BINDING SERVICES
Tel +44 (0)29 2087 4949
Fax +44 (0)29 20371921
e-mail bindery@cardiff.ac.uk

Flexible Information Management Strategies in Machine Learning and Data Mining

A thesis submitted to the University of Wales, Cardiff

For the degree of

Doctor of Philosophy

By

Duc-Cuong Nguyen

Manufacturing Engineering Centre

School of Engineering

University of Wales, Cardiff

United Kingdom

2004

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

Abstract

In recent times, a number of data mining and machine learning techniques have been applied successfully to discover useful knowledge from data. Of the available techniques, rule induction and data clustering are two of the most useful and popular. Knowledge discovered from rule induction techniques in the form of If-Then rules is easy for users to understand and verify, and can be employed as classification or prediction models. Data clustering techniques are used to explore irregularities in the data distribution. Although rule induction and data clustering techniques are applied successfully in several applications, assumptions and constraints in their approaches have limited their capabilities. The main aim of this work is to develop flexible management strategies for these techniques to improve their performance.

The first part of the thesis introduces a new covering algorithm, called Rule Extraction System with Adaptivity, which forms the whole rule set simultaneously instead of a single rule at a time. The rule set in the proposed algorithm is managed flexibly during the learning phase. Rules can be added to or omitted from the rule set depending on knowledge at the time. In addition, facilities to process continuous attributes directly and to prune the rule set automatically are implemented in the Rule Extraction System with Adaptivity algorithm.

The second part introduces improvements to the K-means algorithm in data clustering. Flexible management of clusters is applied during the learning process to help the algorithm to find the optimal solution. Another flexible management strategy is used to facilitate the processing of very large data sets. Finally, an effective method to determine the most suitable number of clusters for the K-means algorithm is proposed. The method has overcome all deficiencies of K-means.

Acknowledgements

I would like to express my sincere gratitude to Professor D. T. Pham, my supervisor, for creating the opportunity of my studying in the UK. I am grateful for his invaluable guidance and for his consistent encouragement during the past three years.

The System Division of the School of Engineering, University of Wales, Cardiff is a good place to study and work. I thank all its members for their friendship and help, in particular, Dr. Stefan Dimov, for his technical advises.

I would especially like to thank my family for their mental support. Thanks also go to my wife, Giao Quynh Nguyen, for her tolerance and belief over these years, and my son, Vinh Duc Nguyen, for his love.

This work is supported by the CVCP and the Manufacturing Engineering Centre, School of Engineering, University of Wales, Cardiff.

Contents

Abstract i

Acknowledgements ii

Declaration iii

Contents iv

List of Figures ix

List of Tables xii

Abbreviations xiii

List of Symbols xiv

Chapter 1 Introduction 1

1.1. Background 1

1.2. Research Objectives 3

1.3. Thesis Structure 3

Chapter 2 Literature Review 5

2.1. Machine Learning & Data Mining 5

2.1.1. Machine Learning 5

2.1.2. Data Mining 9

2.2. Inductive Learning 18

2.2.1. Decision Tree 19

2.2.2. Covering Methods 20

2.2.2.1. Separate-Conquer-and-Reduce Algorithms 22

2.2.2.2. Separate-Conquer-Without-Reduction Algorithms 25

2.2.2.3. Conquer-Without-Separation Algorithms 26
2.2.3. Pre-processing techniques for covering methods 27
2.2.3.1. Discretisation methods 27
2.2.3.2. Scaling-down techniques 28
2.2.3.3. Rule Representation 29
2.2.3.4. Rule Pruning 30
2.2.4. The covering methods developed in this research 32
2.3. Data Clustering 33
2.3.1. Overview of DC approaches 33
2.3.1.1. Hierarchical Clustering 35
2.3.1.2. Partitioning Clustering 38
2.3.1.3. Probabilistic Clustering 39
2.3.2. K-means 41
2.3.2.1. Improving K-means Performance 43
2.3.2.2. Scaling up K-means for large data sets 45
2.3.3. Research on the K-means method in this study 50
2.4. Summary 51

Chapter 3 Rule Extraction System with Adaptivity (RULES-A) 52
3.1. Preliminaries 52
3.2. Algorithm Description 53
3.3. Performance 62
3.3.1. Phase 1 Performance 63
3.3.2. Results after Phases 1 and 2 64
3.3.3. Results after Rule Simplification (Phase 3) 66

3.3.4. Comparison of the Overall Performance of RULES-A with C5 and RULES 3+ 70

3.3.5. Algorithm Complexity 72

3.4. Summary 74

Chapter 4 Improvements to RULES-A 76

4.1. Preliminaries 76

4.2. Improvements 77

4.2.1. Discrete Attributes 77

4.2.2. Continuous Learning 84

4.3. The RULES-A1 Algorithm 85

4.4. RULES-A1 Performance 89

4.5. Performance Improving Techniques 91

4.5.1. Early Stopping 91

4.5.2. Changing the Order of Training Objects 92

4.5.3. Performance Analysis 100

4.6. The Tic-Tac-Toe Problem 104

4.7. Summary 110

Chapter 5 Improvements to the K-means Algorithm 111

5.1 Preliminaries 111

5.2 Incremental K-means Algorithm 113

5.2.1 Conventions 113

5.2.2 Motivation 113

5.2.3 Evaluation of Distortion of Clusters 116

5.2.4 Algorithm Description 121

5.2.5 Performance 123

5.2.6 Further Improvements 132

5.3 Two-Phase K-Means Algorithm 134

5.3.1 Algorithm Description 134

5.3.2 Performance 137

5.4 Summary 145

Chapter 6 Selection of Number of Clusters for K-Means 146

6.1 Preliminaries 146

6.2 Number of Clusters 146

6.2.1 Values of K Specified within a Range or Set 147

6.2.2 Values of K Specified by the User 152

6.2.3 Values of K Determined in a Later Processing Step 152

6.2.4 Values of K Equal to Number of Generators 154

6.2.5 Values of K Determined by Statistical Measures 156

6.2.6 Values of K Equated to the Number of Classes 157

6.2.7 Values of K Determined through Visualisation 158

6.2.8 Values of K Determined Using a Neighbourhood Measure 159

6.3 Factors Affecting the Selection of K 161

6.3.1 Approach Bias 161

6.3.2 Level of Detail 161

6.3.3 Internal Distribution versus Global Impact 162

6.3.4 Constraints for f(K) 163

6.4 Number of Clusters for K-means 163

6.5 Performance 166

6.6 Summary 178

Chapter 7 Conclusion and Future Work 181

7.1 Conclusions 181

7.2 Future Research Directions 183

Appendix A Complexity Estimation of RULES-A 185

Appendix B Data Sets 188

References 192

List of Figures

Figure 2.1 The Machine Learning framework [Langley, 1996]. 6

Figure 2.2 The process model of DM [Chapman et al., 2000]. 12

Figure 2.3 Classification of covering methods. 21

Figure 2.4 A data set and the dendrogram obtained using a hierarchical clustering algorithm [Jain et al., 1999]. 37

Figure 2.5 The original K-means algorithm 42

Figure 2.6 Bradley's scalable framework for clustering [Bradley et al., 1998]. 46

Figure 3.1 The three phases of Rule Extraction System with Adaptivity (RULES-A). 54

Figure 3.2 Phase 1 – Induction. 54

Figure 3.3 Phase 2 – Pruning. 55

Figure 3.4 Phase 3 – Rule simplification. 55

Figure 3.5 The splitting operation. 57

Figure 3.6 Illustrative example of the execution of RULES-A. 60

Figure 3.7 Illustrative induced rule sets. 69

Figure 3.8 The comparison between complexities of C4.5 and RULES-A. 73

Figure 4.1 Training set [Pham and Dimov, 1996]. 80

Figure 4.2 A step by step execution of RULES-A for the training set in Figure 4.1. 81

Figure 4.3 The resultant rule set in Figure 4.2 represented as a decision tree. 83

Figure 4.4 The improved Rules Extraction System with Adaptivity (RULES-A1). 86

Figure 4.5 Phase 1 – Induction. 87

Figure 4.6 Phase 2 – Rule simplification. 88

Figure 4.7 The rule sets refinement during the learning process. 93

Figure 4.8 Conventions and 16 ending principles of the Tic-Tac-Toe game. 105

Figure 4.9 Examples of overlapping end principles of the Tic-Tac-Toe data set. 106

Figure 4.10 A decision tree for the Tic-Tac-Toe data set. 106

Figure 4.11 The resultant rule set from RULES 3+ with PRSET = 10. 108

Figure 5.1 The results of applying K-means (K=4) on two split regions. 114

Figure 5.2 The modified K-means algorithm incorporating the jumping operation proposed by Fritzke [Fritzke, 1997]. 115

Figure 5.3 The splitting of C_z into C_{z1} and C_{z2} after training. 118

Figure 5.4 The Incremental K-means algorithm. 122

Figure 5.5 The object distribution of the contrived data sets. 123

Figure 5.6 Clustering results of K-means, K-means with jumping operation, Incremental K-means and Incremental K-means with termination conditions. 128

Figure 5.7 Comparison of the running times of K-means, Incremental K-means and Incremental K-means with termination conditions. 130

Figure 5.8 Clustering results of K-means, Incremental K-means and Incremental K-means with cluster search. 133

Figure 5.9 The Two-Phase K-Means algorithm. 135

Figure 5.10 The average cluster distortions for eight versions of K-means when applied to the KDD98 data set, (a) 5 permutations and (b) the worst case. 140

Figure 5.11 The average execution times of tested algorithms on the KDD98 data set. 141

Figure 5.12 The average cluster distortions for eight versions of K-means when applied to the CoverType data set, (a) 5 permutations and (b) the worst case. 143

Figure 5.13 The average execution times of tested algorithms on the CoverType data set. 144

Figure 6.1 Data clustering as a pre-processing tool. 153

Figure 6.2 The relationship between clusters can have an effect on the clustering. 155

Figure 6.3 Inappropriate data sets for the K-means approach. 160

Figure 6.4 Variations of the two-ring data set. 160

Figure 6.5 The ratio S_K/S_{K-1} for data sets having uniform distributions 166

Figure 6.6 Comparison of the values of α_K calculated using Equation 6.2 (b) and the ratio S_K/S_{K-1}. 167

Figure 6.7 Data sets and their corresponding $f(K)$. 171

Figure 6.8 $f(K)$ for the 12 benchmark data sets. 177

List of Tables

Table 2.1	Assessment of ML classification algorithm [Michalski, 1998].	8
Table 3.1	Result of cross-validation 10-fold testing of RULES-A compared with C5.	63
Table 3.2	The performance of the rule sets before and after Phase 2.	65
Table 3.3	Comparison of rule sets' performance before and after Phase 3.	67
Table 3.4	Comparison of C5 and RULES-A results.	71
Table 3.5	Comparison of RULES 3+ and RULES-A results.	71
Table 3.6	The number of iterations over the training sets during Phase 1 of RULES-A.	73
Table 4.1	The main parameters of the selected data sets.	90
Table 4.2	Results of the ten-fold cross-validation testing of RULES-A1 against C5 and RULES 3+.	90
Table 4.3	Number of iterations required for RULES-A1 and RULES-A2 to converge on a rule set.	102
Table 4.4	The results of the cross-validation 10-fold testing of RULES-A2 against RULES-A1.	102
Table 4.5	The results of the cross-validation 10-fold testing of RULES-A3 against RULES-A1.	103
Table 4.6	The results of the cross-validation 10-fold testing of RULES-A3 against C5 and RULES 3+.	103
Table 5.1	Characteristics of test data sets	124
Table 6.1	The number of clusters used in different studies of the K-means method.	148
Table 6.2	The recommended number of clusters based on f(K).	179

Abbreviations

AI	Artificial Intelligence
ML	Machine Learning
DC	Data Clustering
DM	Data Mining
ARI	Adaptive Rule Induction
S1	Bradley's version of K-means with a buffer storing 1% of the data set.
S10	Bradley's version of K-means with a buffer storing 10% of the data set.
N1	Farnstrom's version of K-means with a buffer storing 1% of the data set.
N10	Farnstrom's version of K-means with a buffer storing 10% of the data set.
R1	The original K-means algorithm applied on 1% of the data set.
R10	The original K-means algorithm applied on 10% of the data set.
KM	The original K-means algorithm applied on the whole data set.
2PK	The Two-Phase K-means algorithm with a buffer storing 10% of the data set.

List of Symbols

Acc_{Val}	the accuracy of the rule set on the validation set.
ΔD	the estimated decrease in the total distortion error when the centre of a cluster is moved to a new position.
ΔI	the estimated increase in the total distortion error when the centre of a cluster is removed.
ΔM	the estimated change in the total distortion error when the jumping operation occurs.
$\propto_K$	a weight factor.
C	a cluster.
$d(w, x)$	the distance between the cluster's centre w and a position x in the Euclidean space.
$f(K)$	the evaluation function for the clustering result.
I_z	the distortion error of cluster z.
n	the number of objects of the test data set.
N	the number of objects belonging to the cluster (cluster's capacity).
N_d	the dimension of the Euclidean space.
S	the sum of the squared distances between the objects in the cluster and the centre of the Euclidean space.
S_K	the sum of the distortion of clusters when the data is clustered with K clusters by the K-means method.
x_0	the centre of the Euclidean space.
x_i^k	an object belonging to cluster C_k.
w	the centre of a cluster.

Chapter 1

Introduction

1.1. Background

In recent times, with new developments in information technology, information systems can store an increasingly large amount of historical data about daily activities. From the historical data, data mining and machine learning techniques can be used to extract previously unknown and potentially useful knowledge. The derived knowledge can then be applied to achieve economic, operational or other benefits.

Classification is a common task in data mining and machine learning. With the assistance of human teachers, a learning system can induce classifiers from the training data. Learned classifiers can be used to sort new objects into specified classes.

