS component is implemented within a neural network environment – the CANN simulator employing the CNM neural network model.

This chapter describes the implementation of the CANN simulator in aviation weather forecasting. Issues of aviation weather forecasting domain modelling into CANN simulator are described. The CNM learning stage is discussed in the context of aviation weather forecasting, and the user interface functionalities are presented.

6.1 Introduction

The DM-NN model applies artificial neural network technology (NN) for the purposes of processing the knowledge discovered through data mining, implementing learning and reasoning upon this knowledge. The reason for this is because neural networks excel in learning, self-organization, and generalization capabilities, as discussed in Chapter 2. Section 2.2.3, "Intelligent Systems Capabilities" addresses the issues of implementing learning and reasoning capabilities in intelligent systems, and section 2.2.4 discusses the role of specific computing technologies in building intelligent systems. Section 2.4 specifically discusses artificial neural networks.

The component responsible for the implementation of learning, reasoning, issuing recommendations and explanations in the DM-NN model developed in this research is

identified as the Intelligent Advisory System (IAS). Section 4.3.5, "Intelligent Advisory System" discusses the IAS component, its roles and functionalities.

In the research described in this thesis the IAS component is implemented within the CANN simulator environment and the CNM neural network model.

The Combinatorial Neural Model (CNM) was selected as the neural network model in this research not only for its generalization, learning from examples, and self-organizing capabilities, but also because of its compatibility with the knowledge representation formalism of knowledge graphs and association rules. Section 4.4 discusses the knowledge representation schema employed in the DM-NN model, and the CNM NN model is introduced in section 4.6 of that chapter.

Besides implementing learning and reasoning capabilities, another goal of the DM-NN model is to issue recommendations and explanations. For this, the CANN simulator was selected. CANN implements a computer system architecture that combines NN models with a symbolic mechanism to represent NN structures. NN structures, including the knowledge stored across these structures, are symbolically represented in a hierarchical fashion, through an object-oriented design, thus minimizing the problem of implicit knowledge representation in neural networks models.

This chapter describes the application of such an IAS component in aviation weather forecasting. Firstly, the domain of aviation weather forecasting is modelled through a hierarchy of classes. Next, this hierarchy is integrated into the CANN simulator. Once the domain is modelled, CANN accesses the respective knowledge bases (refer to section 5.4.2.3, "The Knowledge Bases") for learning purposes.

Learning is implemented through the CNM learning algorithms, and after the learning is completed the system is ready for consultation, which means testing hypotheses about the specific domain it has been applied to. CANN offers two ways for user interaction, a case consult and a case base consult. In a case consult the user selects a set of evidences and the system advises the hypothesis that is most likely to occur according to the selected group of evidences. The case base consult works in a similar fashion, however, instead of a single case, many cases can be simultaneously presented to CANN for evaluation.

Figure 6.1 illustrates the intelligent advisory system stage from the perspective of the decision support cycle implemented through the DM-NN model, previously discussed in Chapter 4. The activities depicted in Figure 6.1 were developed and discussed thoroughly in that chapter in the context of aviation weather forecasting.

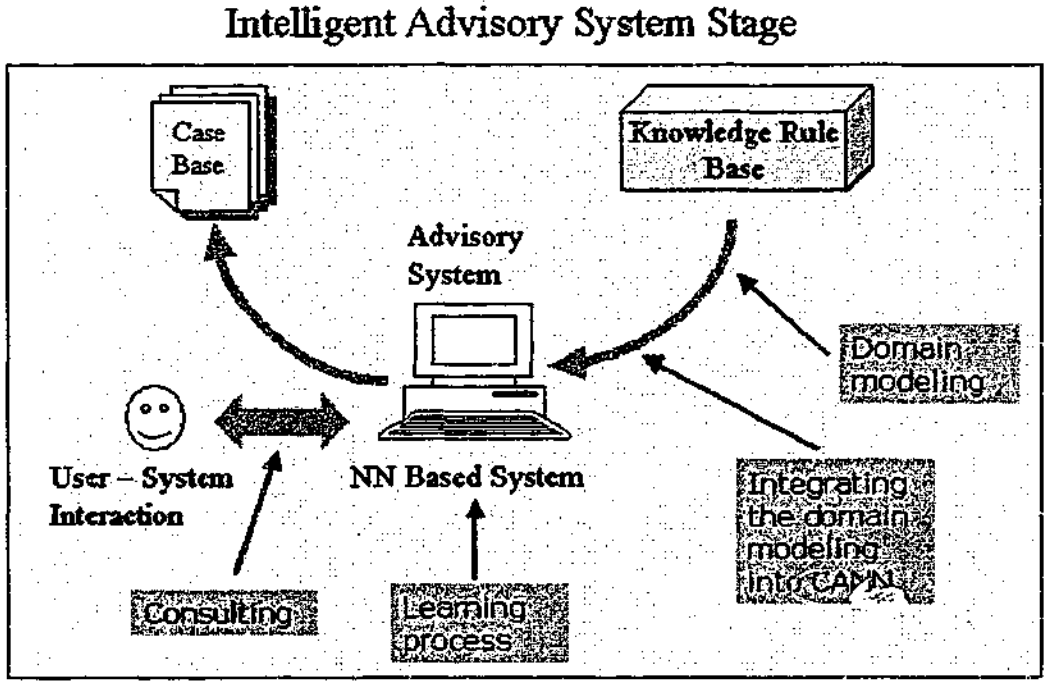

Figure 6.1: The intelligent advisory system stage

This chapter follows on to illustrate the application of one of the generated data models (see Chapter 5) within the IAS component. The stages of domain modelling and integration into the CANN simulator, learning, and consulting are presented, and are illustrated in Figure 6.1.

It has to be emphasized that it is not the purpose of this thesis to deeply discuss implementation and design issues of the CANN simulator, but only to cover issues that are relevant to the understanding of the research developed in this thesis. For further discussions about the CANN simulator refer to (Beckenkamp, 2002), and for the CNM neural network model refer to (Machado, Barbosa and Neves, 1998; Machado and Rocha, 1989).

Additionally, this thesis also assumes that readers are familiar with concepts of object orientation, frameworks and design patterns. These concepts are necessary to understand the subjects discussed in this chapter. Readers interested in further discussion in these concepts should refer to (Pree, 1995) and (Gamma et al., 1995).

6.2 Modelling the Domain of Aviation Weather Forecasting

The knowledge representation schema employed in the DM-NN model has its grounds in the knowledge representation formalism of knowledge graphs (KG). A KG has its structure defined to represent knowledge in classification problems.

Essentially, there are three types of nodes in a KG: *hypothesis, evidence* and *intermediate.* Hypothesis nodes represent the hypotheses, or classes, considered in the graph; evidence nodes represent input information that support a particular hypothesis, and intermediate nodes represent different groupings of evidences that lead to a specific hypothesis.

Section 4.4, "The Knowledge Representation Schema" discusses the various knowledge representation formalisms employed in the DM-NN model. At the data mining level knowledge is represented through association rules. At the CANN level knowledge is represented in two ways: implicitly stored in the neural network topology and through an object-oriented design that reflects common properties of classification problems.

Therefore, the domain of aviation weather forecasting is modelled by a hierarchy of classes and objects describing the domain classes, evidences, attributes and the interrelationships among them. Through the use of the abstraction concepts (Hull and King, 1987) of generalisation, classification and aggregation the domain knowledge can be represented in such a hierarchy.

Figure 6.2 illustrates how the aviation weather forecasting domain was modelled according to this approach. The main classes are *Domain*, *Evidences*, *Attributes* and *Classes*, which correspond to nodes in the KGs. Specifically, hypothesis nodes are presented by instances of *Classes*, evidence nodes are represented by instances of *Evidences* and *Attributes*, and intermediate nodes are represented by aggregation between instances of *Evidences/Attributes* and *Classes.*

An important distinction has to be made about the concepts of *Classes.* The object-oriented concept of *class* must not be confused with the classificatory concept of *class.* In this last case, a class indicates the possible classification patterns in a certain domain, which corresponds to the hypotheses nodes in KGs and output nodes in the CNM neural network model topology.

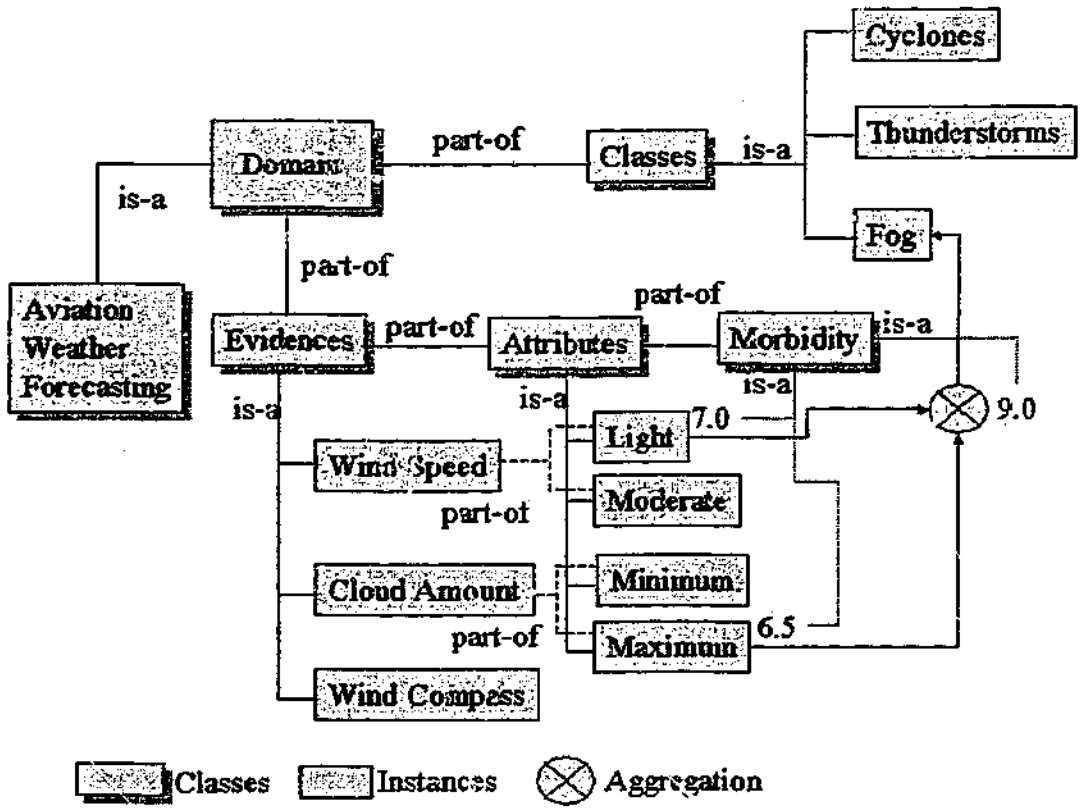

Figure 6.2: The aviation weather forecasting domain model

In Figure 6.2, *Domain* is the higher level class, from where different application domains can be subclassed, through "is-a" relationships. In this case, the Aviation Weather Forecasting Domain is created as a subclass of *Domain*. The *Domain* class is connected by "part of" arcs to *Evidences* and *Classes*, implying that the *Domain* should be made by the composition of these two classes. Therefore, the Aviation Weather Forecasting Domain is made by the composition of instances of *Classes* and *Evidences*.

Instances of *Classes* (again, it is important to make the distinction between the object-oriented definition of Class, and the definition of a class as part of a classification problem) in the Aviation Weather Forecasting Domain are weather phenomena such as *fog, thunderstorms*, and *cyclones*.

Following Figure 6.2, *Evidences* is connected to *Attributes* by a "part of" relationship, implying that *Evidences* instances are composed of instances of *Attributes*. In Figure 6.2, instances of *Evidences* class are *Wind Speed, Cloud Amount* and *Wind Compass*, connected to the *Evidences* class by an "is-a" arc. Similarly, instances of Attributes class are *Light* and *Moderate*

that are part of the *Wind Speed* evidence. In the same way, attribute instances *Minimum* and *Maximum* are part of the *Cloud Amount* evidence.

Associations between *Evidences/Attributes* that together lead to instances of *Classes* are represented by the concept of aggregation. In Figure 6.2, for example, the aggregation of the evidence/attribute pairs *Wind Speed Light* and *Cloud Amount Maximum* leads to *fog*.

Pairs of evidences/attributes have a morbidity value assigned to them. Morbidity is a value that indicates the importance of a particular pair evidence/attribute related to instances of *Classes*. Similarly, aggregations have a morbidity assigned to them, which indicates the importance of a particular evidence/attribute association related to the class indicated by that association.

In Figure 6.2, the value of 7.0 (in a scale from 0 to 10) assigned to the evidence/attribute pair of *Wind Speed Light* means that this evidence/attribute pair has a significant importance in classifying fog cases. In a similar fashion, the value of 9.0 assigned to the association between *Wind Speed Light* and *Cloud Amount Maximum* indicates that this association strongly contributes to a positive case of fog.

The morbidity is modelled as a class connected to the *Attributes* class through "part of" arcs. The concept of morbidity to measure the importance of evidences in classification problems used in this thesis is based in the morbidity scale used by the Internist-I system (Miller, 1986) and later by the hybrid case base reasoning model developed in (Reategui, 1997).

In the KGs, morbidity corresponds to the degree of importance assigned to the evidence nodes (refer to section 4.4 and also Appendix A). Morbidity can also be assigned as weight values into the input neurons in the CNM neural network model, helping the model to more easily converge to a stable state.

Through this modelling schema the domain of aviation weather forecasting can be integrated into the Domain framework implemented in CANN, as introduced in section 4.5.1.2, "Domain Representation."

6.3 Integrating the Domain Model of Aviation Weather Forecasting into the CANN Simulator

The Components for Artificial Neural Networks Simulator (CANN) is a research project that relates to the design and implementation aspects of a framework architecture for decision support systems using artificial neural network technology (Pree, Beckenkamp and Rosa, 1997; Beckenkamp, 2002). Besides the creation of NN components, CANN also supports problem domain modelling.

CANN was initially developed on ideas and applications of NN in classification problems (Pree, Beckenkamp and Rosa, 1997). Thus, NN structures (including the knowledge stored across these structures) are symbolically represented in a hierarchical fashion, through an object-oriented design that reflects common properties of classification problems. In CANN, the domain is represented through four main classes: *Domain*, *Evidence*, *Hypothesis*, and *Attributes*, as illustrated in section 4.5.1.2, "Domain Representation.")

An instance of class *Domain* represents the problem by managing the corresponding *Evidence* and *Hypothesis* objects. The *Domain* class manages hypotheses and evidences associated with a particular domain; and both evidences and hypotheses are described by their respective attributes.

Therefore, the first step in applying the CANN simulator is to integrate the domain of aviation weather forecasting into the CANN modelling schema.

Each previously built data model, describes in section 5.3.4.2, "Generating Data Mining Models," has to be individually modelled into the CANN simulator. The reason for this is that they present different amounts of cases, having specific test sets and knowledge bases (training sets) according to the tables listed in section 5.4.2.3, "The Knowledge Bases." Consequently, for each data model a correspondent domain model was designed, thus, Domain Model1-60, Domain Model1-80, Domain Model2-60 and Domain Model10-60. The domain models might have different amounts and pairs of evidences and attributes, and distinct morbidity values assigned to them. Appendix D lists the domain model for Model1-60, including its evidence list.

Domain Model1-60 is used to explain and illustrate how the CANN simulator was applied in this research, and the other models were applied in a similar fashion. To illustrate this, the components of GUI framework implemented in CANN are employed. The framework's components implemented in CANN were introduced in section 4.5.1.

6.3.1 Domain Elements

The first step in working with the CANN simulator is to define a domain, and create a project associated to that domain. This project will contain the entire domain modelling information.

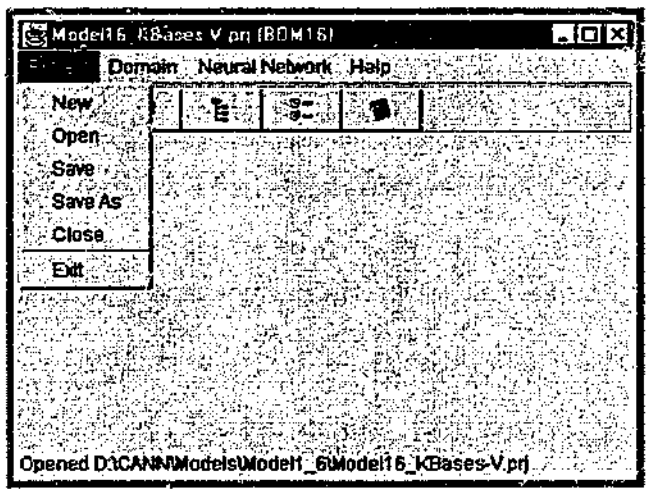

Figure 6.3: Creating a project for domain Model1-60 in CANN

Figure 6.3 shows a project for Domain Model1-60 being created; this project is identified as "BOM16_Kbases-V.prj" at the top of the window. It also shows, between parentheses, the name of the domain model.

Four domain models were created in this research project, *Domain Model1-60*, *Domain Model1-80*, *Domain Model2-60* and *Domain Model10-60*. For each domain model distinct projects were created according to the datasets used for training.

Domain creation is done through the Domain interface (GUI component). The following procedures are executed through the Domain interface:

- Selecting a Domain. Domain instances can be created and selected to manipulate hypotheses and evidences
- Creating and editing evidences

- Creating and editing hypotheses
- Choosing the type of data source, e.g., a database table or an ASCII file
- Selecting the data source for learning and testing, based on the chosen data type

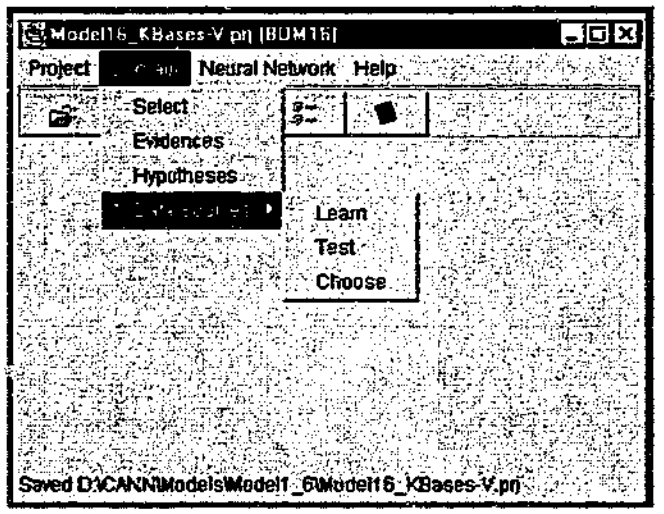

Figure 6.4: The functionalities of Domain GUI

Figure 6.4 shows the functionalities implemented in the Domain GUI component.

The first step is to select or create a domain model, which is done through the *Domain-Select* menu, the first option under Domain dialog. In this option a new Domain is created simply by typing its name and clicking the *Add* button. This domain is then listed at the top of the dialog list of domain names. Next, the evidence and hypotheses associated to that Domain instance have to be created.

6.3.2 Modelling Evidences

Evidences[1] are used by experts to analyze a certain problem in order to arrive at decisions. Evidences in aviation weather forecasting are weather observations, as listed in Table 5.1, in Chapter 5. As such, modelling evidences in this domain involves finding which weather observations contribute to the hypotheses of that domain, to determine the importance of each evidence (evidence/attribute pair), i.e., morbidity, as previously discussed in section 6.2. And finally, modelling evidences requires the specification of the respective position of each evidence in training and testing files.

[1] *Evidences* here are used in a broader context, meaning pairs of evidences and attributes.

The first step is to select the evidences, e.g. weather observations, which contribute to the classes (hypotheses) of the problem domain under study.

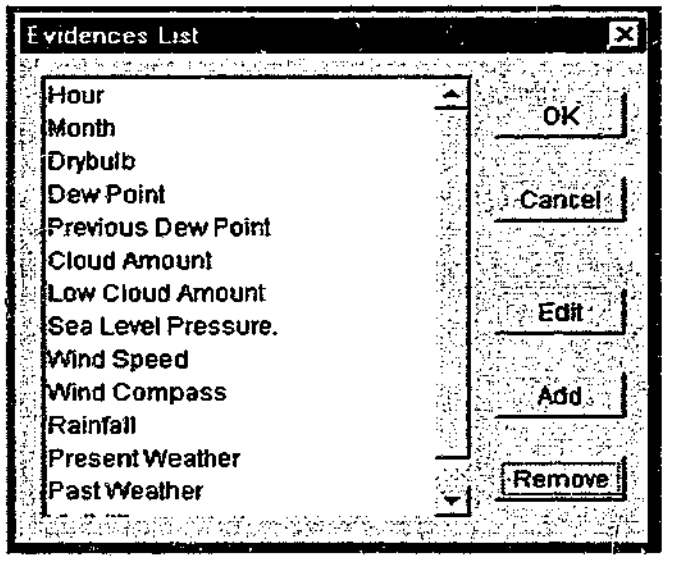

Figure 6.5: Evidence list in the domain of aviation weather forecasting

Figure 6.5 shows a snapshot of the Evidence list dialog. The relevant weather observations are integrated in the CANN simulator by inserting them into the Evidence list.

Next, for each evidence, the respective set of attributes, morbidity values, and file positions have to be specified. For example, considering the *Wind Speed* evidence, the attributes of *light*, *light moderate*, *moderate* and *fresh moderate* need to be inserted. Figure 6.6 illustrates the *Wind Speed* evidence and its attributes.

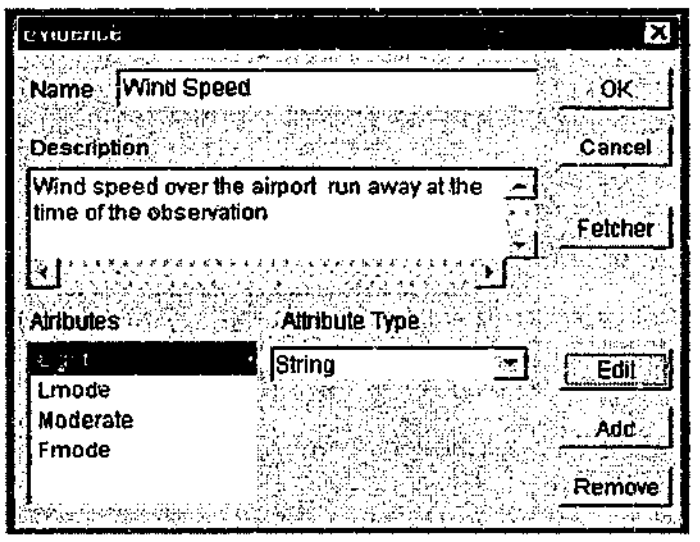

Figure 6.6: Wind Speed evidence and its attributes

Next, a morbidity value has to be assigned to all pairs of evidences/attributes. Morbidity is a measure allocated to each evidence/attribute pair, as already discussed in section 6.2. In the same way KGs require a morbidity value assigned to each evidence/attribute pair, the CANN simulator also requires that information when entering evidences and attributes.

Normally, morbidity is assigned by domain experts in a range of 0 to 1. In the case of this research a specific method had to be implemented to calculate the morbidity. A method based on the computation of a weighted measure of frequency of evidence/attributes for each data model was employed to compute the morbidity. This method was based on the computation of sensibility degrees of the evidences observed in training datasets (Owens and Sox, 1990).

A variable β is used to weigh the frequency measures of evidences. It is used to give an increasing degree of importance for evidences that have been recognized as highly relevant, or the ones that have lower frequency values, but are relevant to the problem under consideration.

The morbidity values are calculated through the following equation:

$$Morb\,(Ev) = (\,freq\,(Ev)/n) * \beta$$

Equation 6.1: Morbidity function

Where:

Morb(Ev) indicates the morbidity value for the evidence *Ev*

Freq(Ev) is the frequency measure of the evidence *Ev* in the dataset

N is the amount of cases in the dataset

β is the weighting variable, usually equal to 1.7

The value 1.7 assigned to the variable β was empirically determined through a set of experiments. It was shown to be sufficiently accurate in most experiments, when discussed with domain experts.

Figure 6.7 illustrates the morbidity value assigned to the evidence *Wind Speed Light*. A morbidity value has to be assigned for all evidence/attribute pairs in all domain models. Identical evidence/attribute pairs might have different morbidity values assigned to them in

different domain models. This is because the morbidity calculation method employed in this research is based on the measure of frequency of each evidence/attribute pair in the dataset.

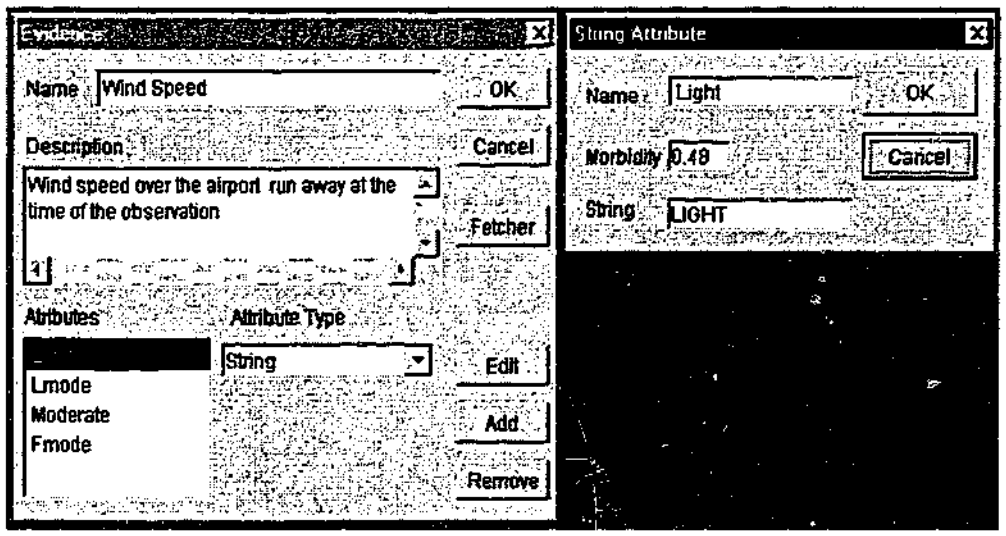

Figure 6.7: Assigning morbidity to Wind Speed Light

As illustrated in Figure 6.7, a value of 0.48 was assigned as the morbidity to *Wind Speed Light* evidence. The *Edit* button in the Evidence dialog is used to assign the morbidity measure.