Rule induction is a common method of generating classifiers. Classifiers in rule induction are in the form of "If *conditions* Then *actions*" rules. Knowledge represented as rules is easy for users to understand and verify. In addition, the rules generated through the learning process can be utilised directly in knowledge-based systems.

Covering methods are common techniques of rule induction. These methods create rules directly by reasoning about the coverage by rules of the training data. They have been applied widely and successfully.

On the other hand, data clustering is often employed to discover natural groups and identify interesting distributions and patterns in the data. Clustering techniques classify objects into groups based on their similarities. The result of clustering is a scheme for grouping the data in a given data set or a proposal concerning regularities or dependencies in the data. With these characteristics, cluster analysis is often used as a pre-processing technique in data mining.

The K-means method is one of the most popular clustering techniques. K-means divides the data into disjoint partitions. Each partition is represented by its centre.

Although rule induction and clustering techniques have found several successful applications, a number of assumptions and constraints in their approaches have limited their capabilities and reduced their performances. For example, the sequential induction of rules by covering methods avoids the processing of relationships between rules. This approach can cause a negative effect on the performance of the resultant rule set. In clustering, the fixed number of clusters during the learning process of K-means requires an unreliable initialisation step. This constraint makes the performance of the method depend on chance. The main aim of this work is to identify these constraints and then to develop flexible management strategies to improve the performance of learning techniques.

1.2. Research Objectives

The overall research aim is to develop flexible management strategies for machine learning and data mining techniques to improve their performance.

The main research objectives are:

- To clarify the effects on the performances of learning algorithms of assumptions about and constraints on their learning approach.
- To develop new covering methods with a more general learning approach using a flexible strategy for managing the rule set.
- To improve existing clustering algorithms with a flexible management strategy.

1.3. Thesis Structure

Chapter 2 briefly reviews Machine Learning and Data Mining. The Data Mining process is discussed. Rule Induction and Data Clustering are also reviewed in this chapter.

Chapter 3 introduces the RULES-A, "Rule Extraction System with Adaptivity", a covering algorithm following the conquer-without-separation approach. The algorithm can induce the entire rule set simultaneously and has the ability to process continuous attributes directly. Rule pruning is also applied to improve the performance of the algorithm and reduce the complexity of the rule set.

Chapter 4 focuses on enhancements to RULES-A. The algorithm is improved with a capability for handling discrete attributes[1]. Rule pruning and continuous learning after

[1] In this thesis, the term 'discrete attribute' is used interchangeably with 'nominal attribute' which means an attribute with an unordered value, such as a label or a symbol.

pruning are embedded in the learning process. Continuous learning after pruning is examined and then an early stopping strategy is suggested. Learning from data sets with varying object orders is also applied to find potentially better rule sets. The performance of improved versions of RULES-A is evaluated on data sets with mixed attribute types.

Chapter 5 describes improvements to the popular K-means algorithm. First, the Incremental K-means algorithm is introduced to reduce the dependence of the algorithm on the initialisation of cluster centres. Second, the Two-Phase K-means algorithm is presented to enable the K-means algorithm to be scaled up for very large data sets.

Chapter 6 reviews and analyses current methods of selecting the number of clusters for the K-means algorithm. The chapter introduces a new measure to determine the number of clusters by comparing the clustering results for the studied data and data with the standard uniform distribution.

Chapter 7 summarises the thesis and proposes directions for further research.

Appendix A discusses the complexity of RULES-A.

Appendix B describes all the data sets used in the thesis.

Chapter 2

Literature Review

2.1. Machine Learning & Data Mining

2.1.1. Machine Learning

One of the long-term objectives of Artificial Intelligence (AI) research is the creation of machine intelligence. If a machine has intelligence, it does not only behave as though it has knowledge equipped by its creator, but it also learns new knowledge from the environment by itself to improve its own performance. Knowledge thus learned by a machine can even improve on human intelligence. Such self-learning is essential for intelligent objects to exist in a changing world. Therefore, Machine Learning (ML) is the key to artificial intelligence.

ML consists of techniques to "acquire high-level concepts and/or problem-solving strategies through examples in a way analogical to human learning" [Michalski et al. 1998]. Through interaction with the environment, an intelligent machine can collect observations and then generalise them to extract useful knowledge. With this new knowledge, the intelligent machine can adapt and improve its behaviour according to changes in the environment.

A framework for ML is shown in Figure 2.1 [Langley, 1996]. In this framework, the *learner* (*learning* in the diagram) collects observations from the *environment* in order

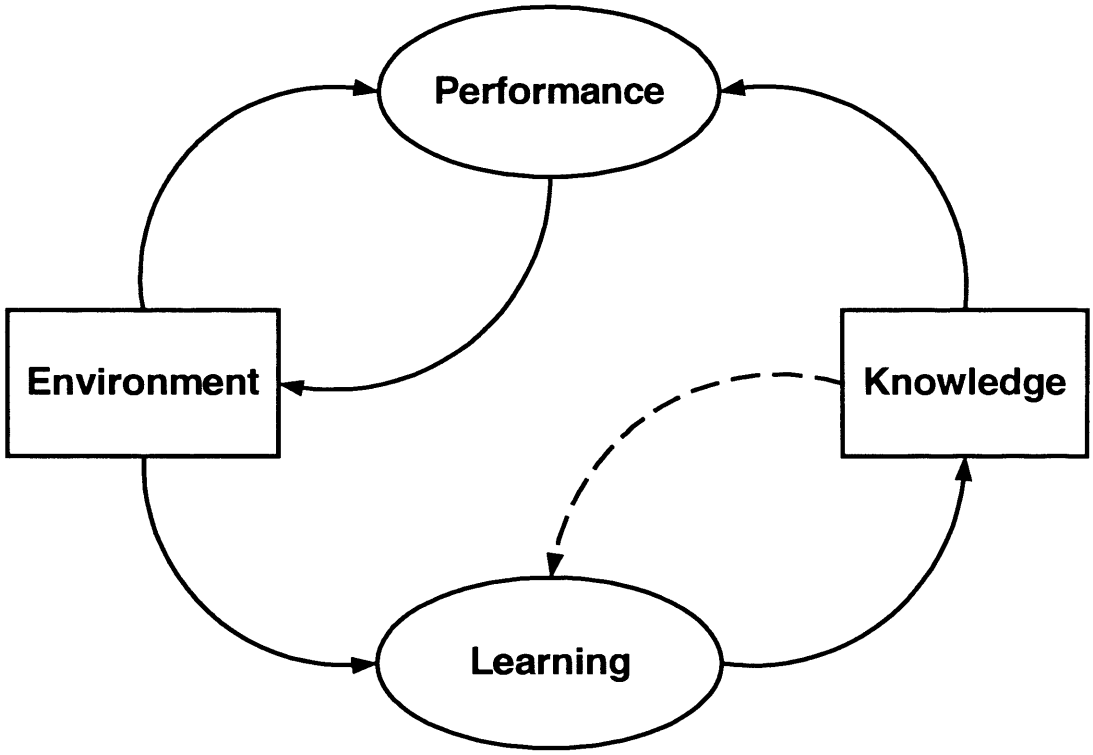


Figure 2.1 - The Machine Learning framework [Langley, 1996].

to extract useful information to update its *knowledge*. The learner uses that knowledge to *perform* tasks and interact with the environment. The dashed line, which indicates an optional link from the knowledge to the learner, means that the learner can use its learned knowledge to improve its learning strategy.

There are two types of learning. The first type is *supervised*, in which feedback takes a large role in guiding the learning process. This feedback is often provided by a human tutor. The second type of learning is *unsupervised*, where there is no feedback. The learner can use unsupervised learning to discover new knowledge by itself.

Classification is one of the main tasks in supervised learning. Learning from a set of pre-classified examples, classification techniques can categorise new observations into pre-defined groups. There are two main phases in classification. Firstly, the classifier learns from the training set of examples labelled with the desired class. Secondly, the resultant classifier is used to classify previously unseen observations. The assessment criteria for classification algorithms are summarised in Table 2.1 [Michalski, 1998].

Data Clustering (DC) is a typical unsupervised technique. DC groups similar objects into clusters. Objects within a cluster are similar to others in the same cluster and dissimilar from those in different clusters. DC is often used as a preliminary data analysis tool to discover potential regularities and principles, and to generate hypotheses concerning the nature of the data. DC is also a popular compression technique in data communication.

Table 2.1 – Assessment of ML classification algorithms [Michalski, 1998].

Criterion	Comments
Accuracy	Percentage of correct classifications
Efficiency	Computational complexity
Robustness	Stability against noise and incompleteness
Special requirements	Incrementality[1], concept drift[2]
Concept complexity	Representational issues
Transparency	Comprehensibility for the human user

[1]Incrementality: capability of refining the previous knowledge.

[2]Concept drift: capability of changing the meaning of concepts from time to time.

2.1.2. Data Mining

Rapid developments in the number as well as the scale of computerised enterprise systems makes many large information sources available. For example, a global enterprise may have millions of daily transactions. A busy web site can be accessed millions of times per day. All these activities are recorded in databases. This logged information contains useful knowledge, which can be analysed to improve business activities and direct future developments. At the same time, advances in computer technologies also bring large increases in computational abilities. There is a requirement for research to develop technologies to use this computational power to discover the recorded information. A young branch of computer science, Data Mining (DM), is a response to this need.

Mitchell [Mitchell, 1999] gave the following definition for DM:
"Data Mining: using historical data to discover regularities and improve future decisions."

With a more application-oriented mindset, Fayyad [Fayyad et al., 1996] stated that:
"Data Mining, which is also referred to as knowledge discovery in databases, means a process of nontrivial extraction of implicit, previously unknown and potentially useful information, such as rules, constraints, regularities data in databases."

Using the recorded information, normally stored in databases, from several areas or disciplines, DM techniques attempt to discover new knowledge. Learned knowledge

can manifest itself in several forms, such as classifiers, predictors, associations or segmentations of data. DM results are often used in a supportive manner for decision making or operational improvement. DM has been applied in many practical applications in biomedical and DNA data analysis, financial data analysis, and engineering [Bose and Mahapatra, 2001; Grossman et al., 2001; Han, 2001]. Several potential problems are still waiting for DM research to be applied to them [Schafer et al., 2001].

With the same purpose of "learning from data", ML algorithms have a central role in DM. However, these algorithms must be developed to suit the particular requirements of DM. The first challenge is the higher level of noise in DM data. The robustness criterion of an algorithm becomes more important while other criteria may be partly relaxed. The second challenge is the large size of processed data sets. DM data sets often have extremely large sizes. Comparing the benchmark data sets of the UCI (University of California Irvine) DM repository [Hettich and Bay, 1999] and the UCI ML repository [Blake et al., 1998], DM data sets are typically 10 and 100 times larger than ML data sets in terms of the number of attributes and the number of objects, respectively. The size of DM data sets in practice is often in the tera-byte range. With such sizes, the processing time is extremely long. In addition, with traditional algorithms, the data set is often assumed to be loaded fully into memory. Although the memory size in computers has expanded rapidly in recent times, this assumption is hardly consistent with current increases in data size. Therefore, the application of probabilistic, sampling, buffering, parallel and incremental techniques to learning algorithms becomes more important.

DM techniques are task-driven and data-driven. Instead of the concentration on symbolic and conceptual knowledge of ML, most developments in DM are tied closely to practical applications and the characteristics of their data. For example, Association Rules is a DM technique that explores relationships between items in market transactions. The learning algorithm is based on data characteristics that are often binary and very sparse, to find correlations between items in transactions.

DM may be shown as an iterative process with 5 stages (Figure 2.2) [Chapman et al., 2000]. A stage can be refined by feedback from later stages.

The first stage, **Business and Data Understanding**, makes a bridge between the DM system and the existing database system. This is carried out through the interaction between DM consultants/developers and users. The DM consultants study domain knowledge about the existing system, including system and knowledge structures, available data sources, the meaning, role and importance of data entities. Unlike with traditional problem solving methods in which the problem is defined precisely in the first stage, DM consultants start with the user's preliminary requirements and recommend potential problems that could be solved with the available data. The set of potential problems is refined and narrowed in later stages of the DM process. Data sources and specifications, which are related to potential problems, are also recognised.

Data Preparation consists of using pre-processing techniques to transform the data and improve its quality to suit the requirements of the learning algorithms. Most current DM algorithms only work on a single, flat data set, so that data has to be

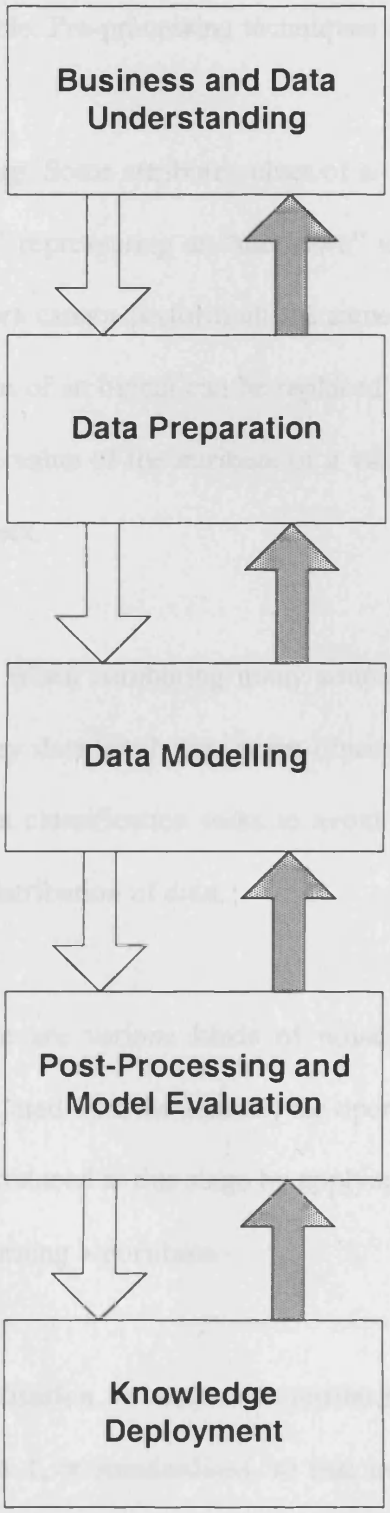


Figure 2.2 - The process model of DM [Chapman et al., 2000].

extracted and transformed from distributed, relational or object-oriented databases to a database with only one table. Pre-processing techniques include:

(1) Missing value processing. Some attribute values of an object can be left empty or can have a special value '?' representing an "unknown" value. This often happens in medical data because doctors cannot perform all the same tests on their patients. The missing value of an attribute of an object can be replaced by the most common value of the attribute, the average value of the attribute or a value calculated by correlation with other values of the object.

(2) Duplicate elimination. When combining many sources to form a single table or reducing certain unnecessary data attributes, some objects could be identical. These objects can be eliminated in classification tasks to avoid redundancy. However, this elimination can affect the distribution of data.

(3) Noise reduction. There are various kinds of noise associated with the input sources, such as those associated with the sensors, the operators or the communication environment. Noise can be reduced at this stage by applying statistical methods or can be processed later by the learning algorithms.

(4) Standardisation/Normalisation. A continuous attribute can be normalised, so that its value is in the range 0 to 1, or standardised, so that its average value is 0 and its standard deviation is 1. These techniques balance the effects of the attributes on the learning algorithms. Where there are mixed attributes, weighting techniques can be used to balance between the effects of continuous and discrete attributes.

*(5) **Discretisation***. Some learning algorithms require continuous attributes to be discretised before their application. A continuous attribute can be discretised in an unsupervised manner into equal intervals or variable-intervals using statistical measures. It can be discretised in a supervised manner with respect to the object's class label [Dougherty et al., 1995]. Many discretisation techniques have been developed recently using entropy-based [Fayyad and Irani, 1993], distance-based [Cerquides and Lopez de Mantaras, 1997], wrapper-based and "minimum-description-length principle"-based [Cai, 2001] approaches.

*(6) **Feature Extraction and Construction***. Useful and meaningful information regarding objects is selected and extracted in the first instance by applying domain knowledge. However, statistical information, for example the average values of attributes, and information from combined attributes, made up of two or more attributes by logical or mathematical methods, are also useful. In addition, feedback from the learning algorithm in the latter stages of the DM process can require the extraction of some extra features to improve the overall performance.

*(7) **Dimension reduction***. DM data often has hundreds of attributes. Some useful feature extraction techniques have been developed to find attributes which are rich in information. Data also can be filtered to find attributes to suit the characteristics of the learning task using wrapper-based techniques [Kohavi and John, 1998], mathematical programming [Bradley et al., 1998b] or principal component analysis [Fedorov et al., 2003].

*(8) **Instance reduction***. The extremely large volume of data involved slows down the entire DM process. Instance reduction techniques are very useful in decreasing the amount of data while only slightly degrading the entire performance. Data sampling, the most common instance-reduction technique, is used to find meaningful representatives, in terms of frequent objects. It has proved to be a useful tool for several tasks, such as text classification [Lee and Corlett, 2003], learning robot navigation [Winters and Victor, 2002], database accessing [Bisbal and Grimson, 2001] and training control systems [Horch and Isaksson, 2001].

The problems identified in the first stage are mainly solved in the third stage, **Data Modelling**. The processed data is utilised by the learning algorithms to find hidden and unknown principles.

The most important task in this stage is the selection of appropriate techniques for the identified problems. The problems can be classified into one of the main DM tasks using their declaration. However, each DM task can utilise a number of different techniques. In addition, a technique often requires some parameters to be specified by the user based on the characteristics of the data. Therefore, the selection of appropriate techniques is dependent on the experience of DM consultants and is often performed in a "trial-and-error" manner.

The learning algorithms also work closely with the pre-processing of the previous stage. The pre-processing techniques have to be selected carefully to make sure there is no loss of valuable information for the learning algorithms. Some specific techniques have to be carried out due to the requirements of the particular learning

algorithm. For example, CN2 [Clark and Niblett, 1989, and Clark and Boswell, 1991] requires continuous attributes to be discretised before applying the algorithm.

The fourth stage is **Post-processing and Model Evaluation**. The preliminary result, learned from the third stage, is introduced to users in order to validate and refine the solution strategies. With the combination of identified problems and potential techniques, several solutions can be induced.

Most of the techniques need some explanation from DM consultants in order for users properly to understand their results. Certain techniques, such as Neural Networks, require extra methods to transform their results into understandable forms. Other techniques, such as Data Clustering, have no common method to evaluate their results. In such cases, visualisation becomes a useful means for the user to evaluate the DM results.

The DM results are validated with real data in the evaluation mode, which is carefully controlled by the DM consultants and users. If the evaluation does not satisfy the user's expectations, or other potential solutions are available, or the processing carried out in earlier stages is shown to be unsuitable, earlier stages in the DM process can be repeated.

When the evaluation on real data of the DM solutions gains the user's acceptance, the learned model is deployed in a suitable and convenient form for users in the fifth stage, **Knowledge Deployment**. The final DM solutions are often deployed in web pages, which can be accessed throughout departments of the user's company.

Authorised users can then apply the deployed model to analyse recent business data to make business decisions.

Current techniques lack self-updating abilities to reflect changes in the business context. Any modification requires a repeat of the DM process. Therefore, commercial DM software is often designed as a flexible environment, in which DM consultants can access and manipulate data, solve problems by means of learning models and test solutions. From the result of evaluations and feedback from users, DM consultants can easily make modifications to the DM process. The DM process is refined through interaction between DM consultants and users until it reaches the expectation of the latter.

The close relationship between stages in the DM process is important for DM research. A DM algorithm cannot be developed in isolation without considering its application context and is often created to serve a specific purpose. Understanding the applied context is therefore essential for the development of a DM algorithm. The techniques applied in previous stages can also affect the results of DM algorithms in a subsequent stage of the process.

2.2. Inductive Learning

Induction is "reasoning from specific cases to general principles" [Forsyth, 1989]. Instead of remembering all experiences, which are increasing rapidly in the information age, human intelligence uses inductive learning to explore historical observations and extract a limited number of general principles. Based on these learned principles, the user can predict what will happen in the future and adopt an appropriate behaviour.

Rule Induction is the branch of inductive learning in which the induced principles are in the form of rules such as "IF *condition* THEN *action*". Given data comprising examples (or "objects") pre-assigned to desired classes, rule induction algorithms can learn rule sets, which can be used to classify previously unseen data. The data used to construct the learning system is often called the training set. A part of the data used to test the system is often called the test set.

Knowledge in the form of rules is easy for users to understand and verify, and can be utilised as classification or prediction models. Furthermore, the rules generated through the learning process can be employed directly in knowledge-based systems to automate the knowledge acquisition process.

Two main approaches exist to extract rules from data. The first approach, known as decision trees induction, creates classified trees that are then transformed into rule sets. The second approach, known as the covering method, creates rules directly from

the data in a way that is more natural. Many algorithms have been developed for both approaches, showing both their efficiency and popularity.

2.2.1. Decision Trees

Decision Trees is one of the most popular methods used to accomplish classification tasks, and are available in almost all DM commercial software. Decision Trees organise consequent decisions in a single-parent tree. Although binary decision trees are often used, decision trees can also be in the form of multi-branch trees.

The most common family of decision trees is ID3 [Quinlan, 1986]. ID3 has been improved several times by a number of researchers, the most recent descendants being C4.5 [Quinlan, 1993] and C5 [Rulequest Research, 2001].

The general decision tree forming procedure [Hunt et al., 1966] for a training data set T starts from a single root node and operates recursively as follows:

- If T satisfies a particular *stopping criterion*, the node is a leaf labelled with the most frequent class in the set.
- If the *stopping criterion* is not satisfied, a decision is made on an attribute, selected by a specific *heuristic measure*, to partition T into subsets T_i of objects. The procedure is repeated on these new subsets.

If it is assumed that there is no noise in the training set, the procedure stops when T contains objects of a single class. To avoid over-fitting in the presence of noise, the

procedure can be stopped earlier by applying pruning techniques. The heuristic measure plays a major role in deciding the quality of the formed decision tree. It helps the forming procedure to select the attribute upon which to divide a node, the divided values of the selected attribute and the number of divided branches.

The decision tree forming procedure utilises the *divide-and-conquer* approach. After each decision, the training set is *divided* into subsets. Each subset is *"conquered"* separately from other subsets in any level. With this strategy, the complexity of the procedure is rapidly reduced. Another advantage of the method is easy understanding and explanation by visualisation for users.

The divide-and-conquer approach has a number of deficiencies. A similar sub-tree may exist many times, in particular in problems that are terminated by a fixed-size tuple of conditions (see section 4.6). The attribute-approach of a decision tree is also unsuitable for data with a large number of missing values, such as medical data sets. For such data sets, the incorrect evaluation made on an attribute can mislead the learning process.

2.2.2. Covering Methods

A proposed classification of current covering methods is shown in Figure 2.3. The first division is made on the strategy employed to induce rules. The "separate-and-conquer" approach induces one rule at a time and sequentially forms rules on the objects not covered by the rule set formed so far. The "conquer-without-separation" approach forms all rules at once.

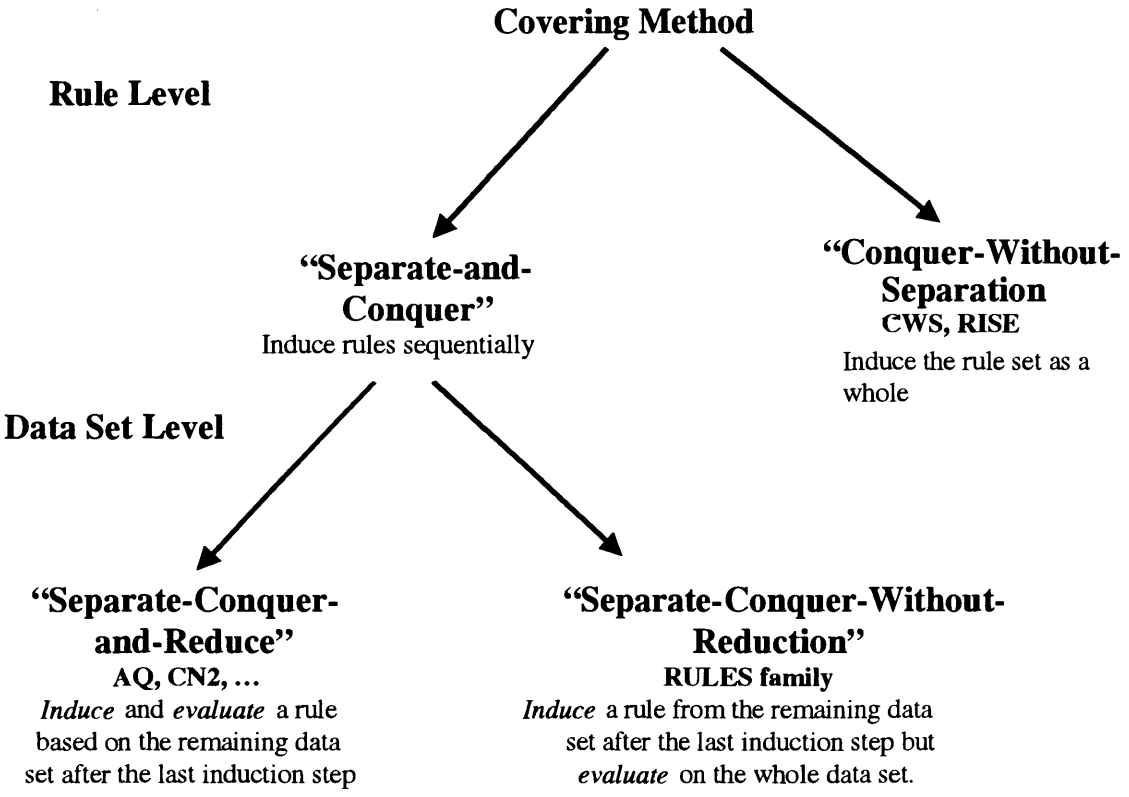


Figure 2.3 – Proposed classification of covering methods.

The second division in Figure 2.3 further specialises methods in the "separate-and-conquer" approach according to their treatment of data. The first branch, called "separate-conquer-and-reduce", induces and evaluates a new rule based on the remaining data after the last induction step. After each induction step, objects covered by the rule set formed so far are omitted from the data set. With the other method, called "separate-conquer-without-reduction", a new rule is induced from the remaining data after the last induction step but is evaluated on the entire data set. After each induction step, objects covered by the new rule are marked "covered" instead of being omitted.

2.2.2.1. Separate-Conquer-and-Reduce Algorithms

The separate-conquer-and-reduce approach is the most popular branch containing several algorithms. The general induction procedure for a training set is a recursive process:

- Form a rule with the highest evaluation measure.
- Omit objects covered by the formed rule.
- Repeat the above two steps until the training set is empty.

The rule forming procedure is different for different covering algorithms. The method used in the AQ family [Michalski, 1977, Michalski et al., 1986, Michalski et al., 1998, and Kaufman and Michalski, 1999] is data-driven. Starting with uncovered examples as seed examples, a sophisticated process is used to produce rule candidates. The candidate with the best evaluation measure on the training set is selected as the new rule.

Another method to form rules is attribute-value pair oriented. CN2 [Clark and Niblett, 1989, and Clark and Boswell, 1991] uses beam search to find the complex of attribute-value pairs with the highest evaluation measure. RND [Liu, 1996, and Liu, 1998] uses the discretisation technique Chi2 to find the most frequent attribute-value pair to initialise the new rule. The ILA family [Tolun and Abu-Soud, 1998 and Tolun et al., 1999] uses the same strategy as CN2 to form rule candidates but only evaluates complexes of the same size.