The next information that has to be specified is the type of data source used for training and testing, i.e., database or ASCII files. In this research the data sources are ASCII files (at the time this thesis was being written only the Fetcher for reading data from ASCII files was implemented in CANN. The Fetcher for reading data from relational databases was not fully available for deployment). Information in ASCII files is identified by its position in the file, i.e. the columns it occupies. As such, a data fetcher for reading data from ASCII files stores the position, from column to column, of the data. Each line of the file stores one case.

Therefore, for each evidence its file position has to be informed to the CANN simulator. Figure 6.8 illustrates the fetcher for the *Wind Speed* evidence.

The button *Fetcher* in Figure 6.8 is used to specify the evidence position in the files. The data in an ASCII file must be organized as records, where each record corresponds to one line in the file. As shown in to Figure 6.8, the *Wind Speed* evidence (and its respective attribute instances) is stored between the column 28 to 32, inclusive, in the training and testing datasets.

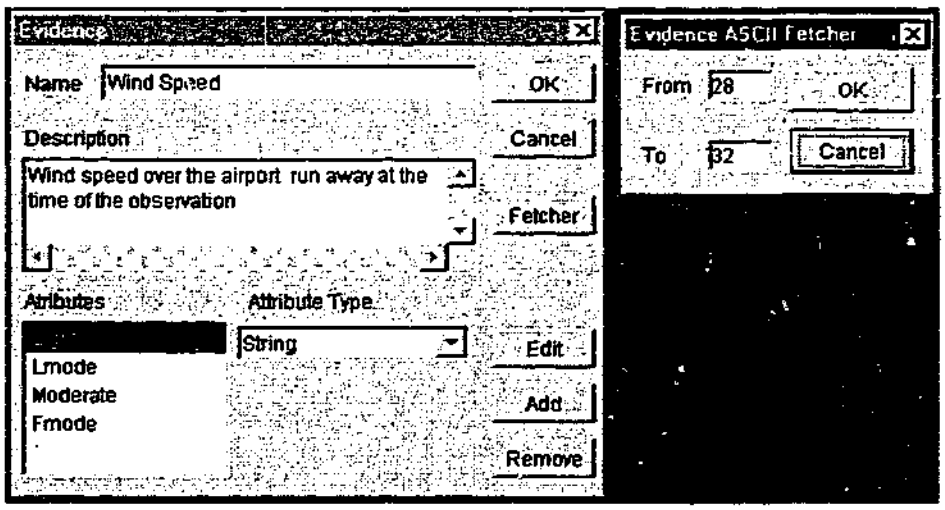

Figure 6.8: Fetching the Wind Speed evidence

Table 6.1 summarizes the information discussed so far in this section, exemplified through the *Wind Speed* evidence. The same information is provided for all evidence instances in all domain models. The complete domain modelling for Model1-60 is included in Appendix D. The other models follow the same structure.

In Table 6.1, the "Fetcher" column indicates the position in the ASCII files (both for training and testing) where the wind speed is stored. In this case, it starts at column 28, is five characters long, and ends at column 32.

Table 6.1: Wind Speed modelling for domain Model1-60

Evidences	Fetcher	Values	Frequency	Percent	Morbidity
Wind Speed	[28; 32]				
		LIGHT	1037	28.47	0.48
		LMODE	852	23.39	0.40
		MODE	940	25.80	0.44
		FMODE	812	22.29	0.38

The "Values" column in Table 6.1 specifies the possible values that can be assigned to *Wind Speed* (refer to Table 5.13, for the discretization of numerical attributes for Model1-60). The wind speed can assume values of *light*, *light moderate* (lmode), *moderate* (mode), and *fmode* (fresh moderate), as illustrated in Figure 6.6.

The "Frequency" column indicates the frequency measured of the *Wind Speed* in the dataset. For instance, *Wind Speed Light* was verified in 1037 cases in the Model1-60 dataset. The "Percent" column indicates the percent of the frequency in the dataset; *Wind Speed Light* was verified in 28.47% of cases in the Model1-60 dataset. This information is necessary to

compute the morbidity, which is specified in the "Morbidity" column. The morbidity for *Wind Speed Light* is 0.48, as a result of the morbidity calculation presented in Equation 6.1.

Once the integration of evidences has been performed, the hypotheses have to be integrated into the domain.

6.3.3 Modelling Hypotheses

The classification categories, e.g. hypotheses, constitute a further core entity of classification problems, and are modelled through the *Hypothesis* class into the CANN simulator.

Hypotheses can be created and edited in a similar fashion as evidences. There is a list of hypotheses that is similar to the list of evidences in Figure 6.5. The dialog for creating and editing hypotheses is showed in Figure 6.9. This dialog has nearly the same functionalities as for the evidences, specifying their respective fetcher (positions) and attributes. There is also a list of related evidences. This list defines which evidences have to be considered in the creation of the NN topology during the training phase, given a certain hypothesis. Normally all the modelled evidences are associated to all hypotheses, but there might be cases where a certain evidence contributes to a certain hypothesis and not to others. Frequency measure is a possible way to approximately estimate this information.

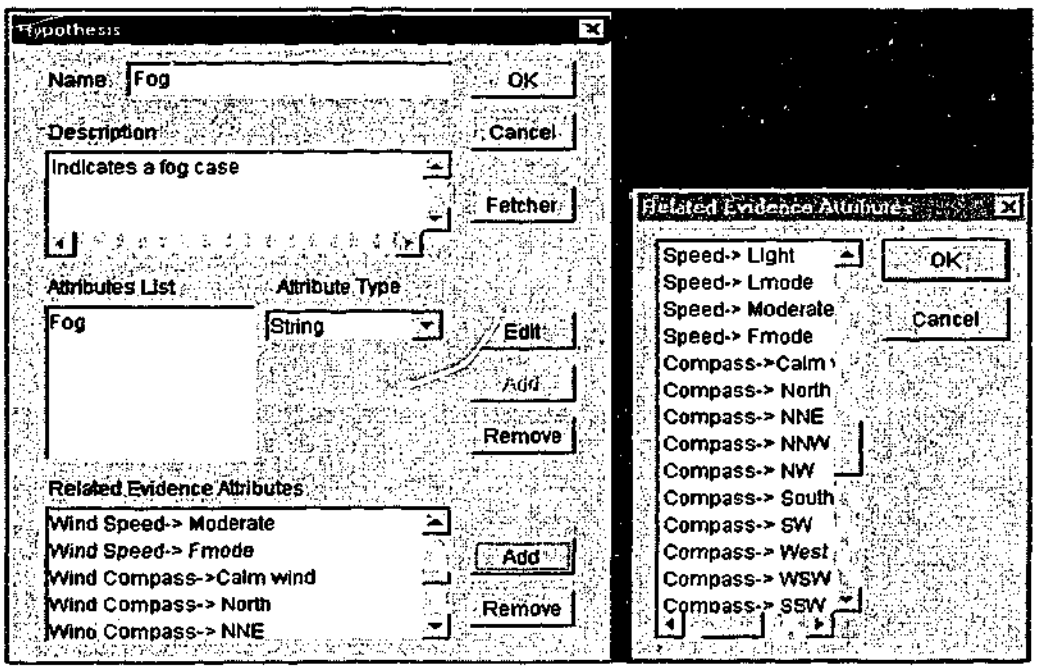

Figure 6.9: Modelling Fog hypothesis

In the study developed in this research, identifying fog occurrence at Tullamarine, two hypotheses were modelled: *fog* and *not fog*. Fog hypothesis is illustrated in Figure 6.9; it has one attribute of string type that has the same name as the hypothesis, e.g., *Fog*. Figure 6.9 also shows part of the list of evidences associated to the *Fog* hypothesis. For example, it shows that the *Wind Speed Moderate* and *Fmode*, together with *Wind Compass Calm*, *North* and *NNE* were selected in the creation of the NN topology related to the *Fog* hypothesis. On the righthand side of this figure there is a list containing all evidences created in the project, from where evidences can be chosen and associated to a hypothesis, or removed (through the *Add* and *Remove* buttons). By selecting or removing evidences in a particular problem it is possible to create different domain models in a unique project.

By associating evidences with input neurons, and hypotheses with output neurons, CANN builds NN topologies.

6.4 The Learning Process

Once the problem domain is modelled into the CANN simulator, the system is ready to proceed with the learning process. For this the training datasets have to be presented to the chosen NN model (the CNM model in this research). All knowledge bases listed in Chapter 5 were presented to CANN for learning, according to their respective domain models: Domain Model1-60, Domain Model1-80, Domain Model2-60 and Domain Model10-60.

6.4.1 The CNM Learning Mechanism

The learning process is executed through the CNM learning algorithms: the Starter Reward and Punishment (SRP) and the Incremental Reward and Punishment (IRP). The CNM learning mechanism was introduced in section 4.6 of this thesis.

Basically, the learning mechanism implemented by CNM is a supervised learning approach, which determines the combinations of evidences that are influential for each class (hypothesis).

The learning approach implemented by CNM is based on the concept of rewards and punishments, analogous to that of the Backpropagation model, to identify successful and unsuccessful pathways. Synapses defined in CNM topology have weights and a pair of

accumulators for rewards (R_{ACC}) and punishments (P_{ACC}). At the beginning all weights are set to one and all accumulators to zero. During the training phase, as each example in the training set is propagated along the network, pathways that lead to correct classifications have their reward accumulator incremented. Similarly, misclassifications increment the punishment accumulators.

This learning approach takes into account the evidential flow when determining the values of punishments and rewards. The evidential flow observed in a connection corresponds to the following product, for excitatory synapses (Reategui, 1997):

- destination-node activation multiplied by connection weight

For inhibitory connections, the product is:

- (1 - destination-node activation) multiplied by connection weight

Weights remain unchanged during the training process. At the end of the training phase, which is done with a single scan over the training set, pathways with more punishments than rewards are pruned, and the remaining connections have their weights calculated.

The IRP and SRP algorithms compute the punishment and reward accumulators. The SRP algorithm is used to initialise the network and calculate the punishment and rewards accumulators according to the training set. The accumulators are set to zero and the weights are set to one. All data examples are applied to the network in a single scan, to update the rewards and punishment accumulators. After scanning the training set and calculating the accumulator values, no-rewarded pathways are pruned and the remaining pathways have their weights updated. The result of this process is a trained network that stores punishment and reward accumulators in the interval $[-n_e,+n_e]$, where n_e is the number of training examples presented to the network. This process generates a non operational network; to become operational the network has to execute the pruning and normalisation algorithm, which converts the accumulators into connection weights.

The IRP algorithm adjusts the knowledge (accumulators and weights values) of the network. The IRP algorithm updates the punishment and reward accumulators, considering the values previously calculated by the SRP algorithm, removing all negative or weak connections. Weak connections are those with weights smaller than a predefined pruning

threshold that is empirically determined by the user. Again, the accumulator values have to be processed by the normalisation algorithm to make the neural network operational. The normalisation process converts the net values of the accumulators, the difference between rewards and punishment, into connection weights between 0 and 1.

Furthermore, the IRP algorithm identifies and keeps all pathognomonic pathways (those with no punishment and a positive reward value), setting their respective weights with values higher than the pruning threshold.

Normally the pruning threshold in the CNM neural network is empirically verified, through the systematic testing of the network with different threshold values.

More detailed explanations about the CNM learning algorithms can be found in (Denis and Machado, 1991; Machado, Barbosa and Neves, 1998) and (Beckenkamp, 2002).

6.4.2 Training CNM with Association Rules

To illustrate the learning approach implemented by the CNM and how it operated in the experiment conducted in this research, an example extracted from the domain Model1-60 is now presented.

Let us consider some association rules extracted from the knowledge base identified as *Train_Model1-60V370* (refer to Chapter 5 for a description of knowledge bases and their naming schema).

Table 6.2: Association rules from Model1-60

Rule number	Class	Weather Observations (Evidences)
17	Fog	Dry Bulb Low (e1) Cloud Amount Max (e2) Sea Level Pressure Vhigh (e3)
25	Fog	Dry Bulb Low (e1) Cloud Amount Max (e2) Wind Speed Light (e4)
246	No Fog	Dry Bulb High (e5) Previous Dew Point High (e6) Cloud Amount Med (e7)
249	No Fog	Dry Bulb High (e5) Cloud Amount Med (e7) Sea Level Pressure Med (e8)

Table 6.2 shows four rules extracted from the rule set *Train_Model1-60V370*, indicating fog and not fog class.

For example, rule number 17 says:

If *Dy Bulb* = *Low* *And*
Cloud Amount = *Max* *And*
Sea Level Pressure = *Vhigh*
Then Class = *Fog*

When training the neural network with these four rules, eight distinct input neurons (each input neuron is associated to an evidence) are instantiated and linked to different combinatorial nodes, e.g., synapsis instances (refer to section 4.5 for discussion about the CANN object model).

The synapses are connected to hypotheses (classes) and a synapse might have one or more input neurons associated to it, depending on the number of the NN combination order. In this example, a maximum of three input neurons are connected to a synapsis.

The SRP algorithm creates a fully connected non operational network topology, linking all evidences and clusters of evidences to synapses and then to hypotheses. (The SRP algorithm implemented in CANN was optimised in order to reduce the number of combinations when the neural network is first created (Beckenkamp, 2002)).

Figure 6.10[1] shows the CNM network built according to the rules listed in Table 6.2. Considering only fog rules, evidences *e1* to *e4* are represented in the figure (representing all eight evidences will result in an incomprehensible drawing as the number of synapses will be too many). For reason of better visualization the network was split into two drawings, one representing the topology associated to fog class (F) and the other to not fog class (NF). The input neurons are the same for both topologies. Although only the evidences supporting fog class are illustrated, the initial neural network topology considers both fog class and not fog class.

After the initial network is built the SPR is then used to train the neural network through the training data, in this example using the rules listed in Table 6.2. The neural network computes punishment and rewards for fog and not fog class.

[1] Figures 6.10 and 6.11 were adapted from (Reategui, 1997).

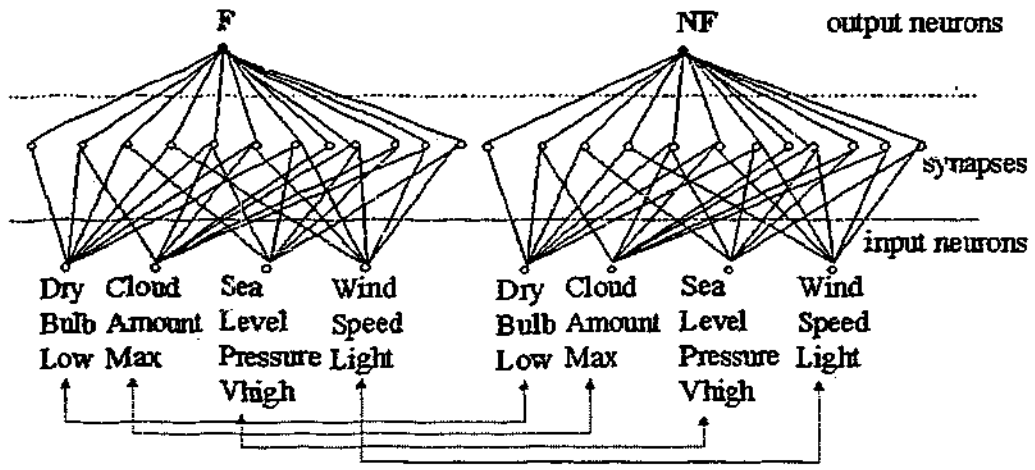

Figure 6.10: The initial non operational CNM neural network

For instance, when presented with the first rule (number 17), the connections (synapses) activated by the evidences *e1*, *e2* and *e3* that lead to the correct hypothesis (fog) are rewarded by the SRP algorithm, and the connections that lead to an incorrect hypothesis (not fog) are punished. This process is illustrated next in Figure 6.11, which shows the punishment and reward effect when the neural network is presented with rule number 17.

In Figure 6.11 the bold links identify the pathways (sets of input neurons, synapses and output neuron) that were rewarded, containing the evidences *e1*, *e2* and *e3* connected to the fog class. Similarly, the dotted links identify pathways containing the same evidences, but connected to the wrong class, not fog. Pathways with evidences not included in rule 17 are not affected.

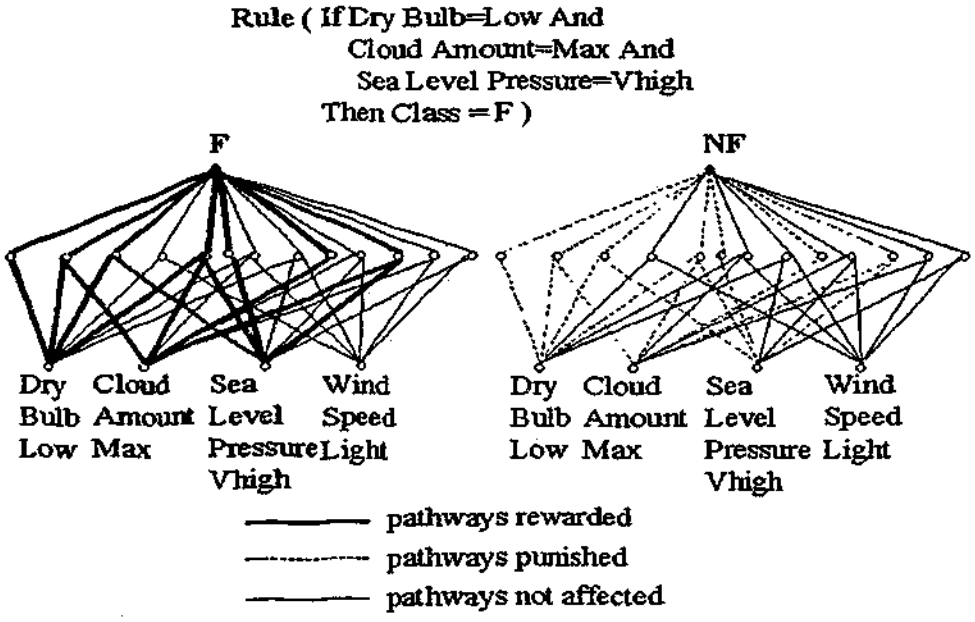

Figure 6.11: Training the CNM to learn an association rule

After scanning the training set and computing the punishment and reward values, no-rewarded pathways are pruned and the remaining pathways have their weights updated. The IRP algorithm updates the punishment and reward accumulators, removing all connections that have a larger number of punishments than rewards and weak connections.

Figure 6.12 illustrates the computation after the CNM has learned the rules listed in Table 6.1 (again considering only the evidences *e1* to *e4* for clarity purposes).

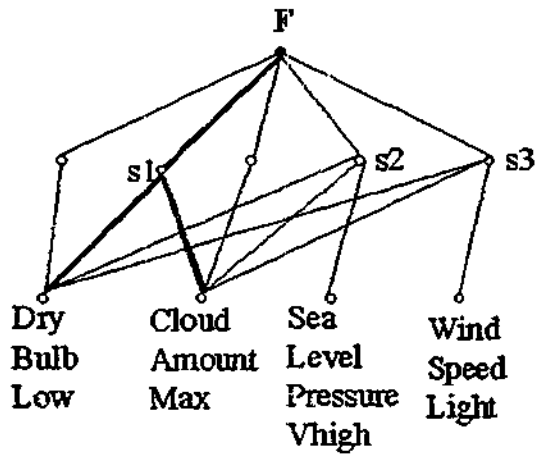

Figure 6.12: The trained CNM neural network

It can be observed in Figure 6.12 that all pathways leading to *not fog* were pruned, as there were no rewarded pathways leading to *not fog* (for the evidences *e1*, *e2*, *e3* and *e4)*. And the pathways leading to *fog* class remain.

The synapse *s1* in the network represents a strong pathway leading to fog class, indicating that the simultaneously occurrence of evidences *e1* and *e2* strongly indicates a fog case. The synapses *s2* and *s3* represent less strong information, and can be considered factual information related to rules 17 and 25 (evidences *e1* and *e2* are present in both fog rules).

The pruning threshold also has to be taken into account. If a higher threshold is set, only the strongest pathways will remain. All the experiments conducted in this research used pruning threshold 0.5, for example if a 0.6 threshold was used only the synapse *s1* would be kept in the network (this more rigid threshold might result in a loss of significant information, that was the reason the value 0.5 was set as pruning threshold).

The CANN simulator contemplates the learning functionality in its GUI, through the *Neural Network* option in its tool bar. When this option is selected, it shows two functionalities:

add a new neural network model and *simulate*. After a neural network model has been selected, the *simulate* option can be accessed to start the learning process. Figure 6.13 illustrates the learning dialog as it is implemented in CANN simulator.

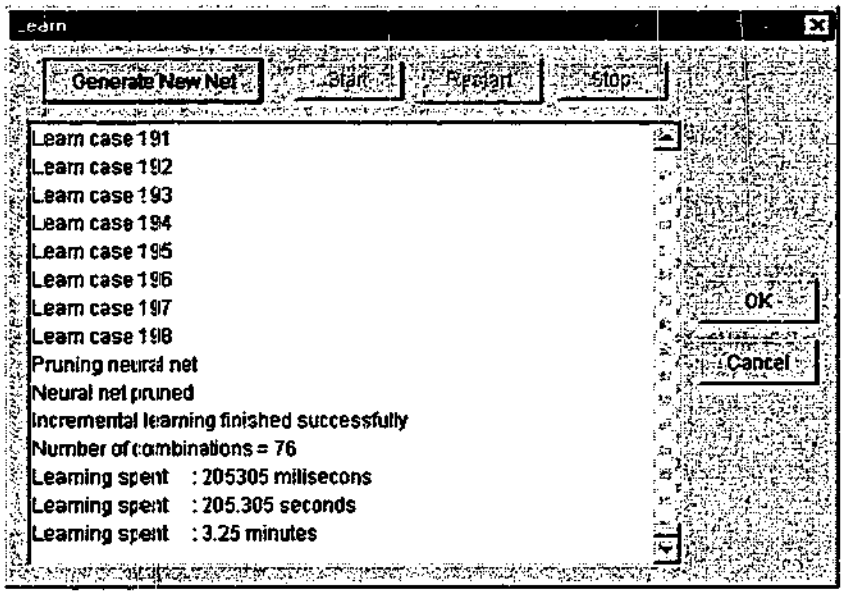

Figure 6.13: CANN Learning GUI

In the example given in Figure 6.13, the CNM network is generated from domain model Model1-60 already discussed in this chapter. Figure 6.13 shows a knowledge base (training set) with 198 rules (CANN does not differentiate rules and cases, any training set is treated as a set of cases), which was presented for learning. The learning process after the IRP algorithm was executed resulted in 76 connections with a learning time of 3.25 minutes.

After the training process is completed the trained network can be saved and the CANN simulator is ready to analyse cases and hypotheses about the problem domain it has been trained.

6.5 Consulting and Testing Cases

After the problem domain has been correctly modelled and integrated into CANN, and all training sets have been presented and the respective training processes finalized the CANN simulator is ready for consultation and test.

Consulting and testing cases and hypotheses are done through the CNM reasoning algorithm, which is accessed through the *Consult* functionality implemented in CANN.

The CNM reasoning algorithm is implemented according to the equations described in section 4.6, fuzzy *AND* and fuzzy *OR*. When presented with a set of evidences to analyse, the CNM input layer (input neurons) bypass incoming signals through the network that are propagated multiplying their values by synaptic weights. Combinatorial neurons propagate incoming signals according to a fuzzy *AND* operation and output neurons propagate incoming signals through a fuzzy *OR* operation. The output value of a fuzzy *OR* operation corresponds to the maximum arriving value from the lower layer, e.g., the highest value computed from the product of input signals with their corresponding synaptic weights. Consequently, CNM indicates the hypothesis (or class) with the higher output value as the possible answer to a consult. However, if the output value (also named *confidence degree*) is lower than a preset threshold the neural network fails to indicate a solution for the presented case.

A full discussion about the CNM reasoning algorithm can be found in (Denis and Machado, 1991), and the algorithm as it is implemented in CANN is explained in (Beckenkamp, 2002).

The CANN simulator implements two ways of consulting and testing cases, a *case consult* and a *case base consult*. In a case consult dialog the user can build cases at runtime to be presented to the NN for evaluation. Cases are built by selecting evidences in the case consult dialog and then activating the NN model to evaluate the selected evidences.

CANN evaluates the case and calculates the confidence degree for each hypothesis. The inference mechanism indicates the hypothesis with the highest confidence degree as the most suitable solution (class) to the problem. Figure 6.14 illustrates a case consult where four evidences are selected from the list of evidence attributes: *Dry Bulb Low*, *Cloud Amount Max*, *Sea Level Pressure Vhigh* and *Wind Speed Light*. The list in the left side of the window shows all the evidences modelled for the domain being analysed. The list in the right side of the window shows the evidences that were selected for evaluation. When clicking the *Test Case* button the NN is activated and evaluates the selected evidences (case).

The bottom part of the window in Figure 6.14 shows the NN output to the presented case. In this example the NN evaluated fog hypothesis as the correct class (winner hypothesis). The explanation for this is the simultaneous occurrence of evidences *Cloud Amount Max* and

Wind Speed Light, with a computed confidence degree of 0.953. This means that, based on what was learned from the training sets, these evidences strongly contribute to a fog occurrence. And *strongly* is here quantified as 0.953, based on the NN reasoning algorithm.

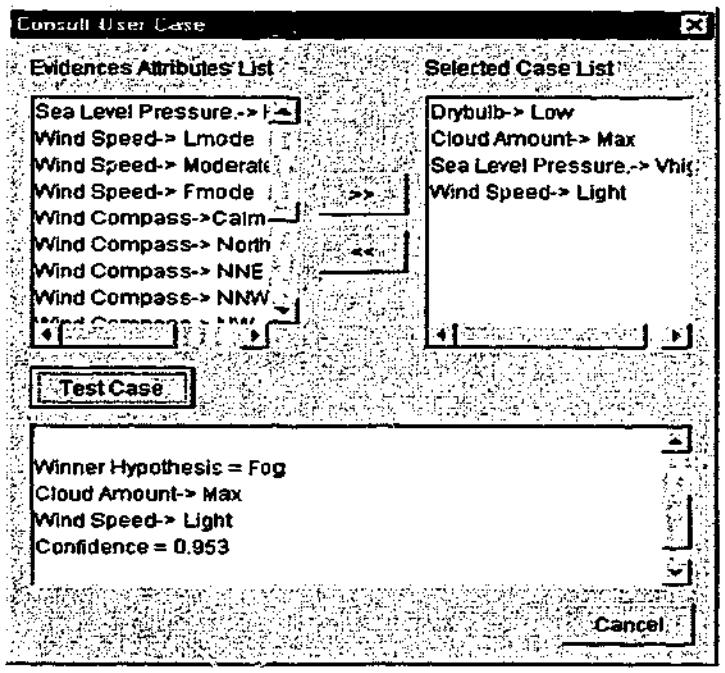

Figure 6.14: A fog case consult in CANN

Figure 6.15 shows the effect of the NN evaluation when the evidence *Wind Speed Light* is removed from the selected list and the case is again presented to the NN for evaluation.

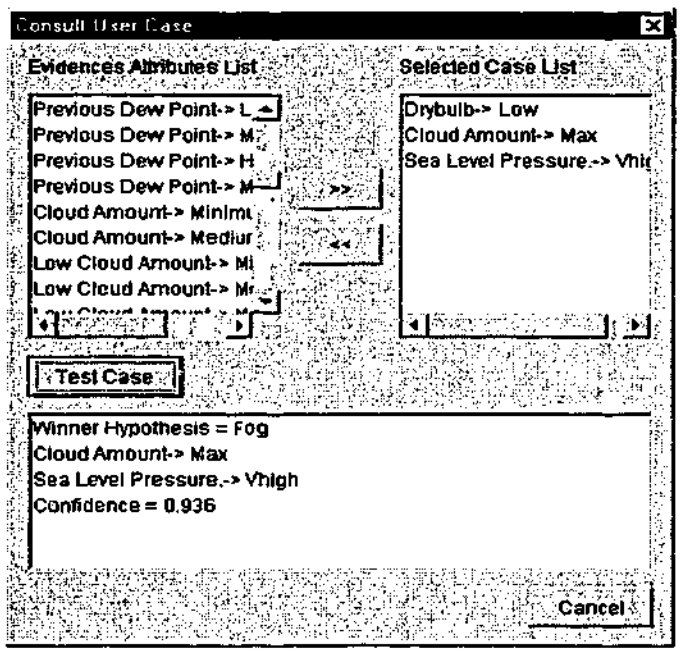

Figure 6.15: A case consult removing the wind speed light evidence

In the example shown in Figure 6.15, when removing *Wind Speed Light* the NN still indicates fog hypothesis as the correct output, however this is now supported by the occurrence of *Cloud Amount Max* and *Sea Level Pressure Vhigh* with a confidence degree of 0.936. From the examples presented above is possible to conclude that *Cloud Amount Max* is highly significant information in supporting fog hypothesis. Also, *Wind Speed Light* is more strongly associated with fog cases than *Sea Level Pressure Vhigh*, as the confidence degree for fog occurrence when *Wind Speed Light* is selected is higher (0.953) than when it is not selected (0.936).

It should to be noted that the examples shown in Figures 6.14 and 6.15 are results of a training process done with a knowledge base with 240 rules, from domain Model1-60. It is not a result from a training process done with the rules listed in Table 6.12, as those rules are for illustration purposes only. Training processes with only four rules or cases are unlikely to produce any NN pathway.

For illustration purposes, Figure 6.16 shows a case where not fog class is indicated as the correct answer by the NN. In this example, the selected evidences are *Dry Bulb High*, *Previous Dew Point High*, *Cloud Amount Medium* and *Sea Level Pressure Medium*. The NN supports its conclusion by indicating the joint occurrence of *Dry Bulb High* and *Cloud Amount Medium* as highly significant for not fog occurrence with a maximum confidence degree of 1.0.

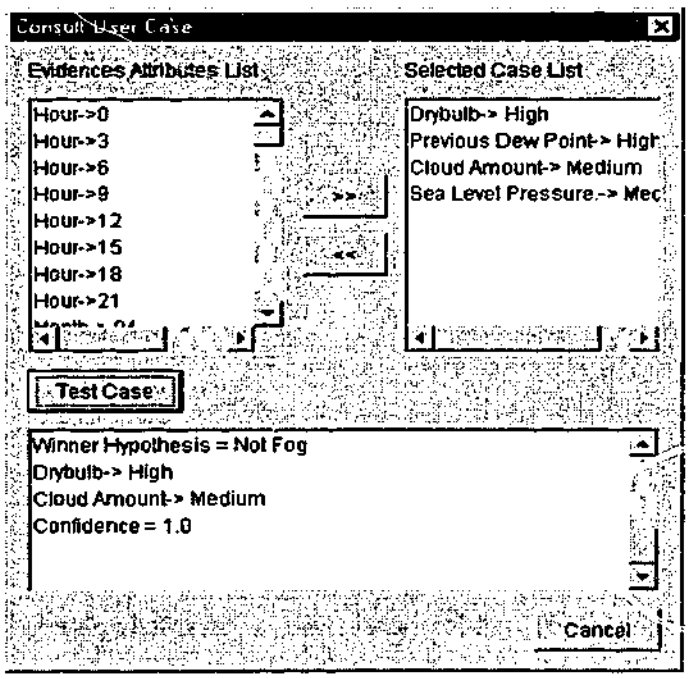

Figure 6.16: CNM evaluating a not fog case

The case base consult functionality is similar to the case consult, however, instead of presenting a single case (or one set of evidences) each time, several cases are simultaneously presented to the NN for evaluation. The list of cases is stored in a separate file, called the testing dataset. In the CANN version used in this research testing files have to be ASCII plain files, similar to the files used for training the NN.

In a case base consult the NN evaluates the set of cases in the same way it does for a single case, and stores the result of its evaluation in an ASCII file. Figure 6.17 illustrates a case base consult process. In this example a set of 120 cases were submitted for evaluation, and the results for the last 2 cases, case numbers 119 and 120 are shown.

Case number 119 resulted in a not fog output, supported by the evidences *Dry Bulb High* and *Cloud Amount Medium* with a 1.0 confidence degree. On the other hand, case 120 resulted in a fog output supported by the evidences *Cloud Amount* and *Low Cloud Amount* both *Maximum*, also with a 1.0 confidence degree. This indicates that the joint occurrence of these evidences is highly significant for their respective outputs, as a result of the dataset employed for training and the domain model as well.

At the end of a case base consult process CANN presents the classes (hypotheses) considered in the evaluation and the number of times these classes were indicated as the correct output to the presented set of cases.

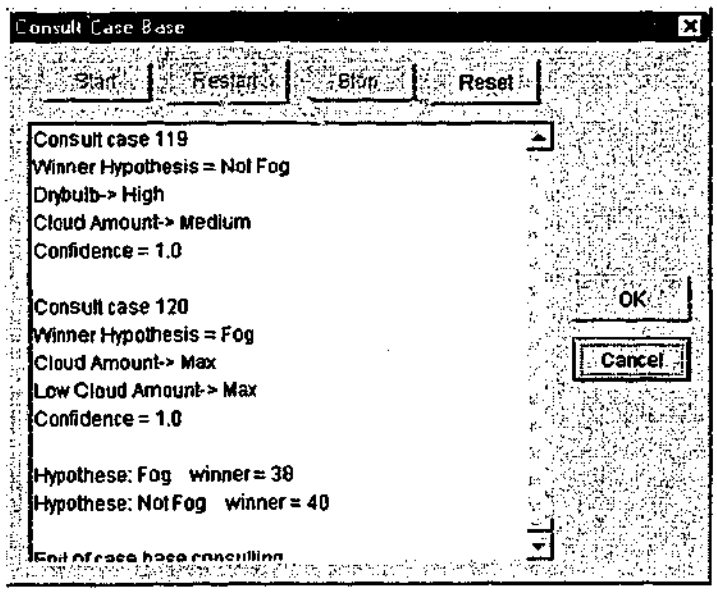

Figure 6.17: CNM evaluating a case base in aviation weather forecasting

For instance, in the example illustrated in Figure 6.17, fog was appointed as the correct solution in 38 cases and not fog in 40 cases. Also, it is possible to conclude that for 42 cases the NN was not able to reach a conclusion. This happens because for those cases there were no pathways in the NN structure leading to the output layer with weights equal or higher than the preset pruning threshold.

The testing datasets discussed in Chapter 5 (refer to section 5.3.4.2, "Generating Data Mining Models") were used to evaluate the performance of the hybrid DM-NN model through the case base consult functionality explained above. The testing datasets contain sets of weather observations that are submitted to the NN model for evaluation.

The next chapter discusses the results obtained from the NN evaluation. The testing datasets are presented and the results are discussed in the context of the DM-NN model's classificatory performance in aviation weather forecasting, specifically identifying fog cases.

6.6 Comments

The DM-NN model for IDSS developed in this research proposes a framework that can be seen as an infrastructure that can be built according to the needs of a particular problem.

In this regard the choice of the CANN simulator to implement the IAS component (refer to section 4.3.5, "Intelligent Advisory System"), besides the reasons outlined in the referred section also took into account the domain model functionality implemented in the CANN simulator, which makes it very suitable for classification problems. The possibility of incorporating a problem domain model based on hypotheses, evidences and attributes, and their combinations in the CANN simulator facilitates its deployment in solving classification problems.

The object-oriented design of NN models implemented in CANN facilitates the symbolic representation of the knowledge implicitly stored across the NN topology. This mechanism enables the identification of pathways that contribute to the system outputs, endowing the system with explanatory capabilities.

Furthermore, this information can be accessed and further employed in other components either to build more sophisticated explanatory features, such as graphical displays, or even to guide a more oriented reasoning process.

One of the drawbacks CANN presents concerns memory allocation. Neural networks with combination orders higher than three caused the system to fall in *exception* situations and stop running in some of the experiments conducted in this research. It might be the case that the large amount of evidences (used to build the CNM input layer) has an impact in generating an extremely large number of combinations, demanding an extremely large main memory capacity.

Another problem concerns data access. The knowledge bases listed in Chapter 5 had to be exported from MS Access tables into ASCII plain files, as the converter to fetch data from relational databases was not deployed in CANN at the time this research was being done. A significant amount of procedures in SQL queries and MS Visual Basic program modules had to be implemented to perform data integration into CANN. The exchange of information among those distinct technologies demanded a complex and timely process.

6.7 Chapter Summary

This chapter presented the Intelligent Advisory System (IAS) stage in the DM-NN model development in aviation weather forecasting. The IAS is the component responsible for implementing learning, reasoning, and issuing recommendations in the DM-NN model for decision support system.

The Intelligent Advisory System component is implemented within a neural network environment – the CANN simulator employing the CNM neural network model.

This chapter described the application of CANN in aviation weather forecasting. Firstly, the aviation weather forecasting domain modelling in CANN was described. It implies in modelling the application domain in a hierarchy of classes and objects describing the domain classes, evidences, attributes and the interrelationships among them.

This chapter also explained the method by which the aviation weather forecasting domain was integrated into CANN. Furthermore, the learning mechanism implemented by the CNM

NN was discussed and illustrated in the context of aviation weather forecasting, and the user interface functionalities were also presented.

Lastly, this chapter commented on some difficulties and drawbacks in the development of this application.

Chapter 7

7 Analysis of Performance

This chapter presents the last stage of the implementation of the DM-NN model in aviation weather forecasting; it discusses the performance of the DM-NN computational model in the context of fog identification at Tullamarine.

This chapter, together with Chapter 5 and Chapter 6 form the whole theme of the DM-NN model implementation in aviation weather forecasting, although each of these chapters covers distinct stages of this application.

7.1 Introduction

The performance of the DM-NN computational model for IDSS has been assessed according to its capability of correctly classifying meteorological observations, specifically fog cases in the context of aviation weather forecasting. It is a quantitative approach where the holdout method was employed.

As part of this research the activities of data preparation were intensively applied in order to tackle the problems related to data quality in the database delivered by the BOM, and a specific sampling design was developed and employed in order to address the issues of low prevalence classification and the necessary amount of cases for training, as described in Chapter 5.

Consequently, in the research described in this thesis the performance of the DM-NN model relies not only on the applied computational technologies e.g. data mining and neural networks, but also on the strategy applied in data modelling and knowledge modelling, as discussed in Chapter 5. In this scope data modelling relates to the activities that transform raw data into data used for data mining, including the activities of data pre-processing, features

selection and data sampling. Knowledge modelling relates to the activities of extracting knowledge from data, including the data mining process, tuning and testing data mining parameters and data models.

A series of experiments were conducted in order to determine the most appropriate set of parameters for data mining in the context of this research. Experiments taking into account different levels of rule confidence degree, rule support degree and rule order were executed within the data sets built according to different sampling proportions.

The knowledge models obtained were incorporated into a neural network based system used as an advisory system, providing the means to apply the discovered knowledge in supporting decision making.

The approach employed to assess the performance of the DM-NN model addresses the sampling design developed, the levels of rule confidence degree, rule support degree and rule order used during the data mining experiments. These issues impact on system performance, and consequently they have to be taken into account when assessing that performance.

Furthermore, it is important to recall that this research is concerned with the overall DM-NN model performance, especially its effectiveness in the decision making problem selected rather than the performance of a single technology. The argument presented in this research is that the DM-NN computational model proposes and describes a framework to support decision making through the cooperation of the knowledge discovery process and intelligent system technologies.

This chapter presents the experiments that were conducted and also the method applied to estimate the performance accuracy.

In this chapter the training sets are referred to by names that identify their original data models, and the data mining parameters used in their generation. Section 5.4.2.3, "The Knowledge Bases" explains the naming schema adopted.

7.2 The Assessment Method

According to what was previously discussed in section 1.5.3, "Performance Assessment Method", a quantitative approach using the holdout method (Weiss and Indurkhya, 1998) was applied to assess the performance of the experiments developed in this research.

In the holdout method the data is randomly partitioned into two mutually exclusive data subsets, a subset for training and another subset for testing. A given learning algorithm is first trained using the training set and tested on the testing set, and the accuracy evaluation is based on the amount of correctly classified cases in the testing set, and error rates. The error rate is the percentage of misclassifications, i.e., the ratio between misclassified cases and the total amount of cases in the testing set (Weiss and Indurkhya, 1998).

In this research the training sets were built through data mining experiments, as described thoroughly in Chapter 5, specially section 5.4.2.3 and Tables 5.17 to 5.20, which enumerate the training sets according to their respective original data mining models.

Testing sets were generated during the data modelling stage, and section 5.3.4, "Data Modelling for Data Mining" explains that process, and how independent testing sets were obtained for each data model. In brief, 7% of the fog population was randomly sampled for testing, resulting in 63 instances (the final fog testing set has 60 instances), excluding instances already selected for mining.

From the not fog population, testing sets were individually selected from each not fog data model in 10% proportions, and then amounts of 60 and 100 instances were selected. For data Model1-60 and Model1-80, 60 instances of not fog were randomly selected from the initial testing data set. For data Model2-60 and Model 10-60, 100 instances of not fog were randomly selected.

Again, instances were randomly sampled and were mutually exclusive from the instances selected for mining. Refer to Table 5.11 in Chapter 5 for a detailed description of the testing sets for each not fog model.

Final testing sets were obtained by combining the fog testing set with each not fog testing sets. This resulted in four testing sets, one for each data model. Table 7.1 describes the testing

sets. Table 7.1 shows the number of cases in each class, fog and not fog, and the total amount of cases for all testing sets.

Table 7.1: Testing sets

Data Models	Test set Data Model1-60	Test set Data Model1-80	Test set Data Model2-60	Test set Data Model10-60
Fog instances	60	60	60	60
Not fog instances	60	60	100	100
Total number of instances	120	120	160	160

Each testing set was employed in the experiments performed within its respective data model. For instance, the experiments that used training sets obtained from the data mining Model1-60 (see Table 5.17, Chapter 5) are tested through the testing set drawn from the same original data model.

Furthermore, during the data mining experiments only the parameters specified in Table 5.14 (Chapter 5) were applied. Regarding the CNM neural network model, all the experiments were done with the same set of parameters; the maximum combination order was set as 3, the acceptance threshold was specified as 0.5 and the pruning threshold as 0.4. The maximum combination order of 3 was specified because higher combination orders caused memory allocation problems during some experiments. The thresholds were arbitrarily selected, based on previous experiments done with the CNM neural network model (Viademonte, Leao and Hoppen, 1995; Pree, Beckenkamp and Rosa, 1997; Reategui, 1997). Those parameters remained unchanged during all the experiments.

7.3 Performance on Data Models

One of the first problems faced during the development of this research was how to deal with the low prevalence classification problem, and consequently which approach to use to sample the weather observation database for data mining and testing purposes. Selecting cases for data mining is recognized as a very unique task, as each application has different information requirements and principles (Liu and Motoda, 2001). Furthermore, according to Brighton (Brighton and Mellish, 2001) the class distribution, whether it is homogeneous or heterogeneous, is a decisive issue in choosing a particular case selection approach.

The weather observations database employed in this research showed a low prevalence classification problem with almost all cases allotted to the not fog class, and far fewer cases to the fog class. Fog class represents 1.92% of the population, and not fog class represents 98.08%.

Because of this significant difference in class distribution a specific sampling design was implemented. It combines stratified sampling with incremental sampling in an exploratory fashion, and can be classified as *stratified multi-stage sampling*. Chapter 5 discusses the sampling design employed in detail.

In the first stage a *stratified sampling* approach was used to separate the survey variable *fog type* dividing the population in two strata, fog and not fog respectively. In the second stage *incremental random sampling* was used to sample only the not fog class. The incremental sampling approach was employed in this research as it was not possible to know beforehand which not fog proportions would lead to a satisfactory performance.

At the third stage, from each sample in each class, *random sampling without replacement* was applied to generate the data sets for mining and testing, meaning that mining and testing sets always have different instances.

In the last stage, subsets of the initial population were reconstructed by joining each not fog class data sets with their respective fog class data sets (e.g. mining and testing data sets), obtaining as a result the final data sets for mining and testing.

As a result of this sampling method four data models for data mining were obtained and named according to their sampling proportions, namely Model1-60, Model1-80, Model2-60 and Model10-60 (refer to Table 5.12 in Chapter 5). Model1-60 means that this model was obtained from a sample of 10% out of the not fog stratum, and 60% of this sample was selected for mining purposes and 10% for testing purposes. After that, testing and mining datasets from the fog and not fog stratum were joined to form the final datasets. Model1-80 is similar to Model1-60, but in this case 80% of the not fog sample was selected for mining purposes.

Model2-60 is a model obtained from 20% out of the not fog stratum, with 60% of this sample selected for mining purposes. And Model10-60 means that the whole not fog stratum was employed, and 60% of the stratum was selected for mining purposes.

The main aim of this sampling design was to achieve a more homogeneous class distribution in the datasets used for data mining, and consequently better descriptive models for the minority class. The sampling strategy employed is explained in detail in section 5.3.4, "Data Modelling for Data Mining."

7.3.1 Performance According to Sampling Proportions

Exploratory multiple runs were conducted through the holdout method to verify whether data mining models result in better classificatory performance. The training sets described in Chapter 5 were employed to train the CNM neural network model within the CANN simulator, and their accuracies were verified through the testing sets described in Table 7.1. To measure the classificatory performance the error rates and classification rates on testing data were computed (Weiss and Indurkhya, 1998).

The training sets listed in Tables 5.17 to 5.20, were employed to assess the performance of the data mining models. The training sets listed in Table 5.17 were employed to assess the performance of data mining model Model1-60, the training sets listed in Table 5.18 were employed to assess the performance of the data mining Model1-80, the training sets listed in Table 5.19 were used for the assessment of data mining Model2-60, and lastly data mining Model10-60 was assessed through the training sets listed in Table 5.20.

The results obtained are listed in Table 7.2 for each model. Table 7.2 shows the number of cases for which the system could not reach a conclusion (which is not the same as misclassified cases), the classificatory performance on testing data, (i.e. the percentage of correctly classified cases) and the error rates.

The classificatory performance for each training set was computed and the average was taken to obtain the measure for each model (Weiss and Indurkhya, 1998; Pyle, 1999), according to the respective training sets (see Section 7.3.2, "Performance on Training Sets.")

Table 7.2: Data mining models performance

Data Mining Model	Not Evaluated	Classificatory Performance	Error Rates
Model1-60	13.19%	64.72%	0.35
Model1-80	6.67%	70.83%	0.29
Model2-60	6.67%	74.58%	0.25
Model10-60	8.75%	70.00%	0.30

The same computation was done for the error rates. The error rate for each training set was computed and the average (Breiman et al., 1984) was calculated to obtain the error rate for each model. Figure 7.1 summarizes the error rates for each data model.

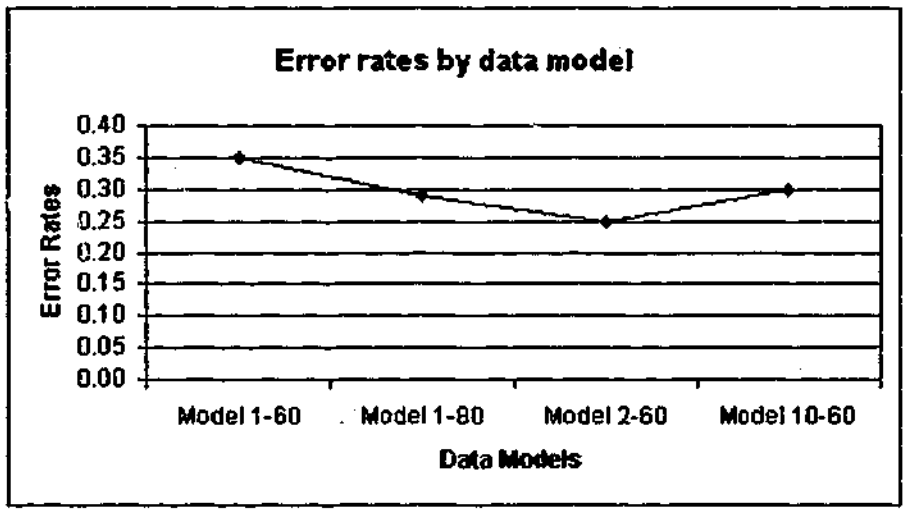

Figure 7.1: Error rates by data model

According to the results illustrated in Table 7.2 and Figure 7.1 Model2-60 showed the best classificatory performance with 74.58% correctly classified cases, and the smallest error rate of 0.25, followed by data model Model1-80 with 70.83% correctly classified cases and an error rate of 0.29. Model1-60 shows the worst classificatory performance with 64.72% correctly classified cases and an error rate of 0.35.

Mode2-60 was obtained out of a 20% sample from the not fog stratum, containing 9572 instances of not fog. The data set for mining obtained from Model2-60 contains a sample of 60% from the not fog stratum, with 5699 not fog instances and 807 fog instances (all data models have the same number of fog instances), with a total of 6506 instances. This data model has a more heterogeneous class distribution than Model1-80 and Model1-60 (see Table 5.12, "Data mining models" in Chapter 5); as such its superior performance cannot be related

to the class distribution but could possibly be associated to the amount of cases in the data mining set. This means that despite having a more heterogeneous class distribution, this model has a higher amount of cases in its data mining set, which resulted in the generation of better association rules for the training data sets used to train the neural network model.

The fact the Model10-60 performed surprisingly better than Model1-60 also supports this conclusion, as Model10-60 has the worst class distribution but has the higher number of cases associated to not fog class. Normally, as the number of cases increases, error rates tend to decrease (Weiss and Indurkhya, 1998), which possibly explains why Model10-60's performance was better than Model1-60, which has a smaller number of cases.

It is very difficult to control the effect of instance selection on the generalization capability of trained neural networks, as this also depends on learning algorithms and neural network topology. However the relation between the size of the training set and algorithm classificatory performance has been studied by several researchers such as (Harris-Jones and Haines, 1997; Weiss and Indurkhya, 1998). Breiman (Breiman et al., 1984), for example shows that error rate decreases as the sample size increases in experiments using decision trees, and Chauchat (Chauchat and Rakotomalala, 2001) shows how the generalization error rate decreases as the sample size increases in experiments conducted with ChAID decision trees.

7.3.2 Performance on Training Sets

The training sets described in Chapter 5 were individually assessed to verify which resulted the best classificatory performance. The classificatory rates, the number of cases for which a conclusion could not be reached, and the error rate in the testing sets were individually computed.

Training sets were assessed according to their respective data mining models.

7.3.2.1 Performance of Data Mining Model1-60

Table 7.3 shows the results obtained for Model1-60. The training set Train_Model1-60V370 showed the best performance with an error rate of 0.32, a classificatory rate of 68.33% and 8.33% of not evaluated cases in the testing set. This training set had the highest number of rules and a better class distribution within its model, with 405 rules, 104 of which describing fog and 301 describing not fog.

Table 7.3: Performance of data mining model Model1-60

Training set	Not Evaluated	Classificatory Rate	Error Rate
Train_Model1-60V270	12.50%	65.00%	0.35
Train_Model1-60V280	12.50%	64.17%	0.36
Train_Model1-60V290	23.33%	56.67%	0.43
Train_Model1-60V370	8.33%	68.33%	0.32
Train_Model1-6V380	10.00%	67.50%	0.33
Train_Model1-60V390	12.50%	66.67%	0.33

As a result of its better class distribution and higher number of rules the better performance of Train_Model1-60V370 was expected.

7.3.2.2 Performance of Data Mining Model1-80

Table 7.4 shows the results obtained for model Model1-80. Here two training sets showed the best classificatory performance, and with similar results. Train_Model1-80V370 and Train_Model1-80V380 both had error rates of 0.26, a classificatory rate of 74.17% and 1.67% of not evaluated cases.