The main advantages of the separate-conquer-and-reduce approach are that the computations required decrease during the learning process and that it does not need to take into account the relationship between the rules in the rule set. After each rule induction step, the size of the training set is reduced. Thus, the complexity of rule forming and evaluation decreases during the learning process. Later rules are induced by only considering the current training set without correlation with previously induced rules. Therefore, the learning process is straightforward.

The separate-conquer-and-reduce approach also has drawbacks. In particular, the relationship between the rules is not explicitly defined and this could have a negative effect on both the rule set induction and application phases. During the induction phase, although a new rule is generated considering only objects not covered by the rule set formed so far, it may also classify other objects in the training set. This focus only on objects not covered so far could lead to a rule set with a poor evaluation measure. At the end of the rule induction phase, the coverage of each rule should be recalculated on the whole training set. If this is not done, the rule set will not contain sufficient information to classify objects covered by more than one rule and its

performance on the training data will be different from that achieved during the learning phase.

The rule searching process in the existing implementations of the separate-and-conquer approach is data-driven and relatively simple. Each object defines a set of possible hypotheses that will be considered to form a new rule. Because the search space is limited by the selected object, the rule forming process could lead to a local maximum (the best rule within the considered set of hypotheses). A backtracking or pre-initialization strategy has not been investigated empirically in existing separate-and-conquer methods. To address this problem in RND [Liu, 1996, and Liu, 1998], it is proposed that the object representing the most frequent pattern be used to initialise the search. To find this object, a metric is utilised to measure the occurrence frequency of different patterns [Liu and Setiono, 1995]. However, this metric has a high complexity that is a function of the number of possible values for each attribute. There are also non data-driven algorithms. For example, ILA [Tolun and Abu-Soud, 1998] and ILA2 [Tolun et al., 1999] induce rules by grouping them in layers depending on the number of conditions included in them. There is an unsolved problem concerning areas covered simultaneously by rules in the same layer in these algorithms.

Another problem with covering methods employing the separate-and-conquer approach is the *fragmentation* of the example space into small areas covered by different rules [Domingos, 1996a]. For example, if noise exists in the training data, an early-induced very general rule for one class may break the object space of different classes into many small sub-areas. This could lead to the creation of a large number of

more specific rules. By applying pre-pruning techniques, this problem could be avoided.

2.2.2.2. Separate-Conquer-Without-Reduction Algorithms

The separate-conquer-without-reduction approach was first established at Cardiff University with the RULES family of algorithms [Pham and Aksoy, 1995a; Pham and Aksoy, 1995b; Pham and Dimov, 1996; Pham and Dimov, 1997]. The general induction procedure for a training set is a recursive process as follows:

- Form a rule to classify a number of uncovered (unmarked) objects which has the highest evaluation measure on the entire training set.
- Mark objects covered by the formed rule.
- Repeat the above two steps until all objects of the training set are marked.

Rules formed using this approach are better evaluated because of their use of the available information. Although the rule forming procedure can select from a decreased set of candidates during the learning process, the rule evaluation has a constant complexity. The relationship between rules is implicitly represented in the ratio between marked and unmarked objects covered by a new rule.

The evaluation of rules on the entire data set, including marked and unmarked objects, can lead to overlapping rules due to a partial correlation between object attributes. The performance of the rule set is not affected, but it may contain more rules than required to cover the training data. The ratio between marked and unmarked objects covered by a new rule should be taken into account when assessing its performance.

2.2.2.3. Conquer-Without-Separation Algorithms

Algorithms employing the conquer-without-separation approach induce rules in parallel instead of sequentially as is the case in separate-and-conquer methods. Obviously, this approach has a higher complexity due to the simultaneous induction of a *combination* of rules. The conquer-without-separation approach forms the whole rule set simultaneously. Algorithms that apply this approach must not only form rules covering objects from the same class but also maintain the relationship between rules of different classes. In addition, the balance between the quality of the rules generated and the compactness of the rule sets is maintained.

In the CWS algorithm [Domingos, 1996a], a general-to-specific search strategy is carried out by starting with the empty rule set and then adding a condition to a rule or by adding a new rule to the rule set in order to increase the rule set accuracy. Each new rule is formed either by specialising further an existing rule or by creating a completely new rule. The learning process is stopped if the accuracy of the rule set does not increase. However, in practical problems, during the specialising process, the rule set accuracy may temporarily decrease. The requirement of a monotonic increase in accuracy after each specialising step limits the search capabilities of the algorithm.

Another algorithm, RISE [Domingos, 1994], employs a specific-to-general search by considering the training set as an initial rule set that is then generalised by removing conditions from the existing rules or removing identical rules from the rule set. The learning process is stopped if the accuracy of the rule set does not increase. This

method is complex because of the initial size of the rule set. It expands the nearest rule to cover an object instead of considering of the most general rule for the class of the object. During this process, the consistency of the newly formed rule is not maintained. Thus, the relationship between rules in the rule set is not maintained efficiently with this method.

2.2.3. Pre-processing techniques for covering methods

2.2.3.1. Discretisation methods

Existing covering algorithms can process only symbolic or discrete attributes directly. To process continuous attributes, their values must be discretised first. Those attributes are then treated as discrete attributes.

Some problems are associated with this discretisation step. The conversion of continuous attributes into discrete attributes restricts the number of value-ranges which can be used and, as a result, limits the flexibility of the induction process. The discretisation technique, if applied as a preprocessing step, may unwittingly remove some valuable information from the training data [Ventura and Martinez, 1995a; and Ventura, 1995b]. In addition, by discretising each attribute independently, any existing high-order correlation between attributes may be reduced [Ventura and Martinez, 1995a; Ventura, 1995b].

If supervised discretisation is applied, to avoid creating a large number of intervals, current algorithms accept some loss of data by setting high thresholds [Dougherty et al., 1995]. The loss of data during the discretisation process may affect the quality of the generated rule set.

If unsupervised discretisation is performed and some objects belonging to different classes are close to one another in Euclidean space, this could have a detrimental effect on the induction capability of the algorithm. The reason for this is that the class of the objects is not taken into account during the discretisation process.

For example, all discretisation algorithms degrade significantly the performance of C4 [Quinlan, 1993] when compared with the results obtained when techniques for direct processing of continuous attributes are applied [Ventura and Martinez, 1995a]. In contrast to decision-tree techniques, at the time this work was carried out, there was no covering algorithm that could process continuous attributes directly. If such capabilities are embedded in these algorithms, this will improve the quality of the generated rule sets.

2.2.3.2. Scaling-down techniques

Scaling-down techniques, such as data clustering or sampling, can be used before the application of induction algorithms to reduce the number of objects considered by the learning process [Pham et al., 2000], thus reducing the complexity of the induction process.

The use of data clustering algorithms that are not appropriate, however, could significantly reduce the efficiency of the inductive learning algorithms. In particular, there is a difference between the Euclidean surface defining a sub-space when data clustering and when covering methods are applied. For example, hyper-polygons or hyper-ellipses are respectively created by these two techniques. Questions also arise regarding the selection of appropriate parameters when clustering is applied to real problems. The difference between the optimisation criteria that drive data clustering methods in minimising the distortion, and those employed by covering methods in maximising the generality and accuracy of rule sets, must be considered.

Data clustering is also an unsupervised technique, so that, in certain cases, objects belonging to different classes can be grouped together. This may decrease the efficiency of the subsequent induction process.

2.2.3.3. Rule Representation

The sub-space defined by each rule is a hyper-rectangle, in which, for discrete attributes, each dimension in Euclidean space is either one unit or an entire axis (when the attribute is not present in the antecedent of the rule). To cover all objects in the training data, it may be necessary to form rules that cover a small number of objects. In many cases, such rules could be considered as accounting for noise, by not paying proper attention to the rules covering objects close to them.

The antecedent part of each rule that is formed by applying the separate-and-conquer approach contains a conjunction of conditions, called attribute-value pairs. This rule

representation technique, however, does not utilise the specific characteristics of continuous attributes. By defining appropriate value ranges for continuous attributes, it is possible to reduce the number of rules created. Unfortunately, the use of uniform intervals to discretise continuous attributes often limits the induction capabilities of covering algorithms.

A different rule representation format is proposed by Domingos [Domingos, 1996b, and Domingos, 1996c]. In this format, there are conditions for each attribute in the antecedent part of the rules. In particular, these conditions could be an attribute-value pair, a range of values for continuous attributes or a condition that is always satisfied. This rule representation technique is more flexible and allows more generic rules to be formed.

2.2.3.4. Rule Pruning

Two types of rule pruning techniques exist [Furnkranz, 1997]. The first technique is called pre-pruning. It is applied to limit the search space of separate-and-conquer methods. This technique uses a threshold to stop the rule forming process when a particular measure for rule evaluation is not met. A prerequisite for applying this technique is for the evaluation measure to have a monotonically decreasing value during the learning process. As mentioned previously, the rule evaluation in the separate-and-conquer approach is limited to only a sub-set of objects in the training set. Therefore, it does not assess correctly the classification capabilities of the formed rules. Current evaluation measures [Furnkranz, 1999] do not take these factors into account.

The second technique is known as post-pruning. It is applied to the induced rule set to make it simpler and more general. This pruning is carried out on the whole rule set instead of on each rule individually, in a similar way to the rule evaluations performed in the conquer-without-separation approach. This technique may be considered a simplified conquer-without-separation method that starts from a given rule set, which is then simplified by either removing conditions or complete rules from it. These two removal operations simplify the rules and make them more general. From a geometric viewpoint, the two operators have a completely different effect on the rule set. The first operator removes conditions from rules, enlarges the covering area of the rules, so that it increases the coverage of the rule set and hence decreases the percentage of unclassified objects. However, it also increases the percentage of misclassified objects. The second operator removes complete rules from the rule set, so that it decreases the coverage of the rule set and as a consequence of this, increases the percentage of unclassified objects. Thus, the effect on the accuracy of the rule set of these two removal operators is different.

Two existing post-pruning algorithms, REP [Brunk and Pazzani, 1991] and GROW [Pagallo and Haussler, 1990], perform exhaustive search and do not take into account the rule characteristics. This increases significantly the computational complexity of these algorithms [Cohen, 1995; Furnkranz, 1997]. The other existing post-pruning algorithm, IREP [Furnkranz and Widmer, 1994; Cohen, 1995], is built into the rule forming process of the separate-and-conquer approach, and therefore cannot be used as a stand-alone post-pruning technique.

2.2.4. The covering methods developed in this research

Of the three main types of covering methods, the separate-conquer-and-reduce approach is the most restricted. The number of rules in the rule set monotonically increases during the induction phase without any reconsideration strategy. The relationship between rules is not explicitly defined in this approach.

Formed rules from the separate-conquer-without-reduction approach are better evaluated in terms of the available information. However, the relationship between rules is not appropriately processed.

The conquer-without-separation approach is the most flexible. There is no limit on the size of the rule set. Rules can be added to or omitted from the rule set during the induction phase. Relationships between rules are maintained in terms of the accuracy of the entire rule set. However, the size of the rule set for current algorithms adopting this approach only increases or decreases monotonically.

The use of separate and inappropriate methods of pre- or post-processing techniques can affect the overall performance of the induction method. These techniques should be related to the approach of the main learning algorithm.

Chapter 3 introduces a new conquer-without-separation algorithm, Rule Extraction System with Adaptivity (RULES-A). The size of the rule set is *flexibly* managed. Rules can be added to or omitted from the rule set during the induction phase. A

pruning technique is embedded in this method. Another distinguishing feature of RULES-A is that this algorithm processes continuous attributes directly, so that a discretisation step is not required.

Chapter 4 describes how RULES-A is improved by adding the ability directly to process discrete attributes and automatically to prune rules. The continuation of learning after pruning and an early stopping strategy are added to the method to improve its learning speed and accuracy.

2.3. Data Clustering

As previously stated, data clustering (DC) is an important data exploration technique for grouping similar physical or abstract objects. The technique allows objects with common characteristics to be lumped together in order to facilitate their further processing. DC is an unsupervised technique that generates hypotheses based on the provided unlabelled objects. This makes this method a very attractive data processing technique for a wide range of applications [Romesburg, 1990].

2.3.1. Overview of DC approaches

A definition of DC terms is needed before applying the technique to practical problems. The following aspects have to be considered based on the nature of the problem:

(a) **Object representation**. An object's meaningful characteristics are extracted by using a Feature Extraction technique. Selected attributes are then considered by Data Preparation techniques such as Feature Selection, Data Cleansing, Missing Values Processing, etc.

(b) **Similarity measurement**. Based on selected attributes representing an object, a distance metric is selected to measure the similarity between two objects. The Euclidean distance is the most common distance metric utilised.

(c) **Cluster definition**. A cluster's characteristics, for example its shape (convex/arbitrary) and cluster's border (line/curve, clear/fuzzy), should be specified. The relationship between clusters (overlapping/disjoint) and the similarity measurement of clusters also have to be specified.

(d) **Clustering criterion**. The clustering criterion can be single/multi-criteria optimisation or the building of a clustering structure (tree, graph or list).

(e) **Number of clusters**. The way this parameter is determined depends on the method and the characteristics of the data.

(f) **Clustering validity and cluster validity**. The method of validating a cluster set and a cluster has to be determined. The test strategy is also selected correspondingly.

(g) **Methods to understand, explain and apply** the clustering results.

These seven aspects have to be considered closely. There are many solutions. This has led to more than 10 different approaches in DC [Han, 2000; Berkhin, 2001; Duda et al., 2001; Grabmeier and Rudolph, 2002]. In this thesis, three main types, Hierarchical Clustering, Partitioning Clustering and Probabilistic Clustering, are reviewed.

The first aspect, *object representation*, is common to most of the methods and is not analysed in this section.