These training sets differ in the levels of rule confidence degrees assigned to them, with 70% and 80% respectively, having the highest amount of cases in their group. Train_Model1-80V370 has a higher number of rules (358 rules) as a result of a more flexible level of confidence degree applied in its generation. Train_Model1-80V380 has 314 rules.

Table 7.4: Performance of data mining model Model1-80

Training set	Not Evaluated	Classificatory Rate	Error Rate
Train_Model1-80V270	3.33%	72.50%	0.28
Train_Model1-80V280	3.33%	71.67%	0.28
Train_Model1-80V290	[illegible]33%	60.00%	0.40
Train_Model1-80V370	1.67%	74.17%	0.26
Train_Model1-80V380	1.67%	74.17%	0.26
Train_Model1-80V390	6.67%	72.50%	0.28

A more detailed analysis showed that Train_Model1-80V370 had a better performance than Train_Model1-80V380 in classifying fog cases, with 65% and 61.67% of correctly classified fog cases respectively. This result can be explained as Train_Model1-80V370 has 67 association rules in the fog class, while Train_Model1-80V380 has 23 rules in the same class. However Train_Model1-80V380 showed a better performance in classifying not fog cases,

with 86.67% being correctly classified, while Train_Model1-80V370 had 83.33% of correctly classified not fog cases. The better classificatory performance of Train_Model1-80V380 in classifying not fog cases was expected as proportionally it has a higher number of not fog cases.

Train_Model1-80V370 showed a better descriptive capability for fog phenomena, possibly because of its higher number of association rules allotted to the fog class.

7.3.2.3 Performance of Data Mining Model2-60

Train_Model2-60V270 and Train_Model2-60V280 equally demonstrated the best performance with an error rate of 0.24, a classificatory rate of 75.63% and 6.25% of not evaluated cases in the testing set.

Train_Model2-60V270 has more association rules (209 rules), than Train_Model2-60V280, with 195 rules. Additionally, Train_Model2-60V270 has more rules describing fog class with 32 rules (15.31%), while Train_Model2-60V280 has 18 rules (9.23%) allotted to the fog class. Because of this difference in between the number of rules and specifically the number of rules allotted to fog class, Train_Model2-60V270 was expected to perform better. The equal performance of these datasets may be explained by the descriptive quality of rules, rather than the amount of rules.

Table 7.5 shows the results obtained for model Model2-60.

Table 7.5: Performance of data mining model Model2-60

Training set	Not Evaluated	Classificatory Rate	Error Rate
Train_Model2-60V270	6.25%	75.63%	0.24
Train_Model2-60V280	6.25%	75.63%	0.24
Train_Model2-60V290	7.50%	73.13%	0.27
Train_Model2-60V370	6.25%	75.00%	0.25
Train_Model2-60V380	6.25%	75.00%	0.25
Train_Model2-60V390	7.5%	73.13%	0.27

However, when analysing the classificatory performance only in the fog class Train_Model2-60V380 showed the best performance with 68.33% of correctly classified fog cases (refer to Table 7.6). Train_Model2-60V270 and Train_Model2-60V280 both have

classificatory rate of 63.33% in the fog class, a difference of 5%. Table 7.6 illustrates the classificatory performance of Model2-60 on fog class.

Table 7.6: Performance of data mining model Model2-60 on fog class

Training set	Classificatory Rate on fog class
Train_Model2-60V270	63.33%
Train_Model2-60V280	63.33%
Train_Model2-60V290	65.00%
Train_Model2-60V370	65.00%
Train_Model2-60V380	68.33%
Train_Model2-60V390	65.00%

Regarding the amount of cases not evaluated Train_Model2-60V380 was the same as Train_Model2-60V270 and Train_Model2-60V280, with 6.25% of not evaluated cases, but with a slightly higher error rate of 0.25 (a difference of 0.01 higher than Train_Model2-60V270 and Train_Model2-60V280).

With its better performance in classifying fog cases and its small difference in error rates, Train_Model2-60V380 was considered as having the best descriptive capability within this data model.

7.3.2.4 Performance of Data Mining Model10-60

Table 7.7 lists the results obtained with Model10-60, through the training sets listed in Table 5.20 (Chapter 5).

Table 7.7: Performance of data mining model Mode10-60

Training set	Not Evaluated	Classificatory Rate	Error Rate
Train_Model10-60V250	10.00%	68.13%	0.32
Train_Model10-60V350	7.5%	71.88%	0.28

This data model was obtained through a random sampling of the entire population.

The better performance of Train_Model10-60V350 was expected as it has a higher number of rules than Train_Model10-60V250, with 298 and 187 rules respectively. Also, both datasets have similar numbers of rules describing the fog class, 18 and 17 rules respectively.

Both data sets showed a surprisingly good classificatory performance, mainly in fog classification, with 71.67% and 65% accuracy for fog class respectively. These results were unexpected, because of the small amount of rules describing fog in both datasets. It potentially

indicates that the CNM neural network model was able to generalize relevant pieces of knowledge about fog phenomena even when presented with small sets of rules.

7.4 Performance on Data Mining Parameters

Besides sampling proportions, another problem faced in this research concerns the selection of data mining parameters, specifically tunning the thresholds of rule confidence degree, rule support degree and rule order. The aim is to achieve an acceptable classificatory performance, as indicated by the error rate on testing data.

Data mining parameters orient the processing of rules, specifically pruning uninteresting rules and ranking rules according to some criteria. The parameters used for rule filtering in this research project are: the minimum level of *rule confidence* and *support degree*, and the maximum *rule order*, e.g. the maximum number of rule antecedent itemsets.

Confidence and support constraints are important because they work as filters for relevant rules; in general, relevant rules have support and confidence degrees above some minimum threshold. The problem is to effectively assign the best-fit thresholds for these parameters in a given situation. Overly restrictive thresholds are likely to miss important information, but thresholds that are too flexible might produce too much uninteresting information.

In the research described in this thesis rules with 50%, 70%, 80% and 90% confidence degree were generated. As it was impossible to know beforehand the amount of rules that could be obtained according to a specific confidence degree, it was decided to use the most frequent thresholds applied in data mining applications (Weiss, Galen and Tadepalli, 1990; Piatetsky-Shapiro and Frawley, 1991; Klemettinen, Mannila and Toivonen, 1997). The goal here was to verify the difference in performance according to different combinations of parameters (confidence degree, minimum support degree and maximum rule order), and to determine which combinations of these parameters lead to a better classificatory performance.

First the performance according to rule confidence degree is discussed, followed by the performance according to rule support and rule order.

7.4.1 Performance According to Rule Confidence Degrees

Rules with 70%, 80% and 90% confidence degrees were generated from all data mining models in this research, with the exception of Model10-60. This model contains the whole population, and an extremely unbalanced class distribution, as such a more flexible confidence degree was allowed in order to obtain a reasonable number of rules for the fog class. Rules with 50% confidence degree were generated specifically for Model10-60.

Table 7.8 shows the classificatory performance and error rate of testing data for training sets according to the levels of rule confidence degree.

The V2 and V3 suffixes assigned to the training dataset names in Table 7.8 identify the minimum rule support and rule order, where V2 identifies minimum support of 8% and maximum rule order of 7, and V3 identifies minimum support of 6% and maximum rule order of 10.

The training sets listed in Table 7.8 are the same listed in section 7.3.2, "Performance on Training Sets," but grouped according to their respective rule confidence degrees.

Table 7.8: Rule confidence degree on testing data

Training datasets	Rule Confidence Degrees							
	50%		70%		80%		90%	
	Correctly Classified	Error Rates	Correctly Classified	Error Rates	Correctly Classified	Error Rates	Correctly Classified	Error Rates
Model1-60(V2)			65.00%	0.35	64.17%	0.36	56.67%	0.43
Model1-60(V3)			68.33%	0.32	67.50%	0.33	66.67%	0.33
Model1-80(V2)			72.50%	0.28	71.67%	0.28	60.00%	0.40
Model1-80(V3)			74.17%	0.26	74.17%	0.26	72.50%	0.28
Model2-60(V2)			75.63%	0.24	75.63%	0.24	73.13%	0.27
Model2-60(V3)			75.00%	0.25	75.00%	0.25	73.13%	0.27
Model10-60(V2)	68.13%	0.32						
Model10-60(V3)	71.88%	0.28						
AVERAGE	**70.01%**	**0.30**	**71.77%**	**0.28**	**71.36%**	**0.29**	**67.02%**	**0.33**

It is important to recall that the levels of confidence degree were specified to generate sets of association rules, which were used to train the neural network model. Consequently the measures of classificatory performance and error rate were taken from the testing data after the neural network system was trained with the sets of association rules.

Table 7.8 shows the average measures calculated according to each level of confidence degrees. The error rates for the experiments with 70% and 80% confidence degrees showed the best performance, with a small improvement in performance in the 70% confidence degree. The experiments with training sets obtained from Model1-80 and Model2-60 showed equal error rates in both 70% and 80% levels of confidence degrees.

A slightly better classificatory performance is verified within training sets obtained from Model1-60 using 70% confidence degree, with 0.35 and 0.32 error rates (70% confidence degree) and 0.36 and 0.3? for 80% confidence degree.

The best results were achieved with the training sets obtained from Modell2-60, with similar error rates of 0.24 and a classificatory rate of 75.63% in both 70% and 80% levels of confidence degrees.

Figure 7.2 summarizes the error rate measures according to the levels of confidence degrees.

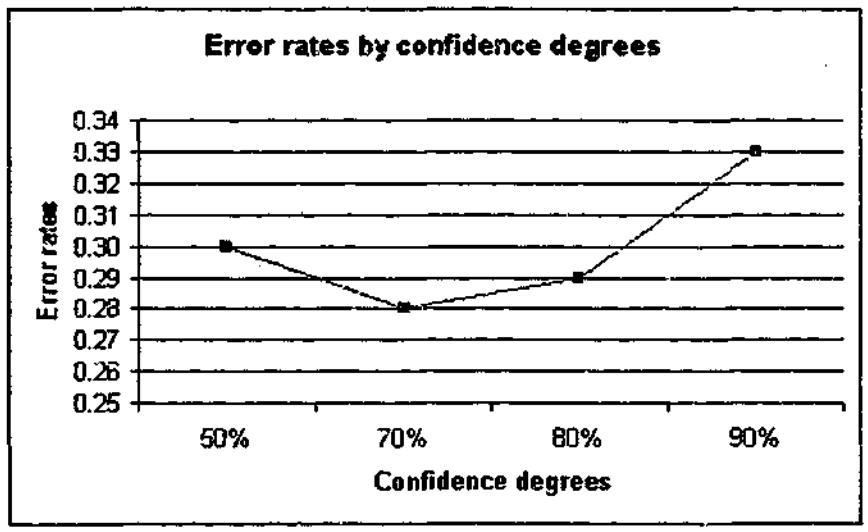

Figure 7.2: Error rates on testing data by levels of confidence degrees

The experiments with 70% confidence degree showed the best results on average with an error rate of 0.28, and a classificatory rate of 71.77%. Average error rates of 0.30, 0.29 and 0.33 were verified for the experiments with confidence degrees of 50%, 80% and 90% respectively.

The experiments with a 90% confidence degree showed the worst performance, and this was expected because of the small number of association rules obtained when setting this

parameter. (Refer to Table 5.15 and Table 5.16 to verify the number of association rules in the training sets). The same observation can be extended to the 50% confidence degree, i.e. the small amount of rules resulted in poor performance, specifically related to the number of rules describing fog class, having 17 and 18 association rules in the experiments conducted. It is possible that a more relaxed level of confidence degree, i.e. less than 50%, assigned to Model10-60 would produce better results, because of the higher number of rules that would be generated for the fog class.

In the experiments conducted in this research the training sets obtained from data mining experiments using a 70% rule confidence degree provided the highest amount of rules for both classes (fog and not fog) comparing within the training sets obtained from same data models using different levels of rule confidence degrees.

Figure 7.2 shows how the error rate increases as the rule confidence degree increases from the level of 70%. This association can be related to the amount of association rules obtained, as increasing the level of confidence degree implies a more constrained parameter to the association rule generator algorithm, consequently fewer rules are generated.

In the experiments with a 50% confidence degree the error rate also increased compared to the 70% confidence degree, but this does not necessarily mean that the error rate also increases when rule confidence degree decreases from 70%. It should be noted that the 50% confidence degree was only applied in Model10-60, and even with this more flexible confidence degree fewer rules of fog class were obtained. Consequently, in the specific case of this research, the same observation to the 90% confidence degree can be extended to the 50% confidence degree applied to Model10-60, that too few association rules for fog class were obtained for a good descriptive capability.

7.4.2 Performance According to Rule Support and Rule Order

Two sets of data mining experiments were conducted in this research, according to different combinations of rule support and rule order degrees (refer to Section 5.4.1.5, "Selecting Mining Parameters" for a discussion about these experiments).

In the first set of experiments the minimum rule support was set to 8% and the maximum rule order to 7, with levels of confidence degree of 50% (for Model10-60 only), 70%, 80%, and 90%. This first set of experiments is identified as **V2**.

The results obtained in these first experiments were considered very restrictive as the numbers of rules obtained, mainly for the fog class, were considered too small for a good descriptive capability (see Table 5.15). Consequently, it was decided to repeat the data mining experiments using more flexible mining parameters, with the aim of achieving a more equally balanced class distribution in the resulting association rule sets (see Table 5.16).

The second set of experiments was conducted keeping the same levels of rule confidence degrees, but relaxing the minimum rule support to 6% and allowing a higher number of rule itemsets, setting the maximum rule order to 10 itemsets. This second set of experiments is identified as **V3.**

Chapter 5 discusses these experiments, and this section discusses the classificatory performance of the training sets according to the levels of rule support and rule order.

Figure 7.3 shows the trend of error rates on testing data according to rule support degree and rule order by each training dataset.

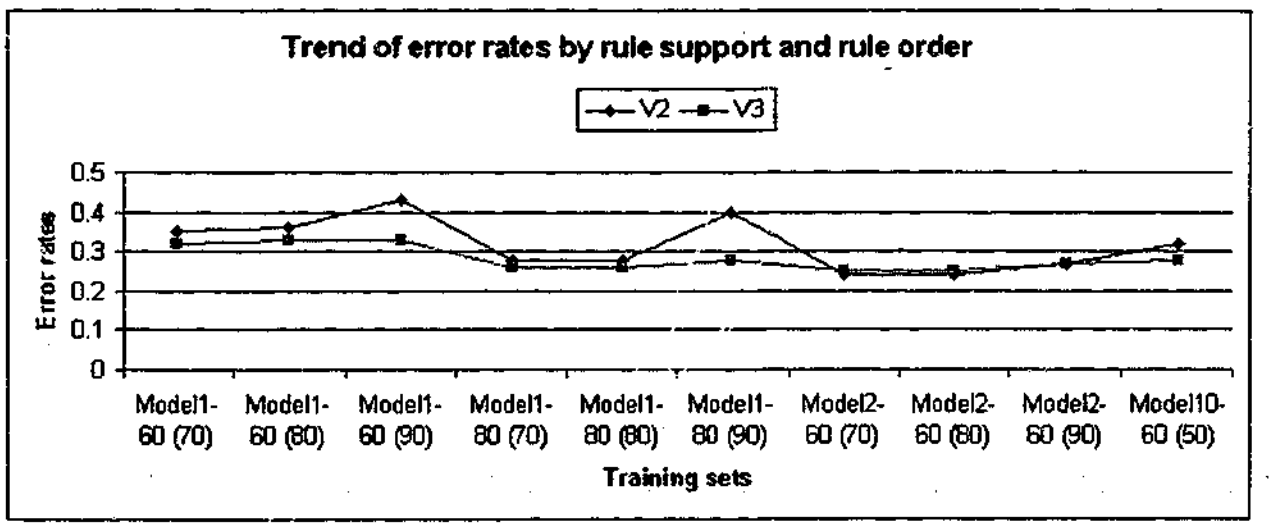

Figure 7.3: Trend of error rates on testing data by levels of rule support and rule order

In Figure 7.3, the training sets are grouped according to their minimum rule support and maximum rule order. For example, the first training sets shown are the training sets obtained from Model1-60, with a 70% rule confidence degree. The square icons represent the

experiment with a minimum rule support of 8% and a maximum rule order of 7, and the round icons represent the experiment with a minimum rule support of 6% and a maximum rule order of 10.

In the research described in this thesis the experiments using a minimum rule support of 6% and a maximum rule order of 10 itemsets (V3 configuration) outperformed the experiments using rule support of 8% and a maximum rule order of 7 (V2 configuration) for almost all training sets. This is mainly because the training sets obtained from the V3 configuration have a higher number of association rules than the training sets obtained from the V2 configuration, mainly concerning the fog class. As such the decision to relax those parameters in order to get a more balanced class distribution was correct.

The only cases where this increment in performance was not observed were in the experiments with training sets obtained from Model2-60 with 70% and 80% rule confidence degrees. In this case a slightly better performance for the V2 configuration was observed with an error rate of 0.24, compared to 0.25 in the V3 configuration. For Model2-60 with a 90% confidence degree the error rates were the same, 0.27 for both configurations.

The possible reason for this exception is that the amount of association rules, mainly in the fog class, slightly differs among the training sets obtained from Model2-60 with 70% and 80% rule confidence degrees in both the V2 and V3 configuration.

The training set obtained from Model2-60 with a 70% confidence degree using the V2 configuration (identified as Train_Model2-60V270) has 32 association rules allotted to the fog class and the training set obtained from the same model with a 70% confidence degree but using the V3 configuration (identified as Train_Model2-60V370) has 45 association rules allotted to the fog class, a difference of 13 association rules. At the 80% confidence degree this difference is even smaller, with 18 association rules allotted to the fog class using the V2 configuration, and 20 association rules using the V3 configuration, a difference of just 2 association rules.

This is because Model2-60 has a more homogeneous class distribution than Model1-60 and Model1-80, this might prevent the mining sets obtained from Model2-60 being too sensitive to changes due to the setting of different data mining parameters.

Figure 7.4 summarizes the results of the V2 and V3 experiments, through the averages of error rate obtained on testing data.

The error rates were computed for all training sets using both V2 and V3 configurations, as shown in Figure 7.4, and then the averages of these results for each configuration were also computed.

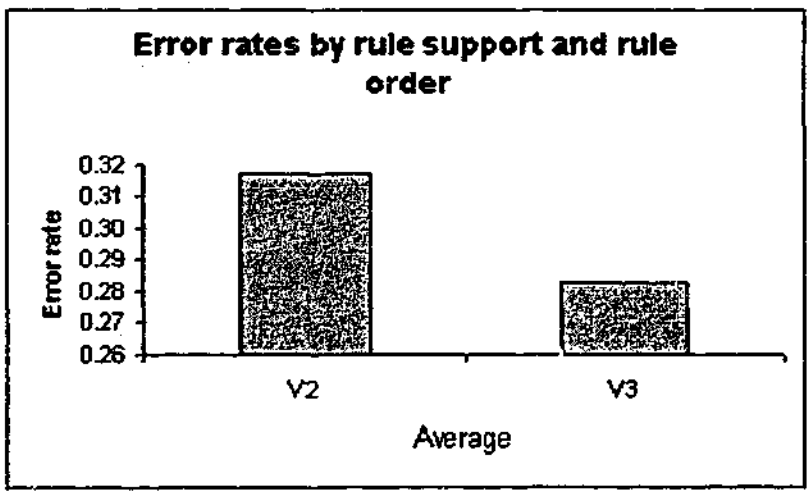

Figure 7.4: Error rates by levels of rule support and rule order

Figure 7.4 demonstrates that the experiments using a minimum rule support of 6% and a maximum rule order of 10 itemsets (V3) outperformed the experiments using a minimum rule support of 8% and a maximum rule order of 7 itemsets (V2). The V3 configuration has an error rate average of 0.28 with a 71.84% classificatory rate, and the V2 configuration has an error rate average of 0.31 with a 68.25% classificatory rate.

This is mainly because the training sets obtained from experiments using V3 configuration have a higher number of association rules than those of V2, mainly concerning the fog class, as already discussed in this chapter. This was the main reason these experiments were conducted, to achieve a more equally balanced class distribution in the association rule sets, with the expectation of a better classificatory performance.

Obviously, new experiments could be conducted to try different combinations of rule support and rule order thresholds in order to achieve a higher classificatory performance. However, this research does not aim to discover the best possible solution for the task of fog classification. The aim here is to achieve satisfactory results that indicate the applicability of the DM-NN model for decision support, rather than optimal results in one specific scenario.

7.5 Some Specific Analysis with Weather Observations

As part of this research, intensive data preparation work was developed with the weather observations database, as discussed thoroughly in Chapter 5.

Features (or attributes) selection contributes to the descriptive capabilities of models, and is particularly relevant to projects dealing with inductive algorithms. The problem is that it is not always clear whether a particular feature is relevant or not to a particular domain, and whether selecting it or not will affect predictive performance. Features selection usually requires domain knowledge, and very often relevant (or irrelevant) features are only verified empirically. In that perspective, additional experiments were conducted as part of this research to assess the relevance of certain weather observations.

7.5.1 Assessing the Visibility Observation

Fog phenomena is primarily defined as restricting visibility, to a level equal or less than one kilometre (Auer Jr., 1992). In the weather observation database there is an attribute (observation) that numerically describes the level of visibility at the airport when the observation was recorded.

The question that arose during the development of this research was that the visibility observation might be synonymous with fog occurrence, in which case it could be discarded from the database or even replace the *Fog Type* attribute.

As a result of the discretization procedures developed (refer to section 5.4.1.4, "Discretization of Numerical Attributes") the visibility observation was discretized into four levels: Level1, Level2, Level3 and Level4. Level1 indicates the worst level of visibility, and Level4 indicates the highest level of visibility, usually higher than 40 kilometres. It has to be noted that fog might occur over a broad area (even in situations with high levels of visibility, there is a possibility of fog patches at the airport. In this case the visibility over the airport can be very poor, even if the visibility in the area is generally high).

The data mining experiments were replicated to include the visibility observation (in the first experiments visibility observation was not included), keeping the same data mining and neural network parameters. The experiments including the visibility observation had the particular purpose of verifying if this information might be considered as synonymous with

fog occurrence. For instance, low visibility values would indicate a definite occurrence of fog, and high visibility values would indicate a definite not fog case. The resulting rule sets from the experiments conducted including the visibility attribute are listed in Appendix C.

Model2-60 was selected to conduct the experiment including visibility observation, because it showed the highest classificatory performance in the experiments previously conducted without the visibility observation. (Any data mining model could be selected; the only requirement is that the same model has to be used in both experiments in order to be comparable). Table 7.9 shows the results obtained.

Table 7.9 lists the training sets obtained from Model2-60, including the visibility observation, their classificatory rates, the number of cases for which a conclusion could not be reached, and the error rate in the testing set. The suffix "V" in the training set names is used to indicate the visibility attribute was included during the data mining runs.

Table 7.9: Performance of Model2-60 including Visibility

Training set	Not Evaluated	Classificatory Rate	Error Rate	Rate on Fog Classification
Train_Model2-60V270_V	1.88%	76.88%	0.23	81.67%
Train_Model2-60V280_V	2.50%	76.88%	0.23	81.67%
Train_Model2-60V290_V	2.50%	78.13%	0.22	68.33%
Train_Model2-60V370_V	1.88%	78.75%	0.21	86.67%
Train_Model2-60V380_V	1.88%	76.88%	0.23	83.33%
Train_Model2-60V390_V	2.50%	76.25%	0.24	76.67%
AVERAGES	**2.19%**	**77.29%**	**0.23**	**79.72%**

According to Table 7.9 all training sets showed a better performance when the visibility attribute was included (refer to Table 7.5 to see the results of not including visibility), with Train_Model2-60V370_V showing the best performance, with 78.75% of correctly classified cases, **86.67%** of correctly classified fog cases, and an error rate of 0.21. These results indicate the relevance of visibility observation in fog classification, with a significant increase in performance compared to the performance of Model2-60, not including the visibility observation.

Figure 7.5 illustrates the average measures from Model2-60 with and without the visibility. It can be observed in Figure 7.5 that including the visibility attribute in the data mining experiments resulted in better descriptive models. The fog classificatory performance with

visibility information is 79.72%, and 65% when visibility is not included. Also, the data models with visibility showed a much better performance concerning cases where the NN model could not reach a conclusion of either fog or not fog. The percentage of not evaluated cases is 2.19% with visibility and 6.67% without, a difference of 4.48%. This indicates that the CNM neural network model demonstrated a better generalization capability when the visibility observation was presented in the set of rules used for training.

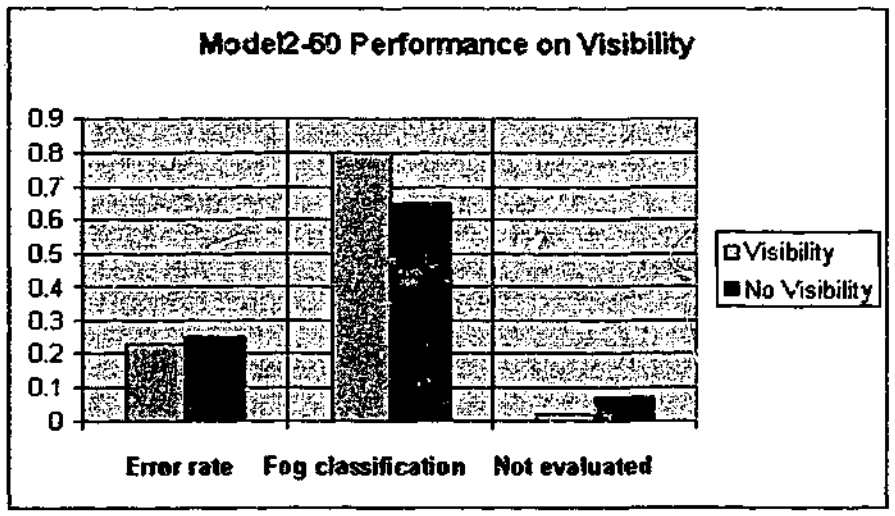

Figure 7.5: Average measures on testing data by Visibility on Model2-60

These results indicate the relevance of visibility observation in fog classification. To clarify this relevance, the neural network pathways with higher confidence degrees were selected when the training set with the best performance (Train_Model2-60V370_V) was presented to the neural network. The neural network pathways were obtained after the cases from the test set were evaluated.

The combinations with higher confidence degrees for the fog class were *wind speed light* and *visibility level1*, with a confidence degree of 1.0, *cloud* and *low cloud amount* both *maximum* with a confidence degree of 0.965, *sea level pressure vhigh* and *visibility level1* with a confidence degree of 0.964, and *low cloud amount maximum* and *visibility level1* with a confidence degree of 0.942. Figure 7.6 illustrates the neural network pathways corresponding to these combinations.

The selected pathways clearly indicate that *visibility level1* (lower levels of visibility) is strongly related to fog classification, and certainly the absence of this information leads to a drop in classificatory performance, as previously discussed (see Table 7.5).

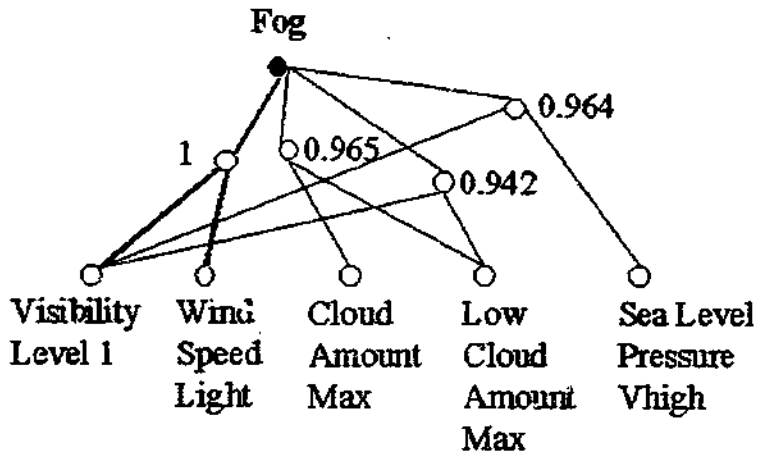

Figure 7.6: Pathways with higher confidence degrees leading to the fog class

For the not fog class the combinations with higher levels of confidence degree were *cloud amount medium* and *visibility level3* with a confidence degree of 1.0, *dry bulb high* and *cloud amount medium* with a confidence degree of 0.968, and *cloud* and *low cloud amount* both *medium* with a confidence degree of 0.913. Figure 7.7 illustrates the neural network pathways corresponding to these combinations.

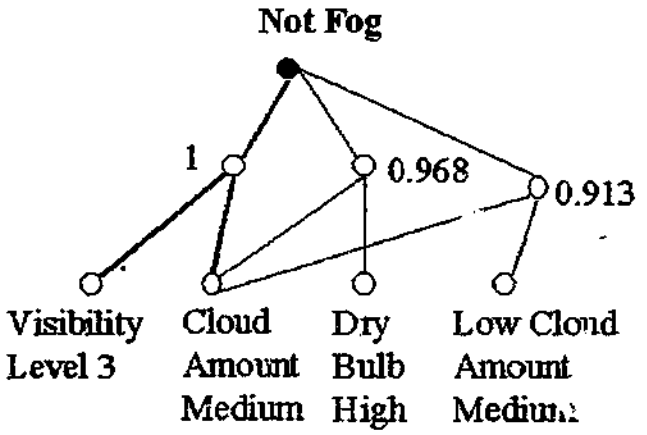

Figure 7.7: Pathways with higher confidence degrees leading to the not fog class

A similar conclusion from the fog class can be extended to the not fog class; in this case the high level of visibility (*level 3*) is strongly related to not fog occurrence.

The training set Train_Model2-60V370_V used in this experiment has 48 *not fog* instances with *visibility level3* (visibility between 35 and 40 kilometres), 5 *not fog* instances with *visibility level2* (visibility between 30 and 35 kilometres), 50 *fog* instances with *visibility level1* (visibility less than 30 kilometres). There are no instances of *not fog* class with *visibility level1* associated to them, and also there are no instances with *visibility level4* in both classes.

Through the experiments discussed in this section and the results described in Table 7.9 and Figure 7.5, it is possible to assess the relevance of visibility observation in fog classification. Although this conclusion does not represent any surprise, the experiments provided a quantitative idea of its relevance.

Regarding the possibility that the visibility observation could be considered as synonymous with fog occurrence, and as such removed from the observation database (or even replace the *Fog Type* attribute), this is not supported by the experiments conducted in this research.

Although the visibility observation was verified in the neural network pathways with a 100% confidence degree (value 1 in Figure 7.6 and 7.7), what surely indicates its relevance to the concept being classified, it is always associated with other weather observations. For instance, in fog class visibility is associated with *wind speed light, sea level pressure vhigh*, and *low cloud amount maximum*. In not fog class visibility is associated with *cloud amount medium*.

As such, in the experiments conducted in this research, a neural network pathway directly connecting the visibility observation to *fog* or *not fog* with a 100% confidence degree was not verified. It indicates that only visibility levels are not enough information to identify fog occurrence, in the scope of the problem addressed in this research.

7.6 Discussions

It is important to recall that the DM-NN computational model is proposed as an iterative and interactive environment for decision support, which defines a decision process and a computational framework linking diverse components in a decision support cycle (Figure 4.13 in Chapter 4 illustrates this concept).

The application of such an approach requires a series of activities in its diverse stages, for example, gathering information about a particular decision problem, analysing such information and preparing data, as well as choosing an adequate technology for mining data, evaluating outcomes and populating knowledge bases. The necessity of gathering new data or making changes in the domain are also considered (even expected), as discovered knowledge is likely to give new insights about new information to be collected or better ways to model the problem under study. For instance, weather observations initially not considered in this

research were suggested by weather forecasters during discussions about the preliminary results (refer to section 5.3.2, "Understanding Meteorological Data" for information about suggested weather observations).

Data mining by itself constitutes an interactive process, as demonstrated by the experiments conducted in this research, where several data mining experiments were performed with diverse settings of rule confidence degree, rule support degree and rule order.

Additionally, the IAS component is likely to require a series of interactions until it achieves its best performance or a stable level of performance, as problem situations are dynamic. Consequently, problem models are expected to change and adapt over time.

The model proposed in this research is said to be evolutionary, as NN is an evolutionary technology regarding its adaptive capabilities. An interactive and evolutionary execution cycle is normally expected in NN applications, until the system achieves a stable learning state. As the environment changes, the model needs to adapt to those changes.

What was discussed in Chapter 6 also has to be taken into account, i.e. how the domain of aviation weather forecasting was modelled and integrated into the CANN simulator. Different ways of modelling the problem might result in better performances that the ones achieved in this research. For instance, the experiments conducted with the visibility observation demonstrated that a higher performance was achieved when this information was included in the data mining models. Additional experiments with other weather observations could certainly result in better classificatory performance; even changing the morbidity values associated to the evidences (weather observations) could potentially improve performance.

The results obtained can be considered satisfactory for fog identification. It has to be recalled that fog phenomena is considered a severe and rare weather event, which is difficult to predict and where false alarms and incorrect forecasts are likely to occur (Keith, 1991). (Refer to section 5.2, "Issues in Aviation Weather Forecasting" for a discussion about weather forecasting at Tullamarine). According to a study developed by Keith (1991) forecasts for Tullamarine demonstrate poor performance for low stratus and fog. This study considers a 5 year means for various airport cities in Australia, taking the latest 5-month running means of

the probability of detection (POD) and false alarm ratio (FAR) for low cloud cases including fog. Tullamarine showed the worst POD with 69% and a FAR of 77%.

The experiments conducted through the DM-NN model, particularly when taking the visibility observation into account, resulted in a fog classificatory performance of 79.72% for Model2-60, with the best individual performance of 86.67% when applying Train_Model2-60V370_V. Figure 7.8 contrasts these results.

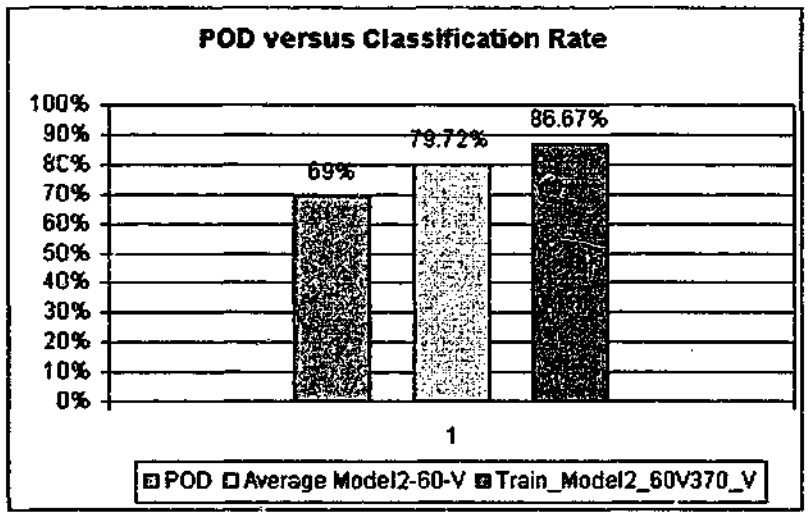

Figure 7.8: Contrasting the POD with classificatory rates

These results are indicative of a higher performance achieved by the DM-NN approach when contrasted with the results reported by Keith's study (Keith, 1991).

The experiments and results conducted in aviation weather forecasting and discussed in this chapter aim to provide ground to assess the feasibility and applicability of the DM-NN approach for decision support, rather than to come up with optimal results. The research described in this thesis concerns in achieving satisfactory results, where satisfactory results are defined by the user decision makers based on their own utility functions about the novelty or usefulness of the outputs given by the proposed approach for decision support.

It is important to remember that the individual performance of the data mining and neural network algorithms are not the main concerns in this research. This research project does not claim contributions on algorithms development and optimisations. This research is concerned with the combined approach, with the DM-NN IDSS model performance as a whole, its

usefulness, suitability and effectiveness in a decision making problem, rather than the performance of a single technology by itself.

Consequently, the results obtained through the classificatory performance have to be analysed taking all these characteristics into account. These results provide directions regarding the DM-NN model's applicability as a decision support framework for a data rich domain, rather than validating any particular algorithm.

7.7 Chapter Summary

This chapter presented the results of the DM-NN computational model performance in the context of aviation weather forecasting.

The performance of the DM-NN computational model was assessed in terms of its accuracy in identifying fog cases at Tullamarine. It is a quantitative approach, where the holdout method was applied to evaluate the DM-NN model through the classificatory performance on the testing data sets.

The analysis of the DM-NN model performance took into account the design employed to generate the data mining models and knowledge models, e.g. sampling strategy and the combinations of data mining parameters employed.

Each of these design issues was individually presented and discussed in this chapter. The sampling proportions were analysed to verify which proportions lead to a better performance. Additionally, the experiments conducted with distinct rule confidence degrees were analysed. For each level of rule confidence degree, the experiments done with different values of rule support and rule order were assessed and discussed.

Specific experiments conducted with weather observations were also discussed in this chapter.

Chapter 8

8 Discussions About the DM-NN Approach

The chapter discusses certain concepts of intelligent systems that relate to the DM-NN architecture. Furthermore this chapter contrasts the DM-NN approach with related works, and discusses some of its limitations.

8.1 Concepts Related to the DM-NN Architecture

This section discusses the DM-NN architecture according to the concepts of intelligent systems (IS) previously introduced in Chapter 2, taking into account the knowledge representation schema adopted, the type of reasoning implemented and level of integration among the components.

The operation mode of the DM-NN model follows a cycle, as discussed in Chapter 4. Cases are extracted from source databases, presented to a data mining component from which a series of association rules are obtained, and them those rules are accessed by a neural network based system for learning and consultation purposes.

Within that schema the DM-NN architecture employs diverse types of knowledge in its reasoning process: *specific* knowledge, *generalized* knowledge, and a type of *compiled generalized* knowledge.

Specific knowledge is represented as cases. Generalized knowledge is extracted from cases and represented as association rules; where rules are generalizations of specific cases. Compiled generalized knowledge results from the neural network learning process and is represented through the neural network structure, as has been discussed in Chapter 6.

In this architecture *specific knowledge* (cases) is used to acquire general expert knowledge (rules). *Generalised knowledge* (rules) is used to guide the neural network learning process. And *compiled knowledge* kept in the neural network is used for problem solving and to build explanations.

The approach of utilizing general and specific (cases) knowledge in a cooperative way has been studied by Aamodt (1995); this approach is grounded in Riesbeck and Schank's psychological theory (1989) which says that humans do not reason from cases when well-established rules are available, and rules are originated through the continuous repetition of a certain activity (Surma and Vanhoof, 1995; Riesbeck and Schank, 1989).

According to this cognitive theory, the human reasoning mechanism operates from the general to the specific when general knowledge is available. This theory considers two basic reasoning heuristics for problem solving. First, a rule based reasoning approach is employed. If there are no rules that cover the problem, then an attempt is made to remember a similar problem (case) that was solved in the past and adapt it to solve the current problem, thus a case based reasoning approach.

Rule based reasoning is the mechanism employed by most expert systems, often called *rule base expert systems* (Hayes-Roth and Jacobstein, 1994), and case based reasoning is the approach employed in CBR (case based reasoning) systems (Riesbeck and Schank, 1989). Although many CBR systems use both these reasoning approaches (Surma and Vanhoof, 1995). For example, the CABARET system described by Rissland (Rissland and Skalak, 1991) integrates rule base and case base reasoning, and the CREEK system (Aamodt, 1991) combines rule based and case based reasoning in a single framework. CREEK first attempts to solve a problem by case based reasoning, and if an acceptable match is not found, a rule based approach is triggered.

Although the DM-NN system does not claim cognitive contributions, its reasoning schema draws from Riesbeck and Schank's theory (1989). It follows a rule based reasoning approach, in which rules are mapped on to the neural network structure and used for problem solving. In this schema, cases are not used in problem solving but to generate rules (generalized knowledge).

Regarding the level of integration between the data mining and neural network components, the DM-NN architecture can be classified as a *loosely coupled system*, according to the intelligent system classification proposed by Medsker (Medsker and Bailey, 1992) and as an *intercommunicating hybrid system*, according to the classification proposed by Goonatilake (Goonatilake and Khebbal, 1995) (refer to section 2.2.4, "Intelligent Systems Classification.")

The proposed architecture can be classified as *loosely coupled* because both components communicate with each other via data files, where the data mining results are stored in knowledge rule bases; these are later accessed by the neural network component for training. The data mining component does not directly interface with the connectionist model.

Moreover, the DM-NN architecture can also be classified as an *intercommunicating hybrid system*, as each component is self-contained and individually performs a particular task, e.g., the data mining component is assigned to implement knowledge acquisition and the neural network component is assigned to implement learning and reasoning. There is no overlap between the roles of each component in the proposed architecture.

Furthermore, the neural network system applied relates more to the *fully integrated* classification proposed by Medsker (Medsker and Bailey, 1992), as CANN implements a computer system architecture that tightly integrates neural network models within an object oriented hierarchy (Beckenkamp, 2002). In this system architecture the neural network topology is symbolically represented as objects in a hierarchy of classes, in which a clear distinction between the connectionist approach and its symbolic representation is very difficult.

8.2 Knowledge Discovery Related Work

This section contrasts the DM-NN approach with some related knowledge discovery based systems, taking into account the architecture implemented, the data preprocessing phase, the data mining algorithms, and the utilization of discovered knowledge. The purposes of this study were to understand the architecture implemented by different authors and also to assess the limitations of the DM-NN model.

A knowledge discovery methodology used to automate knowledge acquisition from large databases was studied first. This methodology has two distinctive characteristics: firstly a large collection of potentially relevant patterns is discovered at once, and next, different views are formed on the discovered patterns in an iterative and interactive fashion (Klemettinen, 1999).

This methodology has been implemented as part of the TASA system (Telecommunication Alarm Sequence Analyser) (Klemettinen, 1999; Klemettinen, Mannila and Toivone, 1999). TASA is a data mining system that discovers recurrent patterns of alarms in telecommunication alarm databases and provides tools for interactive identification of relevant patterns. The purpose of TASA is to support the knowledge acquisition phase for creating alarm correlation patterns, which are further used in the construction of real-time alarm correlation systems. The types of patterns discovered by TASA are episode rules (Mannila, Toivonen and Verkamo, 1995) and association rules (Agrawal, Imielinsk and Swami, 1993).

The knowledge discovery methodology implemented in TASA covers the stages of data preprocessing, discovery, and presentation of discovered patterns, including postprocessing.

Within this methodology large collections of patterns are discovered at once, and iterative information retrieval methods are employed to provide various views of the discovered patterns. Such a knowledge discovery methodology emphasizes two central phases: a pattern discovery phase, finding all potentially interesting patterns according to some loose criteria, and providing flexible methods to present the discovered patterns, allowing iterative creation of different views of the discovered patterns. As such the emphasis of this KDD methodology relies on the presentation of discovered patterns (Klemettinen, Mannila and Toivone, 1999).

During the application's first phase raw data is collected and prepared for the discovery phase (data mining), and relevant features are selected. During the discovery phase all potentially interesting patterns are generated through loose interestingness criteria, for example, setting low values for frequency and confidence thresholds. The objective is to obtain large numbers of patterns, minimizing as much as possible the need for a new discovery process. The main role of the KDD methodology implemented in TASA is to provide tools for displaying and browsing discovered patterns, allowing efficient and interactive views of the patterns obtained. TASA allows users to interactively change rule

pruning thresholds, ranking rules according some criteria such as statistical significance, and even organize rules in structured ways. Additionally, users can define rule templates, which are expressions that describe the forms of rules being generated.

In the TASA application, a large database of alarms is firstly analysed, and in this step episode and association rules are automatically discovered. Then network management specialists analyse the discovered rules, and based on background and inferred knowledge, select interesting rules. Finally, the selected rules are converted into correlation rules that have been applied in real-time fault identification, alarm correlation systems, and network surveillance (Klemettinen, 1999; Klemettinen, Mannila and Toivone, 1999).

There are similarities between the DM-NN model for IDSS described in this thesis and the KDD methodology implemented in TASA. Both approaches aim to automate knowledge acquisition to a certain degree and an interactive cycle is proposed within the application of both approaches. This interactive cycle is due to the recognition that expert background knowledge is an essential part of any KDD process.

On the other hand TASA focuses on knowledge presentation and its further integration into another device, such as the alarm correlation system. The DM-NN model focuses on knowledge acquisition and how to make this process as automated as possible. A smooth and direct integration into a complementary system, called the intelligent advisory system (the IAS component) is also a major goal. In such an approach, the presentation of discovered knowledge is not a major concern as it is in TASA, although it has been contemplated in the DM-NN model. Section 4.7 discusses model functionality and Figure 4.14 illustrates that functionality.

Furthermore, the DM-NN model gives special attention to the utilization of discovered knowledge, with the integration of a complementary component into the KDD process and the functionalities deployed by such a component. Specifically, the IAS component (refer to section 4.3, "The DM-NN architecture") is proposed as an integrated element into the DM-NN architecture. Within this approach the DM-NN model contemplates the whole KDD cycle from the preprocessing level to a fully integrated level with a complementary component in its architecture, e.g. the intelligent advisory system implemented through CANN.

A second work that was studied was a meteorological and data mining environment called MADAME (Buchner et al., 1998). MADAME was primarily designed for meteorological forecasting of high-intensity rainfall in Hong Kong. MADAME was designed as a multilayer, open and extensible architecture compounded by three main elements: a meteorological data warehouse, a knowledge modelling and a knowledge base. MADAME incorporates diverse types of meteorological data and supports machine-learning algorithms for data mining.

MADAME is able to deal with different types of data, such as images obtained from radar and satellite pictures, as well as text data such as wind speed and directions, and various temperature measurements. Hence, data extraction from diverse sources, data transformation, and loading operations are implemented in this architecture, including the design and maintenance of a meteorological data warehouse. Besides historical data, online readings such as rainfall levels, wind and air measurements are also loaded into the data warehouse on a regular basis.

The knowledge modelling component and the knowledge base constitute the knowledge discovery component of MADAME. The MADAME architecture has been designed to be open and extensible, in which different algorithms for data mining can be applied in diverse experiments. Meteorological experiments using the general rule induction algorithm (GRI), classification trees algorithms, neural network models such as Backpropagation, and hybrid approaches of these algorithms have been reported with promising results by Buchner (Buchner et al., 1998).

The third component of MADAME's architecture is a knowledge base, which incorporates domain knowledge. The major objective of incorporating such knowledge is to reduce problem dimensionality and improve the quantity and quality of domain knowledge.

The outcomes (discovered patterns) are used for meteorological analyses and predictions. Furthermore, MADAME can be connected to an existing meteorological prediction system to provide complementary information, or it can be used as a stand alone application. A series of meteorological experiments have been conducted and reported in (Buchner et al., 1998) to assess the applicability of the MADAME approach as a meteorological knowledge discovery environment.

MADAME implements a meteorological KDD environment that includes a meteorological data warehouse and data mining components, including a knowledge base. The basic elements of this architecture are similar to the DM-NN model, mainly in the use of a data warehouse as the data source for the data mining component. The DM-NN model architecture differentiates from MADAME mainly in the utilization of discovered knowledge, as the integration of a complementary component into the KD process is a core issue in the DM-NN model. It is also contemplated in MADAME, but the system can also be used as a stand alone application.

A third work that was studied is a toolkit for knowledge discovery (KD toolkit) that was initially applied in discovering patterns describing average temperatures in northern France. This experiment consisted of generating a set of rules that were used for meteorological predictions in mid-long term forecasts (Howard and Rayward-Smith, 1998). The toolkit comprises features for data preprocessing and data mining, together with database and data analysis functionalities (Howard and Rayward-Smith, 1998).

The database features include data manipulation techniques such as sorting, searching, appending records and fields, data sampling techniques and field selection, among others. The data preprocessing features include simple two dimensional graphs visualization, data histograms and tabular views. Data discretization techniques and missing data summaries are also included as part of the data preprocessing component.

According to Howard (Howard and Rayward-Smith, 1998), the data mining component consists of a rule induction approach, which is implemented through a simulated annealing algorithm and hill-climber search engine. Other induction algorithms are externally available in the toolkit. For data analysis the toolkit provides a rule editor for visual editing of the extracted rules, and it also allows the user to conduct experiments with rule properties, such as selecting or deselecting features in rules (itemsets).

Additionally the KD toolkit implements specialised features for the meteorological domain, such as a series of specific data conversion routines and graphical facilities (Howard and Rayward-Smith, 1998).

One of the particularities of the KD toolkit is that it provides its own features for data manipulation and preprocessing, instead of relying on third party databases. In this way the KD toolkit seems to be a more self-contained approach, with basic features available in the toolkit. For discovered patterns the KD toolkit implements a rule editor facility for rule editing and manipulation. Additionally, the possibility of integration of an existing system within the KD toolkit would be an interesting way to enhance the toolkit capabilities, especially from the utilization of discovered knowledge.

The last work studied was an association classification framework named Classification Based on Associations (CBA). CBA integrates an association rule mining algorithm with a classification rule mining algorithm, and was proposed by Liu (Liu, Hsu and Ma, 1998). The integration in this framework is done by mining a special subset of association rules that the authors called *class association rules* (CARs). An association rule mining algorithm based on the Apriori algorithm (Agrawal et al., 1996) was adapted to generate all the CARs that satisfy user-specified minimum support and confidence constraints (Liu, Hsu and Ma, 1998). Further, a classification algorithm is applied for building a classifier based on the set of discovered CARs. Essentially, the CBA framework aims to build an accurate classifier for prediction from the set of generated rules (CARs). The CBA consists of two main parts: a rule generator (CBA-RG) algorithm for finding association rules, and a classifier builder (CBA-CB). The CBA-RG algorithm is adapted from the Apriori algorithm (Agrawal et al., 1996), and it generates all the frequent set of items (rule items) belonging to each class in the training data set by making multiple passes over the data; where "frequent" indicates that a particular item satisfies a minimum predefined support degree. The basic idea of the CBA-CB algorithm is to choose a set of high precedence rules from the set of CARs to cover the training data (Liu, Hsu and Ma, 1998).

The CBA-CB algorithm performs three steps: first it sorts the CARs rules according to a precedence criteria, then it selects the sorted sequence of rules and searches by the cases in the training set that are covered by the selected rules. Rules that successfully classified one or more cases are marked to become part of the classifier, otherwise they are discarded. When classifying a new case, the first rule that satisfies the case will classify it, and if there is no rule

that satisfies the new case, the algorithm takes on a predefined default class. When there is no rule or training case left, the rule selection process is completed. A detailed discussion about the CBA algorithm can be found in (Liu, Hsu and Ma, 1998). The author reported experimental results showing that the CBA algorithm was in general more accurate than that produced by the classification algorithm C4.5.

The CBA framework is a different kind of work from what has previously been discussed in this section. In the CBA framework there is no explicit emphasis on the KDD process, but in building an accurate classifier from induced sets of association rules. This is a similar approach to the one implemented in the DM-NN model, as both aim to build classifiers for prediction based on association rules. While the CBA framework integrates an association rule mining algorithm with a classification rule mining algorithm, the DM-NN model integrates an association rule mining algorithm with a neural network model. The class association rules defined by Liu (Liu, Hsu and Ma, 1998) are equivalent to the set of association rules that populate the knowledge rule bases in the DM-NN model. Besides employing the same knowledge representation schema (association rules) both approaches also rely on levels of confidence and support degree as the main criteria for rule filtering.

The CBA framework consists of two main parts: a rule generator algorithm for finding association rules, and a classifier builder. Similarly, the DM-NN model has two main stages: first, descriptive models are built, and then predictive models are built based on these descriptive models. Although the CBA framework and the DM-NN model share a similar approach in building descriptive models (generating association rules), they differ in the way the classifiers are built. The CBA framework selects sets of high precedence rules from the set of CARs that cover the cases in the training data, and the DM-NN uses a set of rules as a training set for a neural network model; thus, rule selection is performed through the neural network pruning and reward algorithm.