2.3.1.1. Hierarchical Clustering

Cluster representation: A cluster is a sub-tree of sub-clusters and has only one "father". A leaf node is a data object. The cluster's shape is not pre-specified. Clusters are disjoint. A cluster relates to its siblings by a *linkage metric* [Kaufman and Rousseeuw, 1990].

Clustering criterion: Building of a *dendrogram* (a cluster hierarchy or a tree of clusters) is achieved by minimising the similarity between sibling nodes. An example dendrogram is shown in Figure 2.4.

Number of clusters: This can be a predefined parameter or a user's decision based on an inspection of the dendrogram.

Clustering validity and cluster validity: The determination of validity is achieved by visual examination of the results.

Main approaches: *Agglomerative* (bottom-up) and *divisive* (top-down) [Jain and Dubes, 1988; Kaufman and Rousseeuw, 1990]. An agglomerative clustering algorithm starts with all leaf clusters (one-object clusters) and recursively merges two or more most similar clusters. A divisive clustering algorithm starts with one cluster including all objects and recursively splits the most appropriate leaf cluster to two or more sub-clusters.

Advantages:

- Easy understanding and application for users.
- Flexibility regarding the level of granularity.

Disadvantages:

- Vagueness of termination criteria.
- High complexity, $O(n^2)$ with n being the number of examples.
- Greedy forward or backward computation.

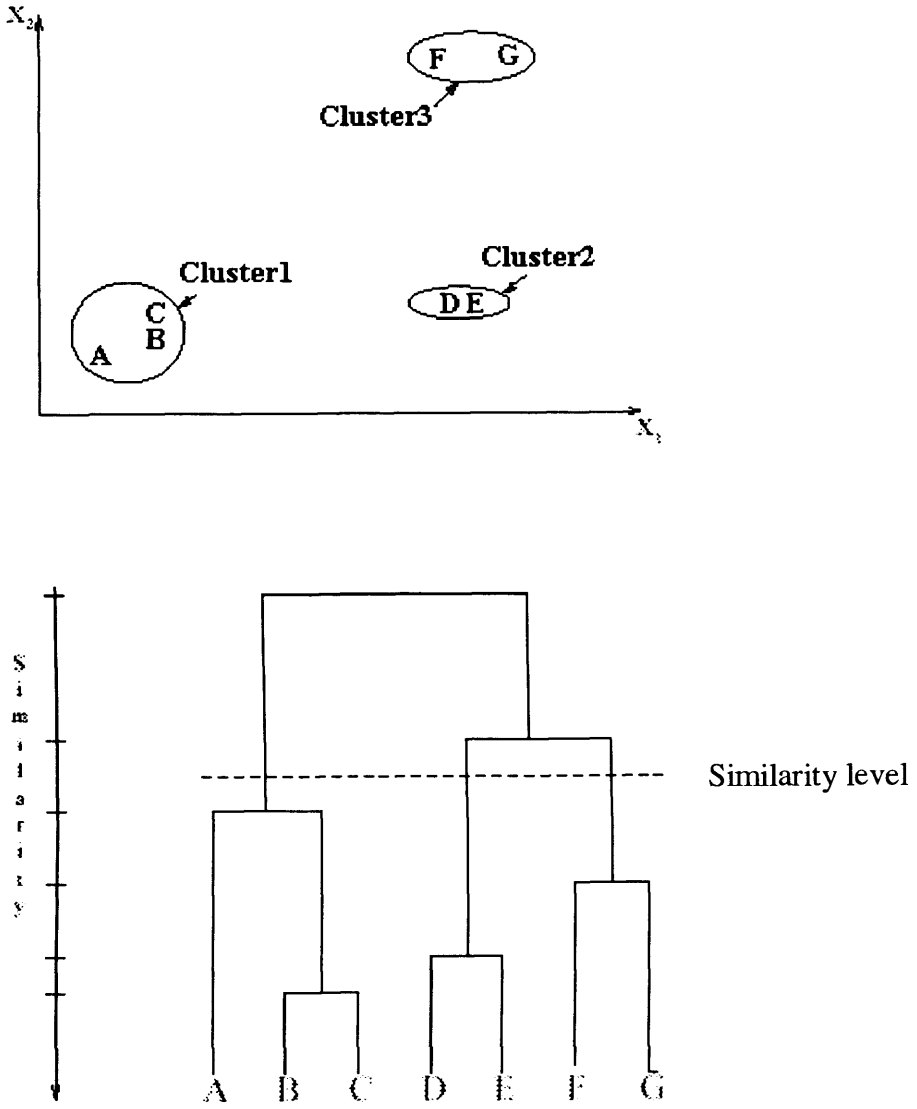


Figure 2.4 – A data set and the dendrogram obtained using a hierarchical clustering algorithm [Jain et al., 1999].

2.3.1.2. Partitioning Clustering

Cluster representation: A cluster is a partition of data and represented by its centre. An object belongs to the nearest cluster measured by the distance between the object and the cluster centre. The cluster's shape is a convex polygon. Clusters are disjoint.

Clustering criterion: The minimising of the inter-cluster similarity measurement (distortion) and/or the intra-cluster similarity measurement.

Number of clusters: This can be a predefined parameter or indirectly specified by other parameters.

Clustering validity and cluster validity: There are many methods of determining validity.

Main methods: The main partitioning clustering methods are K-means, PAM, CLARA and CLARANS.

K-means [MacQueen, 1967] is the proto-typical partitioning clustering algorithm. Each cluster is only represented by its centre, which is the mean of objects belonging to the cluster. The number of clusters is the only predefined parameter of the method. The criterion for K-means is the minimising of the distortion between objects and cluster centres. K-means is relatively efficient with its complexity varying linearly with the number of objects.

PAM (Partitioning Around Medoids) [Kaufman and Rouseeuw, 1990] uses an object as the centre of a cluster. The algorithm attempts to swap a centre object with a non-centre object to improve the cost measure. This recursive swapping causes PAM to have a high complexity.

CLARA (Clustering LARge Applications) [Kaufman and Rouseeuw, 1990] extends PAM with a statistical measure to deal with larger data sets. However, the algorithm still has high complexity and local solutions.

CLARANS (CLustering Algorithm based on RANdomised Search) [Ng and Han, 1994] improves CLARA with random search strategies. The algorithm is more efficient and scalable than PAM and CLARA but depends on random factors.

2.3.1.3. Probabilistic Clustering

Cluster representation: A cluster is a probability distribution, often a multi-variable normal distribution. Clusters have unclear borders and can overlap. An object belongs partially to a cluster related to the probability that the cluster can generate the object. An object belongs only to the cluster with the largest probability when applying the clustering result.

Clustering criterion: Clusters are formed by maximising the overall likelihood, which is the probability that the data can be drawn by the set of generators having a normal distribution.

Number of clusters: This is predefined before applying the algorithm and then determined by probabilistic measures, such as MDL (Minimum Description Length) or BIC (Bayesian Information Criterion), based on the clustering result.

Clustering validity and cluster validity: Using artificial data generated by a number of probabilistic generators and setting the number of clusters equal to the number of generators, the cluster validity is determined based on the difference between a cluster and a generator. The clustering validity obtained is the summation of each cluster's validity.

Main methods: The most general method is ***EM*** (Expectation-Maximisation). EM is a two-step iterative optimisation. Step (E) estimates a predefined number of probabilistic expectations for the data. Step (M) approximates a mixture model for the estimated expectations. The model is refined by iterations over the data. Because of the high complexity of EM, there are variations such as ***SNOB*** [Wallace and Dowe, 1994] using the MML principle, ***AUTOCLASS*** [Cheeseman and Stutz, 1996] relying on Bayesian methodology and ***MCLUST*** [Fraley and Raftery, 1999] using a hierarchical mixture model.

The above overview of the three main approaches shows that DC methods are based on different hypotheses, models and objectives. With the same data, a cluster according to one method may not be regarded as a cluster by other methods and vice versa. Finding a common clustering validation strategy for DC methods becomes an impossible task. Thus, each method needs an appropriate clustering validation strategy suitable for its approach.

The selection of a suitable DC method for a practical problem is a difficult process. Based on the characteristics and the expectations of the problem, the aforementioned seven aspects need to be studied carefully step by step to find the most suitable method. Any incorrect step in this investigation can cause ineffective or unexpected DC results.

2.3.2. K-means

K-means clustering (Vector Quantisation) is one of the most popular data clustering methods because of its simplicity and computational efficiency. The computational effort required to form the clusters grows linearly with the data set size. When applied to small or medium size data sets, K-means clustering gives better results than other methods in terms of clustering performance and computational time [Bilmes et al., 1997].

The K-means method is applicable only to data sets with numerical attributes. The Euclidean distance is employed to measure the distance between objects. The main steps in the algorithm are shown in Figure 2.5.

There are a number of different implementations of the K-means method. For example, Linde-Buze-Gray (LBG) is one version of this method in which a batch update mode is applied [Fritzke, 1997]. Other implementations of the method, ISODATA [Kaufman and Rousseeuw, 1990] and MAXNET [Pao, 1989], restrict the

Step 1: Choose K arbitrary objects for K cluster centres.

Step 2: Assign each object in the training set to the closest cluster and update the centres of the clusters.

Step 3: If the clustering criterion is satisfied (the cluster centres do not move), the algorithm stops.

Otherwise, go to Step 2.

Figure 2.5 – The original K-means algorithm.

clusters diameters and introduce flexibility in specifying the number of clusters. Another version of the K-means method [Theiler and Gisler, 1997] employs a contiguity characteristic to improve the algorithm performance in some specific applications.

K-means clustering has been used as a clustering method in many application areas. For example, this method could be employed for:

- Image segmentation and compression [Theiler and Gisler, 1997; Chinrungrueng and Sequin, 1995].
- Grouping image voxels [Gee et al., 1999].
- Initial clustering before applying more sophisticated iterative methods [Hansen and Larsen, 1996].
- Analysing a robot's trajectory [McGovern, 1998].
- Analysing speech and handwriting feature vectors [Cook and Robinson, 1995; Kosmala et al., 1997].
- Grouping machined parts into families in cellular manufacturing system design [Josien and Liao, 2002; Lozano et al., 2002].

2.3.2.1 Improving K-means Performance

Although the K-means method has demonstrated a number of advantages over other DC techniques, it also has drawbacks. In particular, it often converges at a local optimum and, therefore, acceptable results can be found only after several iterations. The local optimum problem has been studied extensively by a number of researchers [MacQueen, 1967; Bottou and Bengio, 1995; Bilmes et al., 1997; Pena et al., 1999].

Another problem with this method is that it requires the number of clusters to be predefined. This especially becomes a very important issue when DC is used as a data exploration technique. In such applications, it is very beneficial for the algorithm to be capable of automatically identifying the number of clusters depending on the distribution of objects in a particular problem space. There have been some attempts to apply the Bayesian Information Criterion (BIC) or Information Gain measure as a pre-defined parameter to help decide the number of clusters but this may lead to incompatibility with the clustering criterion of K-means.

In recent years, many improvements have been proposed and implemented in the K-means method. A number of researchers have proposed different techniques to improve its convergence speed [Al-Daoud, et al., 1995; Bottou and Bengio, 1995; Alsabti, et al., 1998; Pelleg and Moore, 1999; Castro and Yang, 2000 and Kanungo, et al., 2002]. The effect of finite sample size on the K-means method was studied [Bermejo and Cabestany, 2002]. To obtain better results, other researchers [Al-Daoud, et al., 1995; Epter, et al., 1999 and Bradley and Fayyad, 1998] modified the initialisation procedure by presenting the algorithm with data collected using a density-based approach. Again, to improve performance, Fritzke [Fritzke, 1997] suggested a new jumping operation to facilitate the algorithm's convergence and assist it in escaping from local minima. In the same direction with the Fritzke's work, the utility index is used in Patane's and Russo's work [Patane and Russo, 2001]. Chinrungrueng and Sequin [Chinrungrueng and Sequin, 1995] proposed a new updating method introducing a restriction hypothesis about the problem's underlying distribution. The stochastic relaxation scheme was applied to K-means to improve its performance [Kovesi et al., 2001].

The flexibility of K-means is limited by its use of a fixed number of clusters throughout the learning process without considering the characteristics of the data. From the beginning, the number of clusters is selected arbitrarily and clusters are also initialised randomly. These factors may cause inappropriate positioning during the learning process.

2.3.2.2. Scaling up K-means for large data sets

Although the complexity of the K-means algorithm increases linearly with the size of the data, it requires a number of iterations to refine the clustering results. To speed up the access to data, the data is stored in computer memory before applying the algorithm to it. However, the complexity of the problems to be tackled increases with the size of data sets. Therefore, it becomes infeasible to load the complete data set into the memory. This necessitates the data to be stored on media with relatively slow read access, such as disks or tapes. Taking into account the iterative nature of K-means, the processing of large data sets requires new solutions.

One possible solution to this problem is an incremental accumulative strategy for carrying out the clustering task. Such solutions have already been investigated by some researchers. For example, Bradley [Bradley et al., 1998] suggested a scalable framework for clustering. In particular, the clustering model is modified as shown in Figure 2.6. The "scaled" version of the algorithm uses a buffer to load only part of the data into memory and then after each iteration to refine the accumulated knowledge so far. In the proposed strategy, the accumulated knowledge, in the form of statistical

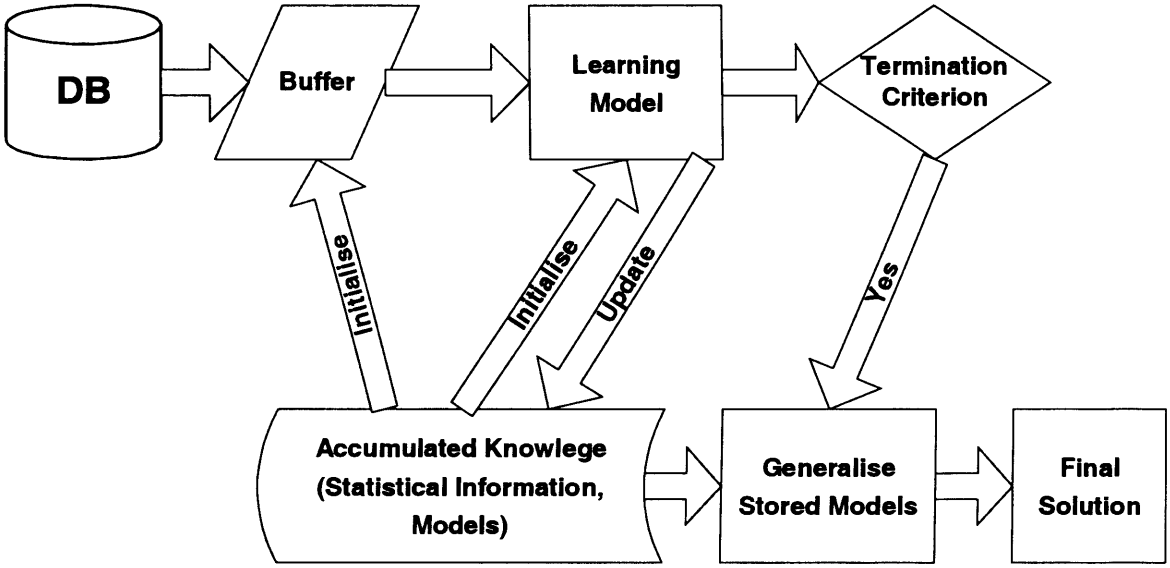


Figure 2.6 - Bradley's scalable framework for clustering [Bradley et al., 1998].

information or models, is used simultaneously to initialise the new learning model and the buffer. The *Termination Criterion* is satisfied when the algorithm processes the whole data set or the user terminates the execution. When the algorithm stops, the accumulated knowledge is generalised to create final solutions. This strategy has already been applied successfully to the K-means algorithm [Bradley et al., 1998; Farnstrom et al., 2000].

To cope with large data sets, K-means should process them in increments and carry out refinements of the formed partitions in a cumulative mode. The algorithm stores the new information in a temporary buffer and then validates and refines the obtained learned knowledge so far. The balance between already learned and new knowledge is a key factor in determining the performance of the scaled version of K-means when comparing it with the original algorithm. This balance depends on the similarity between the distributions of objects in the buffer and the data set. In general, by improving the representation of a data set in the buffer, the performance of the scaled version of the algorithm is also improved and becomes comparable with that of the original algorithm.

The distortion of a cluster is measured by the sum of the squared Euclidean distances between its centre and the objects belonging to it. At the same time, the distortion of a cluster set is the sum of the distortions of its clusters. When very large data sets are processed, each cluster will contain a large number of objects. The analysis of data starts with their normalisation and then the Euclidean distance between a cluster's centre and its objects is limited to a predefined range. Because of this limitation, in case of very large data sets, the distortion of a cluster is more dependent on the

number of objects. Therefore, in such cases, the number of objects belonging to any particular cluster has a higher impact on the distortion of a cluster set and hence on the clustering result.

To achieve a good representation of a data set in the buffer, random sampling is applied. However, there are cases when this technique cannot be applied or is computationally expensive to implement [Bradley et al., 1998]. Thus, in such cases, the buffer will contain only a continuous data section and the distribution of objects in it cannot be considered a true image of the data set. In addition, when large data sets are processed, the distribution of objects in the buffer does not reflect their true distribution in the problem domain, which has an impact on the performance of the algorithm. In the worst case, the buffer may consist of objects belonging to only one expected cluster. The scaled version of K-means should address these balancing problems.

Bradley's scaled version of K-means [Bradley et al., 1998] uses several complex compression schemes to represent the knowledge in the buffer. Three sets of information are employed. The first set called the *discard set* is used to store objects that are unlikely to move to a different cluster and are created by the primary data compression scheme. The second set is called the *compression set* that contains objects created from the accumulated knowledge in the buffer using a secondary clustering scheme. The third set called the *retained set* consists of objects to be kept in the buffer as regular objects. From the objects stored in these three sets, sufficient statistical information could be derived that is representative of the entire data set.

The initialisation of the buffer is very important for Bradley's scaled version of K-means. In the proposed approach, when random sampling is not available to seed the compression schemes in the initial stages of the clustering process, it is suggested that "true random samples" be obtained by some other means. However, finding a *true* random sample in a very large data set is also a high complexity task. In general, the initialisation of a new model for the K-means algorithm and the balancing issues concerning the selection of objects to be stored in the buffer are not addressed adequately by Bradley and are still open problems.

Because of the dependence on the initialisation of the K-means algorithm, Bradley's scaled version produces multi-solutions instead of generalising the accumulated knowledge to produce a single solution. By considering a cluster as an object weighted by its number of objects, Farnstrom et al. [2000] proposed to generalise the accumulated knowledge into the final cluster set.

At the same time, it should be remembered that the application of compression schemes increases the algorithm's complexity. Farnstrom et al. [2000] showed that Bradley's scaled version is slower and performs worse than the original K-means algorithm on the data sets used for benchmarking. However, there are other factors such as the selection of parameters for the compression schemes, the buffer size and the data accessing speed that could affect the algorithm performance.

Farnstrom et al. [2000] suggested a simple scaled version of K-means based only on the retained set of objects called the *discard set*. Each object in the retained set is a cluster created in one of the previous iterations and treated as a regular object

weighted by the number of objects belonging to it. Thus, the K-means algorithm is applied to the buffer, which consists of the retained set and new objects, and the result is used to update the retained set for the next iteration. For the first iteration, the K-means algorithm is initialised randomly. Then, in subsequent iterations, the cluster centres from the last iteration are used to initialise the K-means algorithm. The weight of regular objects in the buffer is used to represent the accumulated knowledge. Managing the relationship between the accumulated and new knowledge is an open issue for this scheme, especially when the set of objects stored in the buffer are not representative for the data set as a whole.

2.3.2.3. Research on the K-means method in this study

Chapter 5 introduces two new versions of K-means to address its deficiencies. The Incremental K-means algorithm can induce the optimal clustering result with flexible management of the number of clusters during the learning process. The Two-Phase K-means algorithm uses a buffer to scale K-means up to very large data sets. The flexible management of the number of clusters during each phase is also used in this algorithm.

Chapter 6 reviews current selection methods for the number of clusters for K-means in several research papers. Then, factors which affect this selection are studied. A new method for choosing the number of clusters for K-means is introduced. The evaluation of the proposed method is based on the comparison between the data distribution and the standard uniform distribution. Users can flexibly select a suitable

number according to the result of the comparison. The new method is analysed on several data sets by inspection.

2.4. Summary

This chapter reviews the main aspects of ML and DM. The DM process is described and its stages are discussed briefly. Two principal branches of ML and DM, Induction Learning and Data Clustering, are reviewed along with a number of algorithms for each branch. Research directions for these two branches are also described.

Chapter 3

Rule Extraction System with Adaptivity (RULES-A)

3.1. Preliminaries

This chapter introduces a new covering algorithm, Rule Extraction System with Adaptivity (RULES-A). RULES-A follows the conquer-without-separation approach. The algorithm forms the whole rule set simultaneously instead of a single rule at a time. RULES-A has the ability to process continuous attributes directly so that the data does not need to be pre-processed.

The size of the rule set in RULES-A is *flexibly* managed. Rules can be added to or omitted from the rule set during the induction phase. The learning strategy of the algorithm is object-oriented. When a new object is presented, depending on the classified state of the object, rules in the rule set can be updated. Rules are checked for their consistency based on their overlapping covering area instead of an evaluation on the training set.

Pruning in RULES-A is split into two phases. First, small rules that cover few objects are pruned from the rule set to improve its quality. Then, rules are simplified to make them more general.

3.2. Algorithm Description

RULES-A is a conquer-without-separation algorithm and, like all algorithms implementing this inductive learning strategy, it forms the whole rule set simultaneously, rather than each rule separately. The evaluation measure used to guide the rule formation is based on an assessment carried out on the whole rule set. Another distinguishing feature of RULES-A is that this algorithm processes continuous attributes directly, so that a discretisation step is not required.

The rule format in RULES-A is similar to that proposed by Domingos [Domingos, 1996c]. In particular, in the antecedent part of the rules, there are conditions for each attribute. These conditions could be attribute-value pairs, ranges of values for continuous attributes or conditions that are always satisfied. These always-satisfied conditions are used during the rule forming process and then deleted before the rule set is ready for use.

The RULES-A algorithm is described in Figures 3.1 to 3.4. The rule set forming process is carried out in three phases (Figure 3.1). In the first phase (Figure 3.2), RULES-A forms a rule set that covers the training data with a minimum number of rules. Each rule formed at this stage contains all attributes in its conditional part. In phase 2 (Figure 3.3), the rule set formed so far is pruned to remove very specific rules (rules with minimum coverage). Then, in phase 3 (Figure 3.4), the rule set is simplified to create the final set of rules.

Phase 1 – Induction: Formation of a rule set that covers all training examples with minimum number of rules.
Phase 2 – Pruning: Removal of the most specific rules from the rule set.
Phase 3 – Rule simplification.

Figure 3.1 – The three phases of Rule Extraction System with Adaptivity (RULES-A).

Step 0. Initialisation of the rule set (Empty rule set).
Step 1. Initialisation of the training set.
Step 2. One epoch:
- 2.1. If the training set is empty, go to step 3.
- 2.2. Remove randomly one object X from the training set.
- 2.3. If X is misclassified by the rule set,
  divide the misclassifying rule into 2 new rules and compute their estimated capacities and coverage.
  Add a new rule which can cover X, go to step 2.1.
- 2.4. Find a rule R for the same class as X that covers or can be expanded to cover X. If there is more than one rule, the rule with the highest evaluation measure is selected.
  If X is covered by an existing rule R, update its capacity and coverage.
  If R can be expanded in order to cover X, R is modified and its capacity updated.
  Go to step 2.1.
- 2.5. If X cannot be covered by an existing or expanded rule, create a new rule and add it to the rule set. Go to step 2.1.

Step 3. Rule set compacting:
- 3.1. Update the coverage and capacity of all rules not modified in this epoch.
- 3.2. Remove rules in the rule set that have a capacity equal to 0.
- 3.3. If the rule set has undergone any changes in this epoch, go to step 1.

Epoch: one iteration over the training set.
Rule capacity: the number of objects in the training set covered by a rule.
Rule coverage: the area in the object space covered by the rule.

Figure 3.2 - Phase 1 – Induction.

Input: The resultant rule set from phase 1, the minimum and the maximum values of the pruning threshold and the pruning set.

Step 1: Initialise the pruning process with a threshold equal to the specified minimum value.

Check the initial rule set on the pruning set. This initial rule set and its accuracy are stored.

Step 2: Create a new rule set from the initial rule set that contains only rules with a capacity above the current pruning threshold.

Check the new rule set on the pruning set; if the new rule set has a higher accuracy than the accuracy of the stored rule set, replace that with the new rule set.

Step 3: Increase the threshold.

If the new threshold is smaller than the specified maximum value

Then Go to Step 2

Else output the stored rule set and stop the pruning process.

Figure 3.3 – Phase 2 – Pruning.

Input: the rule set created after phase 2.

Create a new rule set that is the same as the input rule set.

For each rule in the new rule set:

For each rule condition:

Check that omitting this condition from the rule does not cause overlapping rules for different classes in the input rule set.

If the rule is satisfied then remove this condition.

Output: The simplified rule set.

Figure 3.4 – Phase 3 – Rule simplification.

The algorithm starts with an empty rule set. When a new object is presented to the algorithm, if it is the first object, a new rule is created with all of the attribute-value pairs of the object as conditions. The rule is added to the rule set. Otherwise, the following procedure is carried out. First, the new object is checked to see if it is inside the hyper-rectangle of a rule for a class different from that of the object (Figure 3.5.a). If this is the case, that rule will be split into two new rules in order to avoid misclassifying the object. In this chapter, only the splitting of rules with continuous attributes is discussed. Handling discrete attributes will be the subject of chapter 4.

To split a rule, a condition is selected corresponding to a continuous attribute with the widest value range (Figure 3.5.b). This was found empirically to give the best result for the data sets tested. The distance ε is a predefined constant. Initially, the number of objects covered by each of these two newly created rules is only an estimate. In particular, the total number of objects covered by the initial rule is divided between the two new rules proportionally to their covering areas. At the end of each epoch (one iteration over the training set), this estimate is replaced by the *real* number of objects covered by each rule in the rule set. Also, to cover the new object, a rule is created and added to the rule set.

If the new object is not misclassified by the current rule set, all rules for the same class as that of the new object are checked to see whether they cover the object or can be expanded to cover it. A rule is considered expandable if its consistency can be maintained during the expansion operation. If there is more than one rule that can cover the new object, the rule with the highest evaluation measure will be expanded.

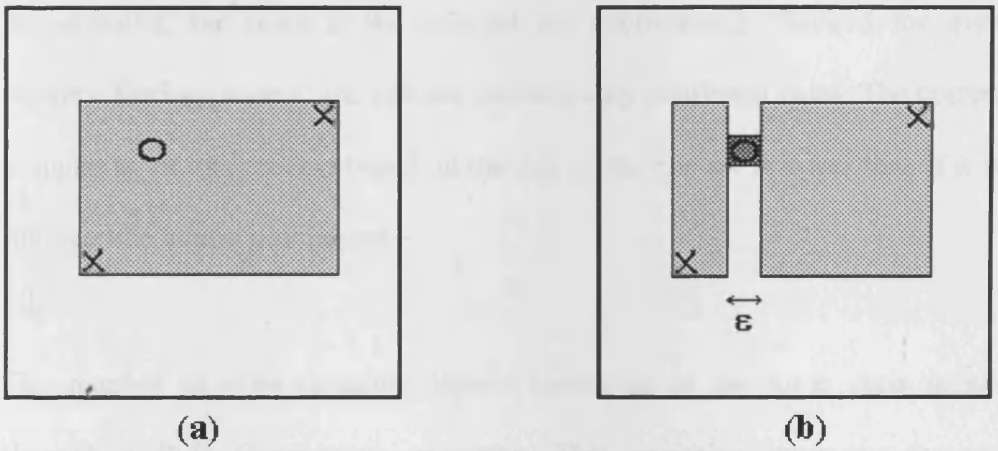


Figure 3.5 – The splitting operation.