Additionally, issues of data sources and data preprocessing that are not specifically addressed in the CBA framework have a definite importance in the DM-NN model.

The works that were studied and introduced in this section bring valuable insights for the development of computerized solutions for decision support based on the knowledge discovery in databases approach.

Firstly, almost all the discussed works somehow address the data preprocessing stage. Solutions vary from the implementation of a dedicated data warehouse as proposed in the MADAME architecture (Buchner et al., 1998), to the implementation of specific database functionalities for data preprocessing and data visualization features, as implemented in the KD toolkit (Howard and Rayward-Smith, 1998). Regardless of the strategy employed, data preprocessing is a relevant activity in almost any knowledge discovery in databases system, and the experiment in aviation weather forecasting described in this thesis consolidates that conclusion.

In the DM-NN architecture, a decision-oriented data repository is introduced as the primary source of information, and ideally, it should be a data warehouse.

Rule induction seems to be the predominant formalism for data mining and the representation of discovered patterns in the works introduced in this section, although decision trees are also contemplated. Different applications and systems apply a range of rule induction approaches, such as episode rules and association rules as in TASA, the general rule induction algorithm implemented in MADAME, the rule induction approach implemented through a simulated annealing algorithm in the KD toolkit, and association rules implemented in the CBA framework which is similar to that implemented in the DM-NN system.

The need to use different approaches for knowledge discovery is also an important feature. For instance, MADAME implements an open and extensible data mining component with the aim of supporting diverse machine learning algorithms (Buchner et al., 1998). The KD toolkit also provides a hill-climber algorithm for data mining and induction algorithms available through external links.

As for the presentation of discovered patterns, TASA provides tools for displaying and browsing rules, the KD toolkit implements a rule editor for visual editing of the extracted rules, and MADAME provides a domain knowledge component in which interesting discovered patterns can be stored for further analysis. The DM-NN architecture incorporates

a rule evaluation stage, although this was not implemented in the application in aviation weather forecasting described in this thesis.

Regarding the utilization of discovered knowledge, almost all authors address the integration of a complementary system into the KDD process. TASA integrates an alarm correlation system within its architecture (Klemettinen, 1999). MADAME includes a knowledge base to store discovered knowledge and a further connection to existing meteorological prediction systems within its architecture is a possibility, although the system can be used as a stand alone application. And the CBA framework applies discovered rules to build a classifier. Although the authors of the CBA framework (Liu, Hsu and Ma, 1998) do not specifically argue that the framework was conceived within a knowledge discovery process point of view, CBA consists of a rule generator (CBA-RG) algorithm for finding association rules and a classifier builder (CBA-CB). In this approach, the rule generator is the data mining component of CBA framework, and the classifier builder is where the discovered knowledge is utilized.

In contrast with the works discussed in this section, the integration of a complementary system into the knowledge discovery process is an important goal in the DM-NN architecture. The purpose of this architecture is to devise a model for decision support systems through the combination of data mining and intelligent computing technologies, specifically neural networks. Data mining is applied for knowledge acquisition, and neural networks are applied to implement learning and reasoning capabilities. In this architecture the emphasis is not specifically on the knowledge discovery process, but in the amplification of that process on the utilization of the knowledge discovered. From that perspective the DM-NN architecture draws from the ideas of intelligent systems, where two or more technologies are combined to overcome limitations of each other, endowing the resulting systems with capabilities that would be very difficult to implement within a single approach.

8.3 Improvements and Limitations

The previous section of this chapter, together with section 2.5, "Applying Intelligent Systems", comprise the study of works related to the DM-NN model proposed and developed

as part of this thesis. The purpose in presenting and discussing them was not to give an exhaustive list, but rather a comprehensive enough sample of the current state of research and development in the field of intelligent systems and knowledge discovery based systems. Furthermore, the author aimed to learn and consolidate his own knowledge in the related fields.

The works discussed, together with the experience gained with the implementation developed in aviation weather forecasting as part of this research helped to assess some improvements and limitations of the DM-NN model for decision support.

The hybrid DM-NN architecture introduced certain facilities related to knowledge acquisition, representation and utilization of specific and general knowledge, as well as implementation of learning and reasoning in a decision support framework. However, there are limitations and costs associated to the proposed model, which will now be discussed.

In the DM-NN architecture, a decision-oriented data repository is introduced as the primary source of information. Ideally, it should be a data warehouse. A meteorological data warehouse was not available in the implementation done in aviation weather forecasting as part of this research. Therefore a significant effort in data preprocessing was needed, as discussed in Chapter 5. The availability of a meteorological data warehouse would significantly minimise the intensive work and time in data preprocessing that was required.

The DM-NN system was successfully able to discover interesting rules about aviation weather forecasting, but does not provide facilities for editing, browsing, or any kind of visualization of the discovered knowledge. While a stage of rule evaluation is suggested in the model application, the emphasis relies on the generation of association rules (domain knowledge) and its further access by the neural network system. Although knowledge acquisition is the primary role of the data mining component, the capability to access and manipulate the discovered knowledge would represent an improvement in the model.

As already mentioned, the proposed architecture is classified as a *loosely coupled system* because both components communicate with each other via data files. Therefore, the data mining component does not directly interface with the connectionist model. Hybrid architectures bring improvements in overcoming limitations of single technologies, hence

empowering systems with better capabilities. But such improvements also bring additional costs, such as the necessity to coordinate diverse components, mainly in architectures where the components are loosely integrated.

The weak integration between the DM and NN components requires a significant amount of work in data transfer. This is a time consuming and error prone situation. For instance, a complex set of procedures was required to transform the data sets obtained through data mining into the final data sets used by CANN in the application developed in this research. The knowledge bases obtained had to be exported from MS Access tables to ASCII plain files in order to be read by CANN, and the attributes had to be carefully adjusted. A significant number of SQL queries and MS Visual Basic program modules had to be implemented to perform this data integration.

Besides that, a set of program modules were implemented and executed in MS Visual Basic to compute the discretization schema discussed in section 5.4.1.4. New attributes had to be created in the database, and each attribute had to be properly adjusted in the correct position and size, as this is necessary when exporting MS Access tables to ASCII files.

It is worth pointing out that a great variety of data formats and software programs were used in this research, from when the first database with weather observations was delivered by BOM until the last stage of this research. Some of these were MS Access and .DBF tables, MS Excel spreadsheets, the SPSS statistical package and its own proprietary data format, and ASCII plain text files.

Furthermore, the weak integration between the data mining and neural network components restricted the usability of the discovered patterns to a single reasoning schema. This also brings problems regarding the redundancy of information, as the information has to be simultaneously represented in the neural network structure, as association rules, and also in the case bases. Rules are mapped on to the neural network structure; both formalisms keep the same type of knowledge. Moreover, cases stored in the case bases are another way of representing the same classificatory knowledge. This redundancy of information requires a control mechanism to ensure that changes are propagated accordingly. For example, if a new feature is inserted or removed as part of the data mining component (or even changes in the

discretization of a particular feature), that change has to be propagated to the neural network structure, even adding or removing the feature from the domain, or deselecting it from the list of evidences associated to its class.

The problem domain has to be modelled into the CANN object hierarchy, as described in Chapter 6. This domain modelling is stored in the object hierarchy implemented in CANN, and is therefore not easily available for other applications; furthermore, changes in the domain need to be propagated into CANN, which might require the execution of a new neural network training procedure.

Specifically with the CNM neural network model, care should be taken to avoid an excessively large combination of input nodes. This leads to the problem of combinatorial explosion, especially in applications where there is a large number of evidences as well as classes. A user should not attempt to combine all the evidences in the domain in all possible ways, for each class. Instead, only evidences observed for a particular class should be selected and used to build the clusters for that class (refer to section 6.3.3, "Modelling Hypotheses" in Chapter 6). Although the CNM implementation in CANN has been optimised to minimise some of the drawbacks of the original CNM model implementation, combinatorial explosion has to be taken into account when performing domain modelling.

Certainly, most of the limitations discussed above can be overcome. A loosely coupled architecture brings several disadvantages, but it also brings several clear advantages. For instance, each component in a loosely coupled architecture can be more easily replaced than in tightly coupled or fully integrated architectures. Normally, loosely coupled architectures are more flexible, and as such, easier to adapt. In this context, it should be observed that the DM-NN model for IDSS does not enforce any particular technology, but rather proposes a framework that can be seen as an infrastructure that can be built according to the needs of a particular problem. As such, a particular component can be replaced to suit the needs of a specific problem.

From this perspective the inclusion of a data warehouse is not a difficult task. In the case of the experiment developed in aviation weather forecasting a data warehouse was not used only because it was not available from BOM at the time it was developed.

The coordination between the data mining and IAS components, as well as the flow of data can be achieved through the implementation of a high level manager component. Such a component would coordinate the actions and events in the DM-NN architecture.

As for the limitations of the CNM neural network model, the first approach is certainly careful domain modelling and dimensionality reduction, to avoid unnecessary features. Besides that, a mechanism to generate only relevant clusters of evidence can be applied. For example, Machado (Machado, Rocha and Denis, 1992; Reategui, 1997) described an approach where genetic algorithms are used to generate and select the most significant clusters associated to a class.

8.4 Chapter Summary

This chapter discussed various concepts of intelligent systems that relate to the DM-NN architecture. The cognitive plausibility of the reasoning mechanism employed by the DM-NN systems was discussed, as well its classification according to the level of integration among its components and their functionalities.

Some related knowledge discovery based systems were introduced and contrasted with the DM-NN architecture, taking into account the architecture implemented, the data preprocessing phase, the data mining algorithms, and the utilization of discovered knowledge. The purposes of this study were to understand and learn from the architecture implemented by different authors.

Furthermore, some limitations and issues of improvement of the DM-NN architecture were discussed as part of this chapter.

Chapter 9

9 Conclusions

This chapter presents the conclusions and contributions of the research described in this thesis, and indicates possibilities for future research.

9.1 Introduction

The research conducted and described in this thesis has been concerned with investigating the combination of knowledge discovery in databases and intelligent computing technologies for decision support. As a result of this investigation, this thesis has proposed a new framework for decision support, combining data mining with artificial neural networks in a hybrid architecture, called the DM-NN model. This architecture employs an association rule generator algorithm for data mining to build domain knowledge from organizational databases, and a neural network based system is used to implement reasoning and problem solving. To assess the applicability of the proposed DM-NN model in practice it was applied to aviation weather forecasting, to identify the occurrence of fog phenomena at Tullamarine.

The DM-NN model for decision support has been presented and evaluated in this thesis. The achieved results, summary of contributions and possibilities for future research are discussed in the next sections.

9.2 Results

The research question as outlined in Chapter 1 was:

What are the components of a framework for intelligent decision support system capable of:

- *utilizing organizational databases as source of information*
- *facilitating automatic knowledge acquisition from those organizational databases*
- *reasoning and learning upon this knowledge*

to support decision making ?

The conceptual studies undertaken in this research contributed to define the architecture of the proposed framework for intelligent decision support, and its components.

The results of this research demonstrated that these components include a decision-oriented data repository such as a data warehouse as the framework's primary data source, case bases that store selected cases from the data warehouse, a data mining component to build domain knowledge from the cases stored in case bases, and knowledge bases to store the knowledge built through data mining.

The framework also includes an intelligent advisory system capable of learning from the knowledge bases, reasoning upon the knowledge learned, and issuing recommendations and drawing justifications.

An architecture that implements the proposed framework was developed in this research.

The results achieved in this research demonstrated that a combination of data mining through an association rule generator algorithm and artificial neural networks was capable of providing a model for decision support systems that automatically builds domain knowledge from organisational databases, and performs learning and reasoning upon that knowledge in supporting decision making.

It has been possible to conclude that:

- The DM-NN model constitutes a suitable technology to implement the IDSS framework. Its performance in identifying fog phenomena demonstrated its potential applicability as a decision support framework. From Table 7.2 it is possible to observe that the DM-NN model achieved average performances of 74.58%, 70.83%, 70% and 64.72% of correctly classified cases according to the data models obtained. From Table 7.9 it can be observed that the highest performance measure was verified when the visibility attribute was included in the

mining and training sets, with 86.67% of correctly classified fog cases, and an **average** performance of 79.72% correctly classified fog cases.

- There is a need for a decision-oriented data repository such as a data warehouse as the primary data source in the DM-NN model. This has been consolidated through the application in aviation weather forecasting. The unavailability of a meteorological data warehouse in the aviation weather forecasting implementation demanded a great amount of time and effort in data preprocessing, and in building the case bases. This therefore compromised the gain obtained in automatic knowledge acquisition through data mining. The experience gained in this research, as well as the discussion of the literature and related knowledge discovery systems presented in Chapter 8 led to the conclusion that a data warehouse is a significant component in any knowledge discovery based system, in order to minimise problems related to data quality and data preprocessing.

- Data mining through an association rule generator algorithm showed a promising and effective approach to facilitate automatic knowledge acquisition from organizational databases. The rule sets obtained (discussed in Chapter 5) and the performance of the proposed model (discussed in Chapter 7) showed that the data mining algorithm employed was able to induce relevant domain knowledge from cases obtained from the meteorological database used in this research.

- The adaptive, learning and generalization capabilities of neural networks facilitated *learning*, *reasoning* and *problem solving* in the proposed architecture. Neural networks proved to be an efficient technology to build generalizations from association rules and to implement reasoning upon that knowledge. Furthermore, neural networks were able to generate satisfactory predictive models about fog occurrence, according to the performance assessment. The best predictive performance averages achieved were 74.58% (refer to Table 7.2) and 77.29% (refer to Table 7.9) of correctly classified cases (both fog and not fog classes), which can be considered satisfactory for fog phenomena, as was discussed in Chapter 7. Additionally, based

on published results, Figure 7.8 contrasts the probability of detection for fog with the DM-NN classificatory performance.

- The Combinatorial Neural Model (CNM) was able to successfully build chunks of *domain knowledge from sets of association rules.* Figure 7.6 and Figure 7.7 illustrate pathways created by the CNM learning algorithm. Furthermore, its compatibility with the knowledge representation formalism of knowledge graphs and association rules facilitated the integration of knowledge represented through rules into its topology. The availability of a data fetcher for relational databases in CANN would significantly improve this integration.

- The CANN simulation environment greatly facilitated *domain modelling*, and implementation of *reasoning, recommendations* and *explanation* capabilities in the DM-NN model. Because CANN is an environment dedicated to the implementation of neural network models, it proved to be a useful approach to implement *learning* and *reasoning capabilities.* Besides that, there are two characteristics in its architecture that are particularly relevant for the purposes of this research: first, CANN was designed to support problem solving and domain modelling in classification problems (Pree, Beckenkamp and Rosa, 1997). The problem domain was easily modelled into the simulator, and a user can interact with the system adding, removing, selecting or deselecting features and classes. This capability allows interactive and iterative characteristics in the DM-NN model. Second, its dual knowledge representation schema helps to overcome one of the major drawbacks in artificial neural networks, which is the difficulty in accessing the knowledge distributed in the neural network structure. This dual knowledge representation schema facilitated the implementation of *explanatory capabilities* in the DM-NN model.

- The DM-NN model proved to be capable of *incremental learning* through the incorporation of new cases in the case bases, generating new sets of association rules and training the neural network with these new rules. Furthermore, the neural

network outputs can be stored as new rules in the knowledge bases, and the neural network can be retrained to learn about these new rules. This is a type of reinforcement learning implemented through a feedback learning approach.

Specific studies about data sampling for data mining and rule filtering thresholds were also conducted as part of this research, and the results obtained from these studies give guidance in developing knowledge discovery applications with similar characteristics to that of aviation weather forecasting.

The meteorological database used in the experiment developed in this research showed a low prevalence classification problem. It had approximately 50000 instances, with two classes representing the survey variable in which the positive class (fog) represented only about 2% of the population and the negative class (not fog) represented the other 98% of that population.

Considering knowledge discovery in database applications with similar characteristics to the application developed in this research, e.g., database size, population distribution, the nature of the problem being addressed, and applying a data mining algorithm with similar characteristics to the Apriori algorithm (Agrawal et al., 1998) for association rules, the research findings are:

- The results obtained through the analysis of performance indicate that the data sampling design elaborated in this research was capable of efficiently capturing information from both classes, fog and not fog. It indicates that in data mining applications where the original population presents a low prevalence classification problem, similar to that found in this research, a *stratified* sampling approach to separate both classes combined with a *multi-stage sampling* approach conducted in the majority class potentially constitute a suitable sampling strategy.

- As for data partitioning, the results discussed in section 7.3, and illustrated in Table 7.2 and Figure 7.1 show that the best classificatory performance was achieved by the data model obtained through a sample of 20% of the majority class population. The data set for data mining was obtained through a subsample of 60% of the majority class sample, together with a sample of 85% of the minority class

population. These results indicate that similar sampling proportions potentially constitute an appropriate choice for data partitioning in knowledge discovery in database applications with similar characteristics to the ones presented in this research.

- According to the results presented in Table 7.8 and Figure 7.2, it can be observed that the data mining experiments with a 70% rule confidence degree produced the best results, with averages of 0.28 of error rate and 71.77% of classificatory performance. In the experiments conducted in this research the training sets obtained from data mining experiments using a 70% rule confidence degree provided the higher number of rules for both classes, compared with training sets obtained from the same data models using different levels of rule confidence degrees. Figure 7.2 illustrates how the error rate increases as the rule confidence degree progresses upwards from 70%. This association can be related to the amount of association rules obtained, as increasing the levels of confidence degree implies a more constrained parameter to the association rule generator algorithm, consequently generating fewer rules. These results support the conclusion that normally error rates decrease as the size of training sets increase when using inductive algorithms for classification. Furthermore, this result indicates that levels comparable to 70% can be considered *suitable thresholds for rule confidence degree*, in experiments with similar characteristics to the one developed in this research.

- According to the results presented in Figure 7.3 and Figure 7.4, it can be observed that the data mining experiments using the configuration of 6% of minimum rule support combined with a maximum rule order threshold of 10 itemsets produced the best results, with averages of 0.28 of error rate and 71.84% of classificatory performance. The reason for this is possibly because the training sets obtained from the experiments using this configuration presented higher numbers of association rules and a more equally balanced class distribution than the training sets obtained using a different configuration. These results support the conclusion

that the classificatory performance of inductive algorithms relates not only to the size of training sets, but also to the distribution of classes within them. Performance is expected to increase as the size of training set increases, and as the distribution of classes in the training set becomes more homogeneous. Furthermore, these results indicate that similar levels to 6% for *minimum rule support* and 10 for *maximum rule order* can be considered *suitable thresholds* in experiments with similar characteristics to the one developed in this research.

Based on the above conclusions, this work has achieved its goals, which were:

- To devise a framework for intelligent decision support system (IDSS) capable of automatically building specific domain knowledge from data rich domains and applying this knowledge in problem solving
- To specify the appropriate technological components of that framework
- To empirically verify the framework in practice.

9.3 Summary of Contributions

The main contributions of this research are:

- the conception and design of a new framework for decision support based on knowledge discovery in databases and intelligent computing technologies, identifying its technological components, roles and respective relationships.
- the development of a hybrid model for intelligent decision support systems that implements that conceptual framework, through the combination of an association rule generator algorithm for data mining and a neural network model. This hybrid model benefits from the inductive capability of an association rule generator algorithm to build domain knowledge from databases, from the learning and reasoning capabilities of neural networks, and from the explanatory capability of a symbolic-connectionist knowledge representation schema.
- a description and empirical validation of the knowledge discovery in databases process from both practical and analytical points of view. The knowledge

discovery process developed in this research contains several stages: the application domain modelling, preparing the data set, mining data, data postprocessing, and integrating a complementary computerized system. Furthermore, the description of the stage of knowledge discovery in this research provides guidance for data partitioning and selection of thresholds for rule confidence, support and rule order in data mining applications.

- provided a technological approach in the hybrid DM-NN model that allows access to the hidden knowledge stored in the neural network structure. This approach facilitated explanatory capabilities in the proposed model for decision support, which otherwise would be very difficult to achieve. It should be noted that the symbolic representation of the knowledge implicitly stored in neural network structures has been a subject of current research in the areas of artificial neural networks and intelligent systems, as discussed in Chapter 2 (refer to section 2.3, "Hybrid Symbolic-Connectionist Systems.") It represents one of the major difficulties in applying neural network technologies in computerized information systems that need to deliver human readable outputs, such as decision support systems.
- the application of the proposed model for intelligent decision support system in aviation weather forecasting. The model was developed covering all stages of knowledge discovery, i.e. building descriptive models of the domain, and building predictive models of that domain through the application of a neural network based system. It demonstrated the applicability and effectiveness of the proposed framework in practice, bringing contributions to the practice and theory of intelligent system and decision support systems.

9.4 Future Work

As a result of the work developed in this research, some subjects for future research were identified and presented in this section. These subjects were classified in four sections:

extending the DM-NN architecture, implementing reasoning and knowledge representation approaches, data preparation and modelling and DM-NN model applicability.

- **Extending the DM-NN architecture**

An interesting subject for future research is investigating the possibilities of expanding the DM-NN architecture towards a more integrated approach. One of the possibilities in that direction would be to coordinate the interactions between the diverse components of the architecture. This can be achieved by integrating a manager component in the DM-NN architecture. For instance, such a manager component would be able to extract cases from the data repository, store them in case bases and then notify (or even activate) the data mining component that new knowledge is available for learning (mining).

Moreover, the manager component would coordinate the operation between the data mining and neural network components. For example, it should notify the neural network system when there is new knowledge available for training, and the neural network system would execute a new training procedure.

Another interesting functionality would be the ability to automatically store neural network outputs back in the knowledge bases, when these outputs indicate novel information according to user criteria.

The implementation of such a manager component would make the current architecture more flexible, with a more intelligent and autonomous coordinating mechanism. This mechanism would lead to some interesting research issues to the areas of intelligent and autonomous agents and distributed computing.

Another possibility is to investigate the extension of the CANN framework, implementing other reasoning mechanisms besides the connectionist one. One possible approach is to plug into the framework a new hierarchy of classes to implement other machine learning algorithms, such as the Apriori algorithm (Agrawal, Imielinsk and Swami, 1993).

Another interesting research subject would be the definition of structures to represent domain modelling and domain knowledge using a metadata schema, and implementing that schema in XML. This would facilitate a fully integrated architecture for the DM-NN model, and would also give access of that domain modelling and knowledge to other computer

systems. It also would facilitate changes in domain modelling, bringing a significant higher level of flexibility into the DM-NN architecture.

- **Implementing reasoning and knowledge representation approaches**

Expanding the DM-NN architecture also brings the possibility of investigating alternative reasoning approaches besides the connectionist. For example, a more traditional rule based reasoning approach could be implemented using the knowledge bases. Similarly, cases stored in case bases could be used to implement a case based reasoning schema. The implementation and integration of these diverse reasoning approaches in a single framework raises interesting and challenging research issues not only from the computational perspective but also in cognitive plausibility, suggesting insights on how the model could be improved from a cognitive perspective.

According to the reasoning mechanism employed in the DM-NN model, rules are obtained from cases and accessed by neural networks for learning and reasoning purposes. As such, rules can be considered generalizations of sets of cases, representing a kind of "condensed" knowledge derived from cases. Thus, it seems reasonable to be able to assign a measurement to rules, representing this "condensed" type of knowledge. In CANN a morbidity value (refer to section 6.3.2, "Modelling Evidences") is assigned to each evidence in the domain and then used as a weighting factor to the input neurons, but there is no mechanism to assign a measurement to specific groups of evidences, because CANN does not differentiate rules from cases and treats them in the same way. A mechanism to differentiate rules from cases would represent a more realistic reasoning approach when training the CNM with rules instead of cases, and it would seem to be more realistic from a cognitive point of view.

The implementation of such a mechanism, and a comparative study of learning from cases and learning from rules, would constitute interesting research issues.

The use of other data mining algorithms in a comparison study would also be an interesting research topic, besides potentially improving the DM-NN applicability. For instance, the use of episode rules (Mannila, Toivonen and Verkamo, 1995) instead of

association rules, or a combination of both approaches in the meteorological and other domains, constitutes an interesting topic to investigate.

Rule evaluation is not considered in the current DM-NN implementation; and the availability of such facilities for rule browsing, evaluation and visualization would certainly represent an improvement to the current architecture.

- **Data preparation and modelling**

Among the most difficult questions that arose during this research were those that related to the necessary number of cases that were sufficient for data mining and training, and to define suitable sampling strategies. The experience gained in this research indicates that these questions are empirically defined, and depend on the nature of the problem being addressed, as well as knowledge about the problem and the information available. To create mechanisms and theories to help these tasks to be automatically performed seems a difficult and exciting challenge for future research.

The work carried out in data preparation and transformation of meteorological data can be useful in further experiments applying data mining, or inductive learning algorithms, in the meteorological domain. In that context, a potential research topic would be to investigate the construction of a library of program modules for meteorological data preprocessing. Such a library could be used together with a meteorological data warehouse, or even an ADAM module, to facilitate further knowledge discovery experiments. In that perspective, the application the DM-NN model for the prediction of other hazardous weather events, such as tropical cyclones and severe thunderstorms, is a possibility for further research.

Another interesting research topic refers to data modelling in meteorology. For instance, the use of fuzzy logic to model and represent semantic variables such as wind speed and direction potentially could avoid oversimplifications when representing meteorological continuous information through sharply defined categorical data.

- **DM-NN model applicability**

From an engineering point of view, it would be interesting to investigate the possibility of having an IAS module (a trained neural network module) inside an aircraft and used as a simulation instrument. For instance, some weather conditions could be communicated to the

IAS module (for example by the crew) in order to verify the likelihood of occurrence of a particular weather event, such as fog.

The application of the DM-NN framework to other domains and problems would help to assess its applicability in different scenarios, raising further issues for improvement in the model, and potentially create new research topics in the areas of intelligent systems, knowledge discovery and decision support. Furthermore, the author believes that incorporating different reasoning schemas and possible different knowledge representation approaches in the framework may contribute to the investigation of cognitive models observed in humans.