X, O: objects of classes X and O, respectively

ε: a predefined parameter

▨ : hyper-rectangle of a rule

If no rule can be expanded, a new rule will be created to cover the object and added to the rule set.

The consistency of the rule set is verified constantly during the rule forming process. In particular, the rules in the rule set are continuously checked for *overlapping regions*. During phase 1, the rule set contains only consistent rules. The computational complexity of this process based on the size of the rule set is lower than if it is carried out over the whole training set.

The number of rules covering objects belonging to the same class is minimised through a *Rule Compacting* operation. This operation improves the evaluation measure of the rules. This measure evaluates the number of objects that each rule covers. When an object is covered by more than one rule, the rule with the highest evaluation measure is selected to classify it. Throughout the iterative rule forming process, the objects covered by rules with a lower evaluation measure are attracted by other rules with a higher evaluation measure. During each epoch, the area of the object space covered by each rule is made more compact. In this way, some rules could become fully covered by other rules in the rule set and thus become redundant.

RULES-A refines the rule set over a number of iterations over the training set. At the end of each epoch, each rule in the rule set will be resized to the smallest hyper-rectangle that covers all of its objects. Rules that do not cover any object will be removed. If the rule set is not modified after an epoch, phase 1 of RULES-A is considered complete.

The resultant rule set from phase 1 is further processed in phase 2 to make it more accurate and general. Before starting the rule forming process, the initial training set is split into a training set and a pruning set with a ratio of 70 to 30. In phase 2 of RULES-A, rules that cover a smaller number of objects than a specified pruning threshold will be removed form the rule set if the accuracy of the resultant rule set is higher. During the pruning process, the initial threshold increases until the specified maximum value is reached.

The rules formed after phases 1 and 2 contain conditions for all attributes. In phase 3, these rules are further processed to make them simpler. This is carried out by reducing the number of conditions while maintaining the consistency of the rule set. The number of rules in the rule set remains unchanged during this operation.

Figure 3.6 illustrates the step by step execution of RULES-A. The distribution of objects in the training set is shown in Figure 3.6.a. Because the objects are selected randomly for processing, Figure 3.6.b shows one of the possible intermediate states. The new object to be processed belongs to class X and there is only one rule formed so far for this class. This existing rule can be expanded to cover this new object while its consistency is maintained. The next object, which belongs to class O, is inside the rule for class X, and hence it is misclassified. Therefore, this inconsistent rule is split as shown in Figure 3.6.c. The rule forming operation continues until all objects are processed. The rule set generated after phase 1 of RULES-A is shown in Figure 3.6.d. Figure 3.6.e depicts the result of rule set pruning after phase 2. Finally, the resulting rule set after phase 3 of RULES-A is given in Figure 3.6.f.

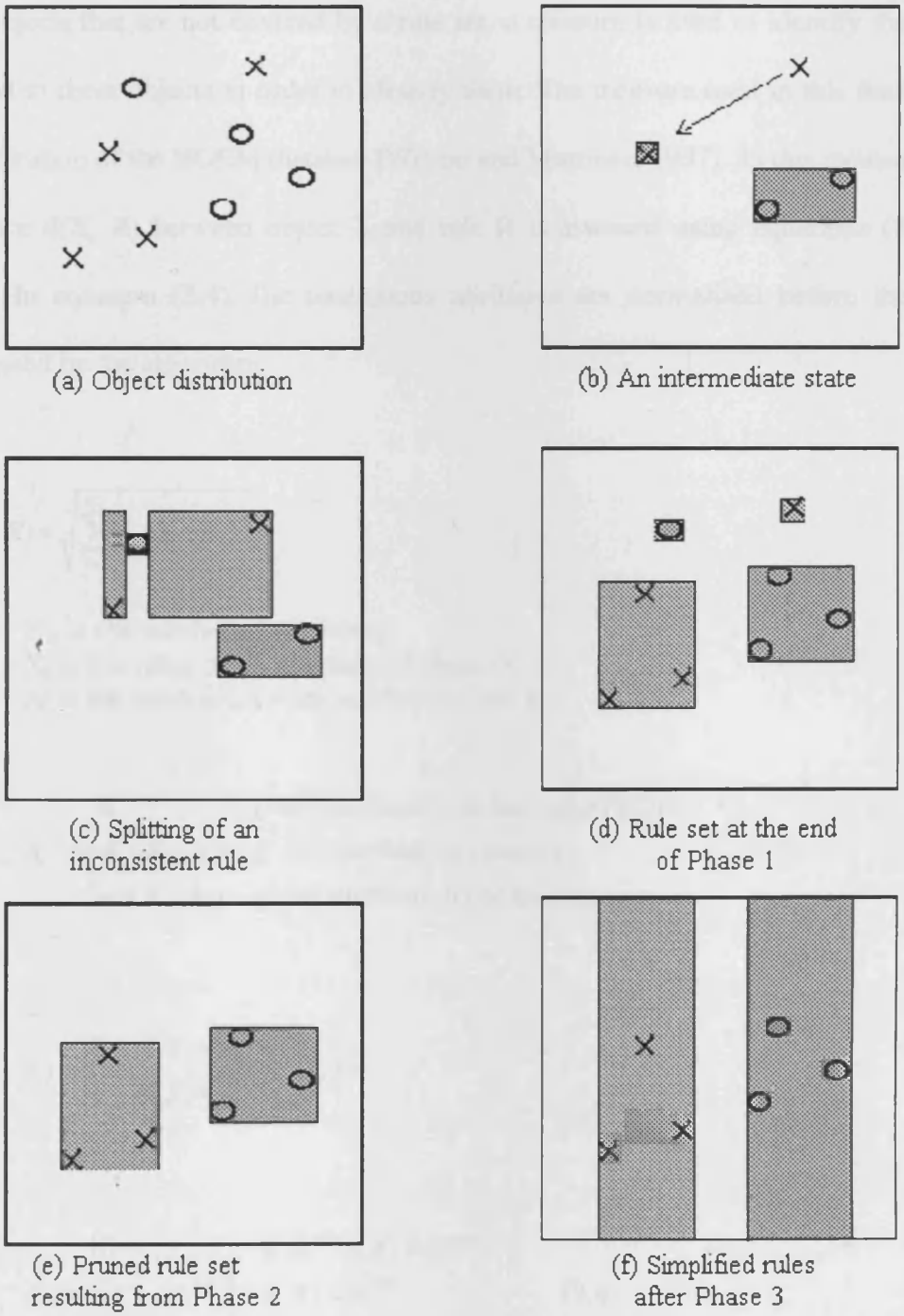


Figure 3.6 – Illustrative example of the execution of ARI.

For objects that are not covered by a rule set, a measure is used to identify the rules closest to these objects in order to classify them. The measure used in this thesis is a modification of the HOEM distance [Wilson and Martinez, 1997]. In this measure, the distance d(X, R) between object X and rule R is assessed using equations (3.1) to (3.4). In equation (3.4), the continuous attributes are normalised before they are processed by the algorithm.

$$d(X,R)=\sqrt{\sum_{i=1}^{N_A} d_1(X_i,A_i)^2} \tag{3.1}$$

where N_A is the number of attributes,
X_i is the value of *ith* attribute of object X
A_i is the condition for *ith* attribute in rule R

$$d_1(X_i,A_i)=\begin{cases} 0 & \textit{if the antecedent } A_i \textit{ has value TRUE} \\ d_2(X_i,A_i) & \textit{if ith attribute is symbolic} \\ d_3(X_i,A_i) & \textit{if ith attribute is continuous} \end{cases} \tag{3.2}$$

$$d_2(X_i,A_i)=\begin{cases} 0 & \textit{if } X_i=A_i \\ 1 & \textit{if } X_i\neq A_i \end{cases} \tag{3.3}$$

$$d_3(X_i,A_i)=\begin{cases} 0 & \textit{if } A_i^{\min}\le X_i\le A_i^{\max} \\ \left(X_i-A_i^{\min}\right) & \textit{if } X_i<A_i^{\min} \\ \left(X_i-A_i^{\max}\right) & \textit{if } X_i>A_i^{\max} \end{cases} \tag{3.4}$$

where A_i^{min} and A_i^{max} are the lower and upper limits of a condition for the *ith* attribute in rule R.

If a is the number of attributes, k is the number of rules, e is the number of objects in the training set, p is the number of objects in the pruning set, r is the number of iterations over the training set and t is the pruning threshold, the complexities of phases 1, 2 and 3 are $O(aek^2r)$, $O(apkt)$ and $O(a^2k^2)$ respectively (see Appendix A). The three phases are run consecutively, so that the complexity of the RULES-A algorithm is considered to be the largest of these three, $O(aek^2r)$. The number of iterations over any particular training set required to form a rule set depends on the specific characteristics of the data set and could be obtained by carrying out tests.

3.3. Performance

RULES-A was tested on 9 data sets from the UCI repository [Blake et al., 1998]. Brief descriptions of the selected data sets are given in the Appendix B. All selected data sets have continuous attributes. Ten-fold cross-validation checking [Michie et al., 1994] was employed to test the algorithm on each data set. The obtained results are compared with the result of the C5 algorithm [ISL, 1998]. C5 was chosen to benchmark the new algorithm as C5 is the best rule induction algorithm available to the author.

The performance of each phase of RULES-A is discussed separately. The results obtained after phase 1 are compared with C5 to assess the predictive capabilities of RULES-A. The results after phases 1 and 2 are then compared to show the effect of

the pruning process. The effectiveness of the simplification procedure of phase 3 is introduced and the overall performance of RULES-A and C5 is compared.

3.3.1. Phase 1 Performance

It can be seen from Table 3.1 that the rule sets created using the RULES-A algorithm have a higher average accuracy when compared with the C5 results. In 7 of the selected 9 data sets, the accuracy of the rule sets generated using RULES-A is higher. However, it should be noted that RULES-A is less stable than C5 due to a higher standard deviation in the classification performance between different executions. In addition, the number of rules generated after Phase 1 is about 2.5 times higher than those produced by C5.

Table 3.1 – Result of cross-validation 10-fold testing of RULES-A compared with C5.

Data Set	C5			RULES-A (Phase 1 only)		
	Accuracy	Standard Deviation	Number of Rules	Accuracy	Standard Deviation	Number of Rules
Australian	83.50	1.30	22	**86.23**	3.32	50
Balance Scale	**77.90**	1.50	29	73.41	3.50	91
Glass 2	78.60	2.70	8	**82.99**	7.04	18
Heart	78.10	2.70	9	**81.11**	8.19	26
Ionosphere	88.30	1.60	10	**91.72**	5.50	15
Iris	94.00	2.10	5	**96.00**	2.00	8
Pima Indian	**76.30**	1.80	16	74.17	4.67	93
Wine	93.20	1.70	6	**96.66**	3.83	8
Zoo	93.10	2.60	14	**94.00**	8.00	9
Average Accuracy	84.77			**86.25**		
Total Number of Rules			119			318

3.3.2. Results after Phases 1 and 2

Phase 2 of the RULES-A algorithm was applied to the results obtained after Phase 1. In Table 3.2, the performances of the rule sets before and after Phase 2 for 5 data sets are compared.

The number of objects in the pruning set is an important factor that influences the efficiency of Phase 2. The more objects the pruning set contains, the more accurate the evaluation is. Taking into account the size of the data sets and the accuracy achieved after Phase 1 of RULES-A, it was decided to set the minimum size of the pruning set to be 50. Table 3.2 only includes data sets that satisfy this condition.

As expected, after the pruning, the performance of the rule set on the training sets is worse. The accuracy of the rule sets decreases monotonically as the pruning threshold increases. This is important in order to identify the upper limit of the accuracy on the test sets when pruning is applied. On most data sets, apart from Balance-Scale, changes in rule sets accuracy on the test sets are not significant. The main effect of the pruning is the reduction in the number of rules in the rule sets.

On the Balance-Scale data set, the pruning process leads to a significant reduction in the number of rules, from 91 to 22, and at the same time, the rule accuracy increases from 73.41% to 82.56%. This could be explained partially by the decrease in the number of misclassifed objects and the simultaneous increase in the number of unclassified objects. Such changes in the rule set performance indicate that the data set contains noise. The small difference between the accuracy on the training set and the test set shows that the applied pruning technique has reached its limits.

Table 3.2 – The performance of the rule sets before and after Phase 2.