Bibliography

Aamodt, A. (1991). *A knowledge intensive, integrated approach to problem solving and sustained learning.* Department of Computer Science, University of Trondheim, Norway. Ph.D (Dr.Ing.) Thesis.

Aamodt, A. (1995). Knowledge Acquisition and Learning by Experience - The Role of Case-Specific Knowledge. In (Ed, Tecuci, G. and Kodratoff, Y.), *Machine Learning and Knowledge Acquisition, Integrated Approaches,* (pp. 197-245). Academic Press.

Ackley, D., Hinton, G. and Sejnowski, T. (1985). A learning algorithm for Boltzmann machines. *Cognitive Science,* 9, 147-169.

Adamo, J. (2000). *Data mining for association rules and sequential patterns: sequential and parallel algorithms.* New York, USA: Springer-Verlag.

Agrawal, R., Gehrke, J., Gunopulos, D. and Raghavan, P. (1998). Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications. *ACM SIGMOD Conference on Management of Data (SIGMOD'98),* (pp. 94-105). Seattle, Washington DC. ACM Press

Agrawal, R., Imielinsk, T. and Swami, A. (1993). Mining association rules between sets of items in large databases. *ACM SIGMOD Conference on Management of Data (SIGMOD'93),* (pp. 207-216). New York, NY: ACM Press.

Agrawal, R., Mannila, H., Srikant, R., Toivonen, H. and Verkamo, I. (1996). Fast Discovery of Association Rules. In (Eds, Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P. & and Uthurusamy, R.), *Advances in Knowledge Discovery and Data Mining,* (pp. 307-328). Menlo Park, CA: AAAI Press / The MIT Press.

Agrawal, R. and Srikant, R. (1994). *Fast Algorithms for Mining Association Rules.* In proceedings of the International Conference on Very Large Databases, (pp. 487-499). Santiago, Chile. September 1994.

Anderson, J. (1995). *An Introduction to Neural Networks.* MIT Press.

Ankerst, M., Keim, D. A. and Kriegel, H-P. (1996). Circle Segments: A Technique for Visually Exploring Large Multidimensional Data Sets. *IEEE Visualization 96 Conference,* (pp. 4-7). San Francisco, CA USA.

Ashby, W. R. (1952). *Design for a Brain.* New York: John Wiley.

Auer Jr., A. (1992). *Guidelines for Forecasting Fog. Part 1: Theoretical Aspects.* New Zealand, Meteorological Service of New Zealand, August 1992.

Australian Bureau of Meteorology (2003). [On line], last accessed on June 12, 2004. http://www.bom.gov.au.

Australian Bureau of Meteorology (2003). [On line]. *Preliminary report on meteorological aspects of the 1998 Sydney to Hobart yacht race.* Last accessed on June 20, 2003. http://www.bom.gov.au/inside/services_policy/marine/sydney_hobart/prelrept.html

Azvine, B., Azarmi, N. and Nauck, D. (2000). *Intelligent Systems and Soft Computing: Prospects, Tools and Applications.* (Ed, Carbonell, J. and Siekmann, J.), Lecture Notes in Artificial Intelligence. Adastral Park, UK: Springer-Verlag.

Barker, V., O'connor, D., Bachant, J. and Soloway, E. (1989). Expert systems for configuration at Digital: XCON and beyond. *Communications of the ACM,* 32 (3), 298-318.

Barto, A. (1985). Learning by statistical cooperation of self-interested neuron-like computing units. *Human Neurobiology,* 4, 229-256.

Barto, A., Sutton, R. and Anderson, C. (1983). Neuron-like adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man and Cybernetics*, SMC 13, 834-846.

Beckenkamp, F.G. (2002). *A Component Architecture for Artificial Neural Network Systems.* Faculty of Sciences, Department of Computer and Information Science, University of Constance, Germany. Ph.D. Thesis.

Beckenkamp, F.G. and Pree, W. (1999). Neural Network Framework Components. In (Ed, Fayad, M., Schmidt, D.C. and Johnson, R.), *Object-Oriented Application Framework: Applications and Experiences.* John Wiley.

Beckenkamp, F.G. and Pree, W. (2000). Neural Networks Components. *Neural Computation 2000 (NC'2000).* Berlin, Germany. May 2000. http://www.icsc-naiso.org/

Bedson, G.J. and Canterford, R.P. (1983). *A study of enhanced GMS imagery for operational forecasting.* Australia, Bureau of Meteorology. Technical Report 53.

Berry, M.J.A. and Linoff, G. (2000). *Mastering Data Mining, The Art and Science of Customer Relationship Management.* (Ed, Elliott, R.M.). USA: John Wiley & Sons, Inc.

Beshers, C. and Feiner, S. (1990). Visualizing n-Dimensional Virtual World with n-Vision. *Computer Graphics*, 24 (2), 37-38.

Bezdek, J.C. (1994). What is Computational Intelligence? In (Ed, Zurada, J.M., Marks II, R.J. & Robinson, C.J.), *Computational Intelligence Imitating Life*, (Chapter 1). New York: IEEE Press.

Bonczek, R.H., Holsapple, C.W. and Whinston, A B. (1981). *Foundations of Decision Support Systems.* New York: Academic Press.

Bond, G. (1981). *Forecasting fog in data sparse areas using IR satellite imagery: a case history.* Australia, Bureau of Meteorology. Meteorological Note 128.

Bonissone, P., Chen, Y-T., Goebel, K. and Khedkar, P.S. (1999). Hybrid Soft Computing Systems: Industrial and Commercial Applications. *Proceedings of the IEEE*, 87 (9), 1648-1667.

Breiman, L., Friedman, J.H., Olshen, R.A. and Stone, C.J. (1984). *Classification and Regression Trees.* Belmont, CA USA: Wadsworth Statistical Press.

Brighton, H. and Mellish, C. (2001). Identifying competence-critical instances for instance-based learners. In (Ed, Liu, H. and Motoda, H.), *Instance Selection and Construction for Data Mining*, (pp. 77-94). Massachusetts, USA: Kluwer Academic Publishers.

Buchanan, B. and Feigenbaum, E. (1978). DENDRAL and META-DENDRAL: Their application dimensions. *Artificial Intelligence*, 1 (11), 5-24.

Buchner, A.G., Chan, C.L., Hung, S.L. and Hughes, J.G. (1998). A meteorological knowledge-discovery environment. In (Ed, Bramer, M.), *Knowledge Discovery and Data Mining*, (pp. 204-226). Portsmouth, UK: Institution of Electrical Engineers.

Burstein, F., Smith, H., Sowunmi, A. and Sharma, R. (1998). Organisational Memory Information Systems: a Case-based Approach to Decision Support. In (Eds, Kersten, G., Mikolajuk, Z., Rais, M. and Yeh, A.), *Decision Analysis and Support for Sustainable Development*, (Chapter 20).

Carpenter, G.A. and Grossberg, S. (1987). A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision, Graphics, and Image Processing*, 37, 54-115.

Catlett, J. (1991). *Megainduction: Machine Learning on Very Large Databases.* School of Computer Science, University of Sydney, Australia. PhD Thesis.

Chauchat, J. and Rakotomalala, R. (2001). Sampling Strategy for Building Decision Trees from Very Large Databases Comprising Many Continuous Attributes. In (Ed, Liu, H.

and Motoda, H.), *Instance Selection and Construction for Data Mining*, (pp. 171-188). Massachusetts, USA: Kluwer Academic Publishers.

Cheeseman, P. and Stutz, J. (1996). Bayesian classification (auto-class): Theory and results. In (Ed, Fayyad, U., Piatetsky-Shapiro, G., Smyth, P. & Uthurusamy, R.), *Advances in Knowledge Discovery and Data Mining*, (pp. 153-180). Cambridge: AAAI Press / The MIT Press.

Chen, F., Figlewski, S., Weigend, A.S. and Waterhouse, S.R. (1998). Modeling financial data using clustering and tree-based approaches. *International Conference on Data Mining*, (pp. 35-51). Rio de Janeiro, Brazil. WIT Press.

Chernoff, H. (1973). The Use of Faces to Represent Points in k dimensional Space Graphically. *Journal of American Statistica Association*, 68 (342), 361-368.

Clancy, W.J. (1983). The Epistemology of a Rule-based Expert System: A Framework For Explanation. *Artificial Intelligence*, 20, 215-251.

Cleveland, W.S. and McGill, M.E. (Eds.) (1988). *Dynamic Graphics for Statistics*. Belmont, CA USA: Wadsworth and Brooks/Cole.

Cohen, P.R. (1996). *Empirical Methods for Artificial Intelligence*. Cambridge: MIT Press.

Colby, K.M. (1965). Computer Simulation of Neurotic Processes. In (Ed, Stacey, R.W. and Waxman, B.D.), *Computer simulation of Personality*. New York: Wiley.

Colby, K.M., Watt, J. and Gilbert, J.P. (1966). A computer model of psychotherapy. *Journal of Nervous and Mental Diseases*, 142, 148-152.

Colquhoun, J.R. (1987). A decision tree method of forecasting thunderstorms, severe thunderstorms and tornadoes. *Weather and Forecasting*, 2, 337-345.

Cox, E. (1994). *The Fuzzy Systems Hand Book*. Boston, USA: Academic Press Professional.

Crevier, D. (1993). *AI: The Tumultuous History of the Search for Artificial Intelligence.* New York, NY: BasicBooks.

Dasarathy, B.V. (1990). *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques.* Los Alamitos, CA USA: IEEE Compute Society Press.

Denis, F.A. and Machado, R.J. (1991). *O modelo conexcionista evolutivo.* Rio de Janeiro, Brazil, IBM Rio Scientific Center. Technical Report CCR-128.

Ernest, G.W. and Newell, A. (1969). *GPS: A Case Study in Generality and Problem Solving.* New York: Academic Press.

Fawectt, T. and Provost, F. (1996). Combining Data Mining and Machine Learning for Effective User Profile. *Second International Conference on Knowledge Discovery and Data Mining,* (pp. 8-13). Portland, Oregon USA. AAAI Press.

Fayyad, U.M., Haussler, D. and Stolorz, P. (1996). Mining Scientific Data. *Communications of the ACM,* 39 (11), 51-57.

Fayyad, U.M. and Irani, K.B. (1993). Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning. In *proceedings of the 13th International Joint Conference on Artificial Intelligence,* (pp. 1022-1027). Chambéry, France.

Fayyad, U.M., Mannila, H. and Ramakrishman, R. (1997). *Data Mining and Knowledge Discovery.* Boston: Kluwer Academic Publishers.

Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P. and Uthurusamy, R. (1996). From Data Mining to Knowledge Discovery: An Overview. In *Advances in Data Mining and Knowledge Discovery,* (pp. 1-34). Cambridge: AAAI / The MIT Press.

Feigenbaum, E.A. (1963). The Simulation of Verbal Learning Behavior. In (Ed, Feigenbaum, E.A. and Feldman, J.), *Computers and Thought.* New York: McGraw-Hill.

Foreman, E.K. (1991). *Survey sampling principles.* Statistics, textbooks, and monographs. New York, NY USA: Marcel Dekker, INC.

Frawley, W.J., Piatetsky-Shapiro, G. and Matheus, C.J. (1992). Knowledge Discovery in Database: An Overview. *AI Maganize*, 13 (3), 57-70.

Freeman, J.A. and Skapura, D.M. (1991). *Neural Networks: Algorithms, Applications, and Programming Techniques.* (Ed, Koch, C.), Computational and Neural Systems Series, Addison-Wesley Publishing Company.

Gamma, E., Helm, R., Johnson, R. and Vlissides, J. (1995). *Design Patterns-Elements of Reusable Object-Oriented Software.* Reading, Massachusetts: Addison-Wesley.

Goonatilake, S. and Khebbal, S. (1995). Intelligent Hybrid Systems: Issues, Classifications and Future Directions. In (Ed, Goonatilake, S. and Khebbal, S.), *Intelligent Hybrid Systems*, (pp. 3-20). New York NY: John Wiley & Sons.

Gottgtroy, B., Rodrigues, N. and Sousa, G. (1998). Data mining agents. *International Conference on Data Mining*, (pp. 171-182). Rio de Janeiro, Brazil. WIT Press.

Grossberg, S. (1982). Adaptive pattern classification and universal recoding: Parallel development and coding of neural feature detectors. In (Ed, Grossberg, S.), *Studies of Mind and Brain*, (pp. 448-497). Boston, USA: D. Reidel Publishing.

Gu, B., Hu, F. and Liu, H. (2000). *Sampling and Its Application in Data Mining*: A Survey. Singapure, Department of Computer Science, National University of Singapure, June 2000. Technical Report TRA6/00.
http://techrep.comp.nus.edu.sg/techreports/2000/TRA6-00

Gu, B., Hu, F. and Liu, H. (2001). Sampling: Knowing Whole from Its Part. In (Ed, Liu, H. and Motoda, H.), *Instance Selection and Construction for Data Mining*, (pp. 21-38). Massachusetts, USA: Kluwer Academic Publishers.

Han, C.Y. and Wee, W.G. (1992). A problem solving systems for data analysis, pattern classification and recognition. In (Ed, Kandel, A. and Langholz, G.), *Hybrid Architectures for Intelligent Systems*, (pp. 279-299). Boca Raton, FL: CRC Press.

Han, J. (1998). Data Mining: An Overview from Databases Perspective. *Tutorial on the Pacific-Asia Conference in Knowledge Discovery and Data Mining (PKDD-98)*, Melbourne, Australia. April 1998.

Hand, D., Mannila, H. and Smyth, P. (2001). *Principles of data mining.* (Ed, Dietterich, T.), Adaptive Computation and Machine Learning Series. Cambridge: The MIT Press.

Harris-Jones, C. and Haines, T.L. (1998). Sample size and misclassification: Is more always better?. In *Proceedings of the Second International Conference and Exhibition on The Practical Application of Knowledge Discovery and Data Mining (PADD'98).* London UK.

Haussler, D., Kearns, M., Seung, H.S. and Tishby, N. (1996). Rigorous learning curve bounds from statistical mechanics. *Machine Learning*, 25, 195-236.

Hayes-Roth, F. (1983). *Building Expert Systems.* Reading, MA: Addison-Wesley.

Hayes-Roth, F. and Jacobstein, N. (1994). The State of Knowledge-Based Systems. *Communications of the ACM*, 37 (3), 27-39.

Haykin, S. (1994). *Neural Networks A Comprehensive Foundation.* (Ed, Griffin, J.), New York, USA: Macmillian College Publishing Company.

Hebb, D.O. (1949). *The Organization of Behaviour.* New York: John Wiley.

Heckerman, D. (1996). Bayesian Networks for Knowledge Discovery. In (Ed, Fayyad, U., Piatetsky-Shapiro, G., Smyth, P. & Uthurusamy, R.), *Advances in Knowledge Discovery and Data Mining*, (pp. 273-305). Menlo Park, CA: AAAI Press / The MIT Press.

Holland, J. (1986). Escaping brittleness: The possibilities of general purpose learning algorithms applied to parallel rule-based systems. In (Ed, Michalski, R., Carbonell, J. and Mitchell, T.), *Machine Learning*, 2, 593-623. Morgan Kaufmann.

Holtzman, S. (1989). *Intelligent Decision Systems.* (Eds, Buchanan, B., Davis, R., Erman, L.D., King, D., McDermott, J. and Stefik, M.), The Teknowledge Series in Knowledge Engineering. Menlo Park California: Addison-Wesley.

Hopfield, J.J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceeding of the National Academy of Sciences of the USA*, USA. (pp. 2554-2558).

Hornik, K., Stinchcombe, M. and White, H. (1989). Multilayer Feedforward Networks are Universal Approximators. *Neural Networks*, 2 (5), 359-366.

Houtsma, M. and Swami, A. (1993). *Set-oriented mining of association rules.* San Jose, CA USA, IBM Almaden Research Center. Research Report RJ 9567.

Howard, C.M. and Rayward-Smith, V.J. (1998). Discovering Knowledge from low-quality meteorological databases. *IEE Colloquium on Knowledge Discovery and Data Mining (1998/434)*, (pp. 4/1-4/5). May 1998. London UK.

Hruschka, E.R. and Ebecken, N.F.F. (1998). Rule Extraction from Neural Networks in Data Mining Applications. *International Conference on Data Mining*, (pp. 303-314). Rio de Janeiro, Brazil. WIT Press.

Hull, R. and King, R. (1987). Semantic databases modelling: survey, applications, and research issues. *ACM Computing Surveys*, 19, 201-260.

Jagielski, R. and Jagielska, I. (1998). Using Genetic Programming for Data Mining in the Development of a Decision Support System. *The Australian Workshop on Intelligent Decision Support and Knowledge Management (AWIDS'98)*, (pp. 101-107). Sydney, Australia. Monash University Press.

Kandel, A. and Langholz, G. (1992). *Hybrid Architectures for Intelligent Systems.* Boca Raton, Florida: CRC Press.

Keith, R. (1991). *Results And Recommendations Arising From An Investigation Into Forecasting Problems At Melbourne Airport.* Townsville, Australia, Bureau of Meteorology, March 1991. Meteorological Note 195.

Kennedy, R.L., Lee, Y., Roy, B.V., Reed, C.D.and Lippmann, R. (1998). *Solving Data Mining Problems through Pattern Recognition.* The Data Warehousing Institute. New Jersey NJ: Prentice Hall PTR.

Kerber, R., Livezey, B. and Simoudis, E. (1995). A Hybrid System for Data Mining. In (Ed, Goonatilake, S. and Khebbal, S.), *Intelligent Hybrid Systems,* (pp. 121-142). London UK: John Wiley & Sons.

Klemettinen, M. (1999). *A Knowledge Discovery Methodology for Telecommunication Network Alarm Databases.* Department of Computer Science, University of Helsinki, Finland. PhD Thesis.

Klemettinen, M., Mannila, H. and Toivone, H. (1999). Rule discovery in telecommunication alarm data. *Journal of Network and Systems Management,* 7 (4), 395 - 423.

Klemettinen, M., Mannila, H. and Toivonen, H. (1997). A data-mining methodology and its application to semi-automatic knowledge acquisition. *Proceedings of the 8th International Conference and Workshop on Database and Expert Systems Applications (DEXA'97),* (pp. 670-677). Toulouse, France. September 1997.

Kohonen, T. (1972). Correlation matrix memories. *IEEE Transactions on Computers,* C-21, 353-359.

Kohonen, T. (1982). Self-organized formation of topologically correct features maps. *Biological Cybernetics,* 43, 59-69.

Kolonder, J. (1993). *Case-based Reasoning.* Mountain View, CA: Morgan Kaufmann.

Kononenko, I., Simec, E. and Robnik-Sikonja, M. (1997). Overcoming the myopia of inductive learning algorithms with RELIEFF. *Applied Intelligence*, 7, 39-55.

Kosko, B. (1988). Bidirectional associative memories. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-18, 42-60.

Kosko, B. (1992). *Neural Networks and Fuzzy Systems.* New Jersey, Englewood Cliffs: Prentice Hall.

Krovvidy, S. and Wee, W.G. (1992). An Intelligent Hybrid System for Wastewater Treatment. In (Ed, Kandel, A. and Langholz, G.), *Hybrid Architectures for Intelligent Systems* (pp. 358-377). Boca Raton FL: CRC Press.

Lawrence, D. (1991). *Handbook of Genetic Algorithms.* New York: Van Nostrand Reinhold.

Leao, B.F. (1988). *Construcao da base de conhecimento de um sistema especialista de apoio ao diagnostico de cardiopatias congenitas (The construction of an expert system knowledge base for the diagnostic support of congenital heart diseases).* University of São Paulo, Brazil. PhD. Thesis.

Leao, B.F. and Reategui, E.B. (1993a). A hybrid connectionist expert system to solve classificational problems. *Proceedings of Computers in Cardiology.* London: IEEE Computer Society.

Leao, B.F. and Reategui, E.B. (1993b). Hycones: a hybrid connectionist expert system. *Seventeen Annual Symposium on Computer Applications in Medical Care (SCAMC),* Washington DC. NY: IEEE Press.

Leao, B.F. and Rocha, A. (1990). Proposed methodology for knowledge acquisition: a study on congenital heart disease diagnosis. *Methods of Information in Medicine*, 29, 30-40.

Lenat, D.B. (1981). AM: An Artificial Intelligence Approach to Discovery in Mathematics as Heursitic Search. In (Ed, Davis, R. and Lenat, D.), *Knowledge-Based Systems in Artificial Intelligence.* New York: McGraw-Hill.

Lenat, D.B. and Guha, R.V. (1988). *The World Accordingly to CYC.* MCC, September 1988. Technical Report ACA-AI-300-88.

Lenat, D.B., Prakash, M. and Shepherd, M. (1986). Cyc: Using Common Sense Knowledge to Overcome Brittleness and Knowledge-Acquisition Bottlenecks. *AI Magazine,* Winter 1986, 6, 65-85.

Lewis, D. and Catlett, J. (1997). Heterogeneous uncertainty sampling for supervised learning. *In proceedings of the 11th International Conference on Machine Learning,* (pp. 148-156). Alicante, Spain. September 1997.

Ling, C.X. and Li, C. (1998). Data Mining for Direct Marketing: Problems and Solutions. *The Fourth International Conference on Knowledge Discovery and Data Mining,* (pp. 73-79). New York, NY: AAAI Press.

Linger, H. and Burstein, F. (1997). Intelligent DSS in the Context of Modern Organisation. *The Fourth Conference of the International Society for Decision Support Systems (ISDSS-97),* (pp. 429-443). Lausanne, Switzerland. July 1997.

Linger, H., Burstein, F., Kelly, J., Ryan, C. and Gigliotti, P. (2000). Creating a Learning Community Through Knowledge Management: The Mandala Project. *IFIP WG 8.3 Conference Decision support through knowledge management.* Sweden. July 2000.

Liu, B., Hsu, W. and Ma, Y. (1998). Integrating Classification and Association Rule Mining. *The Fourth International Conference on Knowledge Discovery and Data Mining,* (pp. 80-86). New York NY: AAAI Press.

Liu, H. and Motoda, H. (2001). Data reduction via instance selection. In (Ed, Liu, H and Motoda, H.), *Instance Selection and Construction for Data Mining* (pp. 3-20). Massachusetts USA: Kluwer Academic Publishers.

Love, G. (1985). *Tropical weather forecasting: A practical viewpoint.* Darwin, Australia, Bureau of Meteorology. Research Paper (unpublished).

Ludermir, T.B., Braga, A.P., Nobre, C.N. and Carvalho, P.L. (1998). Extracting Rules from Neural Networks: a Data Mining Approach. *International Conference on Data Mining,* (pp. 303-314). Rio de Janeiro, Brazil. WIT Press.

Machado, R.J., Barbosa, C. and Neves, A. (1998). Learning in the Combinatorial Neural Model. *IEEE Transactions on Neural Networks,* 9 (5), 831-847

Machado, R.J. and Rocha, A.F. (1989). *Handling Knowledge in High Order Neural Networks: the Combinatorial Neural Model.* Rio de Janeiro, Brazil, IBM Rio Scientific Center. Technical Report CCR076.

Machado, R.J. and Rocha, A.F. (1990). The combinatorial neural network: a connectionist model for knowledge based systems. In (Ed, Bouchon-Meunier, B., Yager, R.R. & Zadeh, L.A.), *Uncertainty in knowledge bases.* Berlin, Germany: Springer Verlag.

Machado, R.J. and Rocha, A.F. (1992). A hybrid architecture for fuzzy connectionist expert systems. In (Ed, Kandel, A. & Langholz, G.), *Hybrid architectures for intelligent systems* (pp. 135-152). Boca Raton FL: CRC Press.

Machado, R.J., Rocha, A.F. and Denis, F. (1992). Evolutive Learning in Neural Networks. *IEEE International Conference on Fuzzy Systems,* (pp. 762-767). San Diego CA.

Mannila, H., Toivonen, H. and Verkamo, I. (1995). Discovering frequent episodes in sequences. *First International Conference on Knowledge Discovery and Data Mining (KDD'95),* (pp. 210-215). Montreal, Canada. August 1995. AAAI Press.

Mayer, R.E. (1992). *Thinking, Problem Solving, Cognition.* (Ed, Atkinson, R.C., Lindzey, G., Thompson R.F.), A Series of Books in Psychology, Second Ed., New York: W. H. Freeman and Company.

McClelland, L., Rumelhart, E. and Hinton, E. (1988). The appeal of parallel distributed processing. *Parallel Distributed Processing*, 1, 4-44.

McCulloch, W.S. and Pitts, W.H. (1943). A Logical Calculus of the Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics*, 5, 115-33.

McFadden, F.R., Hoffer, J.A. and Prescott, M.B. (1999). *Modern Database Management.* 5th Ed. Addison-Wesley Longman, Inc.

Medsker, L. and Liebowitz, J. (1994). *Design and Development of Expert Systems and Neural Networks.* Macmillan College Publishing Company Inc.

Medsker, L.R. (1991). (Guest Editor) Special Issue on The synergism of expert system and neural network technologies. *Expert Systems With Applications: An International Journal*, 2, 1-2.

Medsker, L.R. (1995). *Hybrid Intelligent Systems.* Kluwer Academic Publishers.

Medsker, L.R. and Bailey, D.L. (1992). Models and Guidelines for Integrating Expert Systems and Neural Networks. In (Ed, Kandel, A. and Langholz, G.), *Hybrid Architectures for Intelligent Systems* (pp. 153-171). Boca Raton FL: CRC Press.

Meneses, C.J. and Grinstein, G. (1998). Categorization and Evaluation of Data Mining Techniques. *International Conference on Data Mining*, (pp. 53-80). Rio de Janeiro, Brazil. September 1998. WIT Press.

Michie, D., Spiegelhalter, D.J. and Taylor, C.C. (Eds.) (1994). *Machine Learning, Neural and Statistical Classification.* Midsomer Norton, UK: Ellis Horwood Limited.

Miller, R.A. (1986). Internist-I: an experimental computer-based diagnostic consultant for general internal medicine. In (Eds, Reggia, J.A. and Stanley, T.), *Computer-assisted medical decision making*, 2, 139-158. New York: Springer Verlag.

Minsky, M. and Papert, S. (1969). *Perceptrons*. Cambridge, MA.: MIT Press.

Mohammed, J.Z., Parthasarathy, S., Li, W. and Ogihara, M. (1996). *Evaluation of Sampling for Data Mining of Association Rules*. NY, The University of Rochester, Computer Science Dept., May 1996. Technical Report 617.

Molina, R., Blanca, N.P. and Taylor, C.C. (1994). Modern Statistical Techniques. In (Eds, Michie, D., Spiegelhalter, D.J. and Taylor, C.C.), *Machine Learning, Neural and Statistical Classification*, (pp. 29-49). Ellis Horwood Limited.

Moustakis, V.S., Letho, M. and Salvendy, G. (1996). Survey of expert opinion: which machine learning method may be used for which task? *International Journal of HCI, Special issue on machine learning*, 8 (3), 221-236.

Newell, A., Shaw, J.C. and Simon, H.A. (1958). Chess-playing programs and the problem of complexity. *IBM Journal of Research and Development*, 2, 320-335.

Newell, A. and Simon, H.A. (1963). GPS: A Program That Simulates Human Thought. In (Ed, Feigenbaum, E.A. and Feldman, J.), *Computers and Thought*, (pp. 279-293). New York: McGraw-Hill.

Newell, A. and Simon, H.A. (1972). *Human problem solving*. Englewood Cliffs, NJ.: Prentice-Hall.

Nunamaker, J., Chen, M. and Purdin, T. (1991). Systems Development in Information Systems Research. *Journal of Management Information Systems*, 7 (3), 89-106.

Owens, D.K. and Sox, H.C. (1990). Medical decision making: probabilistic medical decision making. In (Ed, Shortlife, E.H., Perreault, L.E., Wiedehold, G. and Fagan,

L.M.), *Medical Informatics: computer applications in health care*, (pp. 70-116). MA: Addison-Wesley.

Park, J.S., Chen, M. and Yu, P.S. (1995). An Effective Hash Based Algorithm for Mining Association Rules. ACM SIGMOD *Conference on Management of Data (SIGMOD'95)*, (pp. 175-186). San Jose, California USA. May 1995. ACM Press.

Piatetsky-Shapiro, G. and Frawley, W. (1991). *Knowledge Discovery in Databases*. MIT Press.

Piramuthu, S., Shaw, M.J. and Gentry, J.A. (1990). *Classification Using Multi-Layered Perceptrons*. Robotics Institute, Carnegie Mellon University, December 1990. Technical Report CMU-RI-TR-90-29.

Pree, W. (1995). *Design Patterns for Object-Oriented Software Development*. Reading, MA: Addison-Wesley/ACM Press.

Pree, W., Beckenkamp, F.G. and Rosa, S.I.V. (1997). Object-Oriented Design & Implementation of a Flexible Software Architecture for Decision Support Systems. *Proceedings of the 9th. International Conference on Software Engineering & Knowledge Engineering (SEKE'97)*, (pp. 382-388). Madrid, Spain. June 1997.

Provost, F. and Buchanan, B. (1995). Inductive policy: The pragmatics of bias selection. *Machine Learning*, 20, 35-61.

Provost, F., Jensen, D. and Oates, T. (2001). Progressive Sampling. In (Ed, Liu, H. and Motoda, H.), *Instance Selection and Construction for Data Mining*, (pp.151-170). Massachusetts USA: Kluwer Academic Publishers.

Provost, F. and Kolluri, V. (1999). A Survey of Methods for Scaling Up Inductive Algorithms. *Data Mining and Knowledge Discovery*, 3 (2), 131-169.

Pyle, D. (1999). *Data Preparation for Data Mining*. San Francisco, California USA: Morgan Kaufmann Publishers.

Quinlan, J.R. (1979). Discovering rules from large collection of examples: a case study. In (Ed, Michie, D.), *Expert Systems in the micro eletronic age.* Edinburgh, UK: University Press.

Quinlan, J.R. (1987). Simplifying decision trees. *International Journal of Man-Machine Studies*, 27 (3), 221-234.

Quinlan, J.R. (1993). C4.5: *Programs for Machine Learning.* San Mateo, CA.: Morgan Kaufmann.

Reategui, E.B. (1997). *Combining Case-Based Reasoning with Neural Networks in Diagnostic Systems.* Computer Science Dept., University of London, UK. PhD Thesis.

Reategui, E.B. and Campbell, J. (1995). A classification system for credit card transactions. *Advances in Case-Based Reasoning: Second European Workshop (EWCBR-94),* (pp. 280-291). Chantilly, France. Berlin, Germany: Springer Verlag.

Reategui, E.B., Campbell, J. and Borghetti, S. (1995). Using a neural network to learn general knowledge in a case-based system. *First International Conference on Case-Based Reasoning: Research and Development (ICCBR-95),* (pp. 528-537). Sesimbra, Portugal. Berlin, Germany: Springer Verlag.

Reategui, E.B., Campbell, J. and Leao, B.F. (1997). Combining a Neural Network with Case-Based Reasoning in a Diagnostic System. *Artificial Intelligence in Medicine*, 9 (1), 5-27.

Reinartz, T. (2001). A Unifying View on Instance Selection. In (Ed, Liu, H. and Motoda, H.), *Instance Selection and Construction for Data Mining,* (pp. 39-56). Massachusetts, USA: Kluwer Academic Publishers.

Reitman, W.R. (1965). *Cognition and thought: An information processing approach.* New York: Wiley.

Rekimoto, J. and Green, M. (1993). The Information Cube: Using Transparency in 3D Information Visualization. *Proceedings of the Third Annual Workshop on Information Technologies & Systems (WITS'93)*, (pp. 125-132). Orlando, FL USA.

Riesbeck, C.K. and Schank, R.C. (1989). *Inside Case-Based Reasoning.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Rissland, E.L. and Skalak, D.B. (1991). CABARET: rule integration in a hybrid architecture. *International Journal of Man-Machine Studies*, 34, 839-887.

Rojas, R. (1996). *Neural Networks: A Systematic Introduction.* Springer-Verlag.

Rosenblatt, F. (1958). The Perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65, 386-408.

Rosenblatt, F. (1962). *Principles of Neurodynamics.* Washington DC: Spartan Books.

Rumelhart, D.E., Hinton, G.E. and Williams, R.J. (1986). Learning representations by back-propagating erros. *Nature (London)*, 323, 533-536.

Rumelhart, D.E. and McClelland, J.L. (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition.* Cambridge MA: MIT Press.

Schank, R.C. (1982). *Dynamic Memory, a Theory of Understanding and Learning in Computers and People.* Cambridge UK: University Press.

Schneiderman, B. (1992). Tree Visualization with Treemaps: A 2D Space-Filling Approach. *ACM Transactions on Graphics*, 11 (1), 92-99.

Shim, J.P., Warkentin, M., Courtney, J.F., Power, D.J., Sharda, R. and Carlsson, C. (2002). Past, present, and future of decision support technology. *Decision Support Systems*, 33 (2), 111-126.

Siegler, W. and Steurer, E. (1998). Forecasting of the German stock index DAX with neural networks: Using daily data for experiments with input variable reduction and a modified error function. *International Conference on Data Mining*, (pp. 265-273). Rio de Janeiro, Brazil. WIT Press.

Simon, H. (1977). *The New Science of Management Decisions*. New Jersey, NJ: Prentice Hall.

Simpson, P.K. (1992). Foundations of Neural Networks. In (Ed, Sanchez-Sinencio, E. and Lau, C.G.Y.), *Artificial Neural Networks: Paradigms, Applications, and Hardware Implementations*, (pp. 3-24). New York: IEEE Press.

Smyth, P. and Goodman, R. (1992). An information theoretic approach to rule induction from databases. *IEEE Transactions on Knowledge and Data Engineering*, 4 (4), 301 -316.

Sprague, R. H. (1993). A Framework for the Development of Decision Support Systems. In (Eds, Sprague, R.H. and Watson, H.J.), *Decision Support Systems Putting Theory into Practice*, (pp. 3-26). New Jersey, NJ: Prentice-Hall International.

Sun, R. (1995a). A Two-level architecture for structuring knowledge for commonsense reasoning. In (Ed, Sun, R. & Bookman, L.A.), *Computational Architectures Integrating Neural and Symbolic Processes: a Perspective on the State of the Art*, (pp. 247-281). Dordrecht, Netherlands: Kluwer Academic Publishers.

Sun, R. (1995b). An introduction on symbolic processing in neural networks. In (Ed, Sun, R. & Bookman, L.A.), *Computational Architectures Integrating Neural and Symbolic Processes: a Perspective on the State of the Art*, (pp. 1-20). Dordrecht, Netherlands: Kluwer Academic Publishers.

Sun, R. (2001). Artificial Intelligence: Connectionist and Symbolic Approaches. In (Ed, Smelser, N.J. and Baltes, P.B.), *International Encyclopedia of the Social and Behavioral Sciences*, (pp. 783-789). Oxford England: Pergamon/Elsevier.

Sun Microsystems (2004). [On line]. *Java Technology, Products & Technologies.* Last accessed on June 12, 2004. http://java.sun.com.

Surma, J. and Vanhoof, K. (1995). Integrating Rules and Cases for the Classification Task. *Case-Based Reasoning: Research and Development. First International Conference (ICCBR-95),* (pp. 325-334). Sesimbra, Portugal. Berlin: Springer Verlag.

Tecuci, G. and Kodratoff, Y. (1995). *Machine Learning and Knowledge Acquisition: Integrated Approaches.* London UK: Academic Press.

Teng, J.T.C., Mirani, R. and Sinha, A. (1988). A Unified Architecture for Intelligent DSS. In *proceedings of the 21st Annual Hawaii International Conference on System Sciences (HICSS-21),* (pp. 286-294). Hawaii USA: IEEE Computer Society Press.

Turban, E. and Aronson, J. (1998). *Decision Support Systems and Intelligence Systems.* 5th Ed., New Jersey: Prentice-Hall.

Turetken, O. and Sharda, R. (2004). Development of a fisheye-based information search processing aid (FISPA) for managing information overload in the web environment. *Decision Support Systems,* 37 (3), 415-434.

Viademonte, S. (1994). *SECOX-HI: Aplicacao de um Modelo Hibrido para Sistemas Especialistas em um Processo Decisorio Complexo (SECOX-HI: Application of a Hybrid Model for Expert System in a Complex Decision Process).* Programa de Pos-Graduacao em Administracao, Federal University of Rio Grande do Sul. Porto Alegre, Brazil. Msc. Thesis.

Viademonte, S. and Burstein, F. (2001a). An Approach to Knowledge Management in the Context of Intelligent Decision Support. In (Ed, Joyanes, L., Fyfe, C., Alonso, L. and Cochardo, J.M.), *Knowledge Management: The Future Has Arrived,* (pp. 29-38). Paisley, UK: University of Paisley Press.

Viademonte, S. and Burstein, F. (2001b). An Intelligent Decision Support Model for Aviation Weather Forecasting. *Advances in intelligent data analysis: 4th international conference (IDA 200)*, (pp. 278-288). Cascais, Portugal. Berlin, Germany: Springer-Verlag.

Viademonte, S., Burstein, F. and Beckenkamp, F.G. (2000). An Empirical Study of Distribution based on Voyager: A Performance Analysis. In *proceedings of the Thirty-Third Annual Hawaii International Conference on System Sciences (HICSS-33)*, (CD-ROM). Hawaii USA: IEEE Computer Society Press.

Viademonte, S., Burstein, F., Dahni, R. and Williams, S. (2001a). Discovering Knowledge from Meteorological Databases: A Meteorological Aviation Forecast Study. *Third International Conference on Data Warehousing and Knowledge Discovery (DaWaK 2001)*, (pp. 61-70). Munich, Germany. Berlin, Germany: Springer-Verlag.

Viademonte, S., Burstein, F., Dahni, R. and Willians, S. (2001b). Knowledge Discovery in Databases in an Intelligent Decision Support Context: A meteorological forecasting case study. *Fourth International ICSC Symposia on Soft Computing and Intelligent Systems for Industry (ISF 2001)*, (CD-ROM). Paisley, UK. June 2001. ICSC-NAISO Academic Press.

Viademonte, S., Hoppen, N. and Beckenkamp, F.G. (1997). The Application of Fuzzy Logic to Model Semantic Variables in a Hybrid Model for Classification Expert Systems. *Second International ICSC Symposium on Fuzzy Logic and Applications (ISFL'97)*, (pp. 301-307). Zurich, Switzerland. February 1997.

Viademonte, S., Leao, B.F. and Hoppen, N. (1995). Hybrid Model for Classification Expert System. *XXI Latin America Conference on Computer Science*, (pp. 639-648). Canela, Brazil.

Wang, J. (1994). Artificial neural networks versus natural neural networks: A connectionist paradigm for preference assessment. *Decision Support Systems*, 11 (5), 415-429.

Weiss, S.M., Galen, R.S. and Tadepalli, P.V. (1990). Maximizing the predictive value of production rules. *Artificial Intelligence*, 45, 47-71.

Weiss, S.M. and Indurkhya, N. (1998). *Predictive Data Mining: A Practical Guide.* (Ed, Morgan, M.B.), San Francisco, CA USA: Morgan Kaufmann Publishers, Inc.

Weizenbaum, J. (1968). Contextual understanding by computers. In (Ed, Eden, P.A. & Kolers, M.), *Recognizing patterns.* Cambridge: M.I.T. Press.

Werbos, P.J. (1974). *Beyond regression: New tools for prediction and analysis in the behavioural sciences.* Harvard University, Cambridge, MA. Ph.D. Thesis.

Widrow, B. and Hoff, M.E. (1960). Adaptive switching circuits. *IRE Western Electric Show and Convention Record*, Part 4, (pp. 96-104). USA.

Wu, X. (1995). *Knowledge Acquisition From Databases.* Norwood, NJ: Ablex Publishing.

Ye, L.R. and Johnson, P.E. (1995). The Impact of Explanation Facilities on User Acceptance of Expert Systems Advice. *MIS Quarterly*, 19 (2), 157-172.

Zadeh, L.A. (1973). Outline of a new approach to the analysis of complex system and decision processes. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-3, 28-44.

Zadeh, L.A. (1994). Fuzzy Logic, Neural Networks, and Soft Computing. *Communications of the ACM*, 37 (3), 77-84.

Zadeh, L.A. (2000). From Computing with Numbers to Computing with Words-From Manipulation of Measurements to Manipulation of Perceptions. In (Ed, Azvine, B., Azarmi, N. and Nauck, D.), *Intelligent Systems and Soft Computing: Prospects, Tools and Applications*, (pp. 3-40). Berlin, Germany: Springer-Verlag.

Zanella, M. and Gianfranco, L. (1999). Justification Dimensions for Complex Computer Systems. In *proceedings of the 5th International Conference on Information Systems, Analysis and Synthesis*, (pp. 317-324). Orlando, FL USA.

Zanella, M., Piero, M. and Guida, G. (1997). User Interaction in Decision Support Systems: The Role of Justification. *International Conference on Systems, Man, and Cybernetics*, (pp. 3215-3220). Orlando, FL USA: IEEE Inc.

Zurada, J.M. (1992). *Introduction to Artificial Neural Systems.* St Paul: West Publishing Company.

Appendices

Appendix A

The Knowledge Acquisition Methodology for the Elicitation of Knowledge Graphs[1]

Phase 0 - Problem Definition

1) Define the problem domain

2) Define the hypotheses (classes) that will be part of the domain

3) Define a list of evidences and attributes, with the assistance of an expert or a group of experts

Phase 1 - Knowledge Acquisition

1) Select one of the hypothesis (classes)

2) Ask the expert to indicate, in the list of evidences, the items necessary for the formulation of the hypothesis, by which a working subset of evidences is defined

3) Ask the expert to rank the working subset according to the importance of the items for the hypothesis

4) Ask the expert to assume the items of the ordered list as the evidence nodes of a knowledge graph and to associate them in the form that he judges necessary to establish an adequate foundation for the hypothesis (generating intermediate nodes that converge to the hypothesis)

---

[1]Adapted from (Reategui, 1997; Leao and Rocha, 1990).

5) Ask the expert to assign a membership degree between 0 and 10 for the information represented in each node of the graph in relation to the hypothesis

6) Ask the expert to define the logical operators (AND, OR, NOT) associated with the nodes of the graph

7) Repeat the process for the other hypotheses, to form a family of graphs.

Appendix B

Discretization of Numerical Attributes for Model1-80, Model2-60 and Model10-60

Attribute	Mining Model 1-80	Mining Model 2-60	Mining Model 10-60	Labels
Dry Bulb (Celsius degrees)	< = 8.5 > 8.5 & <= 12 > 12	< = 8.5 > 8.5 & <= 12 > 12	< = 8.5 > 8.5 & <= 12 > 12	Low Med High
Dew Point and Previous Dew Point (Celsius degrees)	< = 4 > 4 & < = 6 > 6 & < = 9 > 9	< = 4 > 4 & < = 6 > 6 & < = 9 > 9	< = 4 > 4 & < = 6 > 6 & < = 9 > 9	Low Med High Max
Total Cloud Amount (Eighths)	< =4 > 4 & < = 7 > 7	< = 4 > 7 & < = 7 > 7	< = 4 > 4 & < = 7 > 7	Min Med Max
Total Low Cloud Amount (Eighths)	< = 1 > 1 & < = 6 > 6	< = 1 > 1 & < = 6 > 6	< = 1 > 1 & < = 6 > 6	Min Med Max
Sea Level Pressure (HpA)	< = 1013.9 >1013.9 & <= 1019.8 >1019.8 & <= 1025.2 > 1025.2	< = 1013.6 >1013.6 & <= 1019.7 >1019.7 & <= 1024.9 >1024.9	< = 1013.6 >1013.6 & <= 1019.2 >1019.2 & <= 1024.9 > 1024.9	Low Med High VHigh
Wind Speed (meters/second)	< = 1.5 > 1.5 & < = 4.1 > 4.1 & < = 6.2 > 6.2	< = 2.1 > 2.1 & < = 4.1 > 4.1 & < = 6.7 > 6.7	< = 2.6 > 2.6 & < = 4.6 > 4.6 & < = 7.2 > 7.2	Light LMode Mode FMode
Visibility (Kilometers)	< = 25 > 25 & < = 35 > 35 & < = 40 > 40	< = 30 > 30 & < = 35 > 35 & < = 40 > 40	< = 30 > 30 & < = 35 > 35 & <= 40 > 40	Lev1 (level 1) Lev2 (level 2) Lev3 (level 3) Lev4 (level 4)

There are certain conventions in the labels, for instance: *Med* indicates *medium*, *Min* indicates *minimum*, *Max* indicates *maximum*, *VHigh* indicates *very high*, *LMode* indicates *light to moderate*, *Mode* indicates *moderate* and *FMode* indicates *fresh to moderate*.

Appendix C

Rule Sets by Mining Models Selecting Visibility Observation

Data Mining Model	Generated Rules by Rule Confidence Degree - Model V2 - Rule Support 8%, Maximum Rule Order 7, Selecting Visibility											
	50%			70%			80%			90%		
	F	NF	Total	F	NF	Total	F	NF	Total	F	NF	Total
Mining Model1-60				148	180	328	92	177	269	23	138	161
Mining Model1-80				209	234	443	120	234	354	22	182	204
Mining Model2-60				86	215	301	36	215	251	13	207	220
Mining Model10-60	28	228	256	11	228	239	7	228	235	7	228	235

Data Mining Model	Generated Rules by Rule Confidence Degree – Model V3 Rule Support 6%, Maximum Rule Order 10, Selecting Visibility											
	50%			70%			80%			90%		
	F	NF	Total	F	NF	Total	F	NF	Total	F	NF	Total
Mining Model1-60				243	310	553	149	305	454	83	295	378
Mining Model1-80				332	377	709	188	377	565	38	314	352
Mining Model2-60				144	315	495	51	351	402	16	254	358
Mining Model10-60	32	375	407	14	375	389	9	375	384	9	375	384

Appendix D

Domain Modelling in the CANN Simulator for Model1-60.

The domain modelling for Model1-60 is presented here. The other models (Molde1-8, Model2-6 and Model10-60) follow the same structure. Different models might have different evidences, evidence values and morbidity values associated to them. Chapter 6 discussed issues about domain modelling into CANN.

In the table's header, the first column indicates the evidence (including the value ranges in the case of numerical evidences). The "Fethcer" column indicates the position in the ASCII files (both for training and testing) where the evidence is stored.

The "Values" column indicates the possible attributes that can be assigned to the evidences. The "Frequency" column indicates the frequency measured of each evidence/values pair in the dataset. The "Percent" column indicates the percent of the frequency in the dataset. The "Morbidity" specifies the morbidity value of the evidences, as a result of the morbidity calculation presented in Equation 6.1.

Hour	Fetcher	Values	Frequency	Percent	Morbidity
	[1,2]	0	425	11.67	0.20
		3	490	13.45	0.23
		6	528	14.49	0.29
		9	585	16.06	0.32
		12	425	11.67	0.20
		15	363	9.96	0.17
		18	392	10.76	0.18
		21	387	10.62	0.18

Month	Fetcher	Values	Frequency	Percent	Morbidity
	[3 , 4]	04	502	13.78	0.23
		05	569	15.62	0.27
		06	613	16.83	0.29
		07	555	15.23	0.26
		08	496	13.62	0.23
		09	471	12.93	0.22
		10	437	12.00	0.20

Drybulb	Fetcher	Values	Frequency	Percent	Morbidity
<= 8.5	[5 , 8]	LOW	1217	33.41	0.57
]8.5 ; 12]		MED	1225	33.63	0.57
> 12		HIGH	1201	32.97	0.56

Dew point	Fetcher	Values	Frequency	Percent	Morbidity
<= 4	[9 , 12]	LOW	972	26.68	0.45
]4 ; 6]		MED	874	23.99	0.41
]6 ; 9]		HIGH	1186	32.56	0.55
> 9		MAX	611	16.77	0.42

Previous Dew p.	Fetcher	Values	Frequency	Percent	Morbidity
< =4	[13 , 16]	LOW	950	26.08	0.44
]4 ; 6]		MED	912	25.03	0.43
]6 ; 9]		HIGH	1221	33.52	0.57
> 9		MAX	558	15.32	0.38

Cloud	Fetcher	Values	Frequency	Percent	Morbidity
<= 4	[17 , 19]	MIN	1327	36.43	0.62
]4 ; 7]		MED	1761	48.34	0.48
> 7		MAX	545	14.96	0.75

Low cloud	Fetcher	Values	Frequency	Percent	Morbidity
<= 1	[20 , 22]	MIN	1431	39.28	0.67
]1 ; 6]		MED	1284	35.25	0.60
> 6		MAX	908	24.92	1.00

Sea level pressure	Fetcher	Values	Frequency	Percent	Morbidity
<= 1014.1	[23 , 27]	LOW	914	25.09	0.43
]1014.1 ; 1020.2]		MED	899	24.68	0.42
]1020.2 ; 1025.6]		HIGH	903	24.79	0.42
> 1025.6		VHIGH	910	24.98	0.45

Wind speed	Fetcher	Values	Frequency	Percent	Morbidity
<= 1.5	[28 ; 32]	LIGHT	1037	28.47	0.48
]1.5 ; 3.6]		LMODE	852	23.39	0.40
]3.6 ; 6.2]		MODE	940	25.80	0.44
> 6.2		FMODE	812	22.29	0.38

Rainfall	Fetcher	Values	Frequency	Percent	Morbidity
no rain	[36]	0	3127	85.84	0.43
[0.1 ; 2.4]		1	435	11.94	0.20
]2.4 ; 4]		2	43	1.18	0.11

Visibility	Fetcher	Values	Frequency	Percent	Morbidity
<= 25	[41 ; 44]	LEV1	1098	30.14	0.75
]25 , 35]		LEV2	1032	28.33	0.48
]35 , 40]		LEV3	1338	36.73	0.62
> 40		LEV4	175	4.80	0.08

Windcompass	Fetcher	Values	Frequency	Percent	Morbidity
Calm	[33 ; 35]	CLM	584	16.03	0.27
		N	1162	31.90	0.54
		NNE	159	4.36	0.07
		NNW	218	5.98	0.10
		NW	149	4.09	0.07
		S	222	6.09	0.10
		SW	188	5.16	0.09
		W	333	9.14	0.16
		WSW	183	5.02	0.09
		SSW	109	3.00	0.05
		WNW	113	3.10	0.05
Variable		VAR			0.05
Unknown		UNK			0.05

Present weather	Fetcher	Values	Frequency	Percent	Morbidity
	[37 ; 38]	1	314	8.62	0.15
		2	1259	34.56	0.59
		3	361	9.91	0.17
		4	289	7.93	0.13
		5	119	3.27	0.06
		15	228	6.26	0.11
		40	213	5.85	0.10

Past weather	Fetcher	Values	Frequency	Percent	Morbidity
	[39 ; 40]	1	650	17.84	0.30
		2	1163	31.92	0.54
		3	243	6.67	0.11
		4	159	4.36	0.07
		15	111	3.05	0.05
		40	185	5.08	0.09
		44	162	4.45	0.08
		80	230	6.31	0.11

The weighting variable β is usually equal to 1.7. There were cases in which a different value was arbitrarily assigned to β. This is because domain knowledge indicated that a certain evidence value might have a higher or lower value than 1.7. For example, Visibility level 1, β = 2.5. This is because lower levels of visibility are strongly associated to fog occurrence. Rainfall

level 0, $\beta = 0.5$. This is because rainfall level 0 is associated with fog and other weather events as well. And a high number of rainfall level 0 was verified in the weather observation database.

The evidences with different values than 1.7 assigned to β are: Rainfall level 2 (β=9.5), wind compass unknown and variable (β=0.05), sea level pressure VHIGH (β=1.8), low cloud amount MAX (β=4), cloud amount MAX (β=5), cloud amount MED (β=1), previous dewpoint MAX (β=2.5), dewpoint MAX (β=2.5) and hour 6 and 9 (β=2).