Data Set		RULES-A (Phase 1 only)	RULES-A (Phase 1 + 2)
Australian	Accuracy On Training Set	100.00	96.85
	Accuracy On Test Set	86.23	86.09
	Standard Deviation	3.32	3.19
	Percentage of Misclassified Objects	5.22	5.22
	Percentage of Unclassified Objects	34.20	34.20
	Number Of Rules	50	37
Balance Scale	Accuracy On Training Set	100.00	87.38
	Accuracy On Test Set	73.41	82.56
	Standard Deviation	3.50	4.38
	Percentage of Misclassified Objects	12.81	8.77
	Percentage of Unclassified Objects	18.25	25.84
	Number Of Rules	91	22
Heart	Accuracy On Training Set	100.00	98.54
	Accuracy On Test Set	81.11	81.48
	Standard Deviation	8.19	8.45
	Percentage of Misclassified Objects	7.04	7.04
	Percentage of Unclassified Objects	53.70	53.70
	Number Of Rules	26	24
Ionosphere	Accuracy On Training Set	100.00	100.00
	Accuracy On Test Set	91.72	91.72
	Standard Deviation	5.50	5.50
	Percentage of Misclassified Objects	1.43	1.43
	Percentage of Unclassified Objects	35.35	35.35
	Number Of Rules	15	15
Pima Indian	Accuracy On Training Set	100.00	91.34
	Accuracy On Test Set	74.17	74.57
	Standard Deviation	4.67	5.41
	Percentage of Misclassified Objects	9.10	8.97
	Percentage of Unclassified Objects	48.95	49.08
	Number Of Rules	93	53
	Average Accuracy On Test Set	81.32	83.28
	Total Number of Rules	275	151

In addition, it was found that the pruning process could slightly affect the stability of the rule set. This is indicated by an increase in the standard deviation in accuracy. This is more than compensated, though, by a significant decrease in the number of rules in the rule sets, by 45.10%.

3.3.3. Results after Rule Simplification (Phase 3)

The efficiency of Phase 3 is assessed by comparing the performance of the rules before and after the rule simplification process (Table 3.3).

This simplification operation enlarges the coverage area of the rules, which increases the probability for classifying or misclassifying previously unseen objects. The results from the tests carried out on the selected data sets are given in Table 3.3. They show that the percentage of misclassified objects increases and the percentage of unclassified objects decreases while the accuracy and deviation only change slightly. This means that the resulting rule sets are simpler but not more general [Domingos, 1998].

Examples of rule sets generated using RULES-A are given in Figure 3.7. Each illustrative rule set is formed by a training set having 90% of the objects in the data set. Because of the random selection of objects from the data set to form the training set, the size of the induced rule set may be different from that stated in Table 3.1. The rules are relatively simple and comprehensible to the user and employ different ranges for conditions.

Table 3.3 – Comparison of performance of rule set before and after Phase 3.

Data Set		RULES-A (Phases 1 + 2)	RULES-A (Phases 1,2 + 3)
Australian	Accuracy On Training Set	96.85	96.97
	Accuracy On Test Set	86.09	86.67
	Standard Deviation	3.19	2.96
	Percentage of Misclassified Objects	5.22	6.52
	Percentage of Unclassified Objects	34.20	24.64
	Number Of Rules	37	37
Balance Scale	Accuracy On Training Set	87.38	87.41
	Accuracy On Test Set	82.56	82.40
	Standard Deviation	4.38	4.42
	Percentage of Misclassified Objects	8.77	8.94
	Percentage of Unclassified Objects	25.84	24.88
	Number Of Rules	22	22
Glass2	Accuracy On Training Set	100.00	100.00
	Accuracy On Test Set	82.99	81.55
	Standard Deviation	7.04	8.43
	Percentage of Misclassified Objects	4.38	6.25
	Percentage of Unclassified Objects	51.22	33.19
	Number Of Rules	18	18
Heart	Accuracy On Training Set	98.54	98.54
	Accuracy On Test Set	81.48	80.74
	Standard Deviation	8.45	7.73
	Percentage of Misclassified Objects	7.04	11.48
	Percentage of Unclassified Objects	53.70	31.48
	Number Of Rules	24	24
Ionosphere	Accuracy On Training Set	100.00	100.00
	Accuracy On Test Set	91.72	90.58
	Standard Deviation	5.50	4.63
	Percentage of Misclassified Objects	1.43	4.57
	Percentage of Unclassified Objects	35.35	15.68
	Number Of Rules	15	15

Table 3.3 (continuation)

		RULES-A (Phases 1 + 2)	RULES-A (Phases 1,2 + 3)
Iris	Accuracy On Training Set	100.00	100.00
	Accuracy On Test Set	96.00	96.00
	Standard Deviation	2.00	2.67
	Percentage of Misclassified Objects	5.33	5.33
	Percentage of Unclassified Objects	15.33	5.33
	Number Of Rules	8	8
Pima Indian	Accuracy On Training Set	91.34	91.36
	Accuracy On Test Set	74.57	74.17
	Standard Deviation	5.41	5.52
	Percentage of Misclassified Objects	8.97	10.66
	Percentage of Unclassified Objects	49.08	42.04
	Number Of Rules	53	53
Wine	Accuracy On Training Set	100.00	100.00
	Accuracy On Test Set	96.66	96.07
	Standard Deviation	3.83	5.26
	Percentage of Misclassified Objects	0.40	2.16
	Percentage of Unclassified Objects	49.13	21.84
	Number Of Rules	8	8
Zoo	Accuracy On Training Set	100.00	100.00
	Accuracy On Test Set	94.00	94.00
	Standard Deviation	8.00	6.63
	Percentage of Misclassified Objects	1.00	4.00
	Percentage of Unclassified Objects	17.00	2.00
	Number Of Rules	9	9
Average Accuracy On Test Set		87.34	86.90
Total Number of Rules		194	194

One of the induced rule sets for the Ionosphere data set:

(0.31 ≤ Att2) & (0.16 ≤ Att4) & (-0.18 ≤ Att7) & (-0.11 ≤ Att11) & (-0.27 ≤ Att13) & (-0.89 ≤ Att22) & (-0.87 ≤ Att29) ⇒ g

(Att0 = 1) & (0.48 ≤ Att2) & (0.43 ≤ Att4) & (0.65 ≤ Att6) & (Att21 ≤ 0) & (-0.92 ≤ Att31) ⇒ g

(0.50 ≤ Att2) & (-0.03 ≤ Att3) & (0.39 ≤ Att4) ⇒ g

(0.65 ≤ Att2) & (0.58 ≤ Att4) ⇒ g

(Att2 ≤ 0) ⇒ b

(Att4 ≤ 0) ⇒ b

(Att0 = 0) ⇒ b

(Att7 = -1) & (0.63 ≤ Att21) ⇒ b

(-0.94 ≤ Att3 ≤ 0) & (-0.04 ≤ Att6) & (Att11 ≤ -0) ⇒ b

(Att26 = 1) & (0.87 ≤ Att31) ⇒ b

(-0.1 ≤ Att13 ≤ 0.82) ⇒ b

(Att28 = -1) & (Att29 = -1) ⇒ b

(0.57 ≤ Att6) & (Att22 = 1) ⇒ b

One of the induced rule sets for the Iris data set:

(Att3 ≤ 0.50) ⇒ Iris-setosa

(5.00 ≤ Att0 ≤ 7.00) & (1.00 ≤ Att3 ≤ 1.70) ⇒ Iris-versicolor

(Att1 = 3.20) & (Att2 = 4.80) ⇒ Iris-versicolor

(4.90 ≤ Att2) & (1.80 ≤ Att3) ⇒ Iris-virginica

(Att0 = 4.90) & (Att3 = 1.70) ⇒ Iris-virginica

(Att2 = 5.80) ⇒ Iris-virginica

(Att1 = 3.00) & (Att3 = 1.80) ⇒ Iris-virginica

One of the induced rule sets for the Wine data set:

(3.52 ≤ Att9 ≤ 9) & (2.51 ≤ Att11) & (795.00 ≤ Att12) ⇒ 1

(1.25 ≤ Att6 ≤ 5) & (2.01 ≤ Att11) & (345.00 ≤ Att12 ≤ 718) ⇒ 2

(1.95 ≤ Att9 ≤ 3) ⇒ 2

(0.47 ≤ Att6 ≤ 1) & (3.85 ≤ Att9 ≤ 11) ⇒ 3

(1.27 ≤ Att11 ≤ 1.33) ⇒ 3

One of the induced rule sets for the Zoo data set:

(Att3 = 1) ⇒ 1

(Att9 = 0) & (Att13 = 1) ⇒ 4

(Att3 = 0) & (Att12 = 2) & (Att13 = 1) ⇒ 2

(Att9 = 0) & (Att13 = 0) ⇒ 7

(Att8 = 0) & (Att12 = 0) ⇒ 7

(Att9 = 1) & (Att12 = 6) ⇒ 6

(Att3 = 0) & (Att12 = 4) & (Att13 = 1) ⇒ 5

(Att3 = 0) & (Att11 = 0) & (Att12 = 0) & (Att13 = 1) ⇒ 3

Figure 3.7 – Illustrative induced rule sets.

3.3.4. Comparison of the Overall Performance of RULES-A with C5 and RULES 3+

The overall performance of the RULES-A algorithm is compared with the results of C5 (Table 3.4) and RULES 3+, an earlier member in the RULES family (Table 3.5).

The rule sets obtained after the three phases of the RULES-A algorithm perform better than C5 on eight of the nine tested data sets and their average accuracy is higher. The total number of rules generated by RULES-A, however, is significantly (about 1.63 times) higher than the number generated by C5.

In comparison with the results obtained using RULES 3+ (Table 3.5), the accuracy of RULES-A is higher on five of the nine tested data sets. The average accuracy of RULES-A is also higher than that of RULES 3+. For the remaining 4 data sets, apart from the Heart data set, the accuracy of RULES 3+ is slightly better but the rule sets are less stable than for RULES-A, as can be seen in their high standard deviations. The total number of rules generated by RULES 3+ is almost five times higher than the number generated by RULES-A.

Table 3.4 – Comparison of C5 and RULES-A results.

Data Set	C5			RULES-A		
	Accuracy	**Standard Deviation**	**Num. Of Rules**	**Accuracy**	**Deviation**	**Num. Of Rules**
Australian	83.50	1.30	22	**86.67**	2.96	37
Balance Scale	77.90	1.50	29	**82.40**	4.42	22
Glass 2	78.60	2.70	8	**81.55**	8.43	18
Heart	78.10	2.70	9	**80.74**	7.73	24
Ionosphere	88.30	1.60	10	**90.58**	4.63	15
Iris	94.00	2.10	5	**96.00**	2.67	8
Pima Indian	**76.30**	1.80	16	74.17	5.52	53
Wine	93.20	1.70	6	**96.07**	5.26	8
Zoo	93.10	2.60	14	**94.00**	6.63	9
Average / Sum of Rules	84.77	**2.00**	**119**	**86.90**	5.36	194

Table 3.5 – Comparison of RULES 3+ and RULES-A results.

Data Set	RULE 3+			RULES-A		
	Accuracy	**Standard Deviation**	**Num. Of Rules**	**Accuracy**	**Deviation**	**Num. Of Rules**
Australian	82.93	2.78	181	**86.67**	2.96	37
Balance Scale	80.03	3.39	271	**82.40**	4.42	22
Glass 2	71.19	10.68	51	**81.55**	8.43	18
Heart	**83.33**	6.87	78	80.74	7.73	24
Ionosphere	90.03	6.07	54	**90.58**	4.63	15
Iris	**96.67**	4.85	14	96.00	2.67	8
Pima Indian	66.96	7.07	259	**74.17**	5.52	53
Wine	**96.66**	5.84	29	96.07	5.26	8
Zoo	**94.09**	7.02	12	94.00	6.63	9
Average Accuracy / Sum of Rules	84.65	6.06	949	**86.90**	**5.36**	**194**

3.3.5. Algorithm Complexity

The number of iterations required over the training sets during phase 1 of the RULES-A algorithm is given in Table 3.6.

C5 has a complexity similar to that of C4.5. The latter has been estimated as $O(ae^2 \log e)$ [Paliouras and Bree, 1995]. The complexities of C5 and RULES-A are compared in Figure 3.8. On the smaller data sets such as Iris, Wine or Zoo, the complexities of the two algorithms are approximately the same but on large data sets, RULES-A is less efficient.

Table 3.6 – The number of iterations over the training sets during Phase 1 of RULES-A.

Data set	The average number of data set iterations
Australian	37.5
Balance-Scale	19.2
Glass 2	19.0
Heart	24.1
Ionosphere	18.2
Iris	8.2
Pima	50.4
Wine	6.8
Zoo	5.0

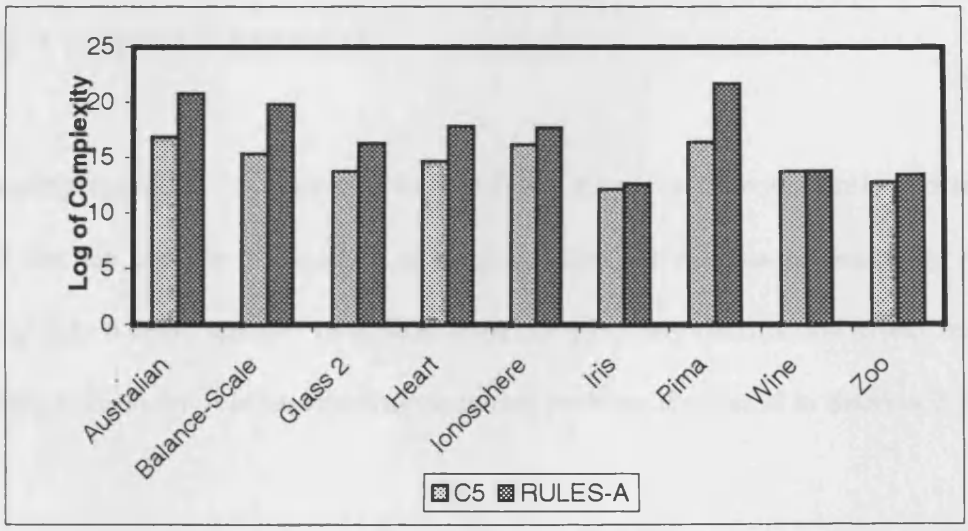


Figure 3.8 – Comparison between the complexities of C5 and RULES-A.

3.4. Summary

The RULES-A algorithm generates rules from observations by applying the conquer-without-separation approach. The adaptive capability of RULES-A allows the relationships between the rules in the rule set to be taken into account during the induction process. This permits the overlapping areas between rules of the same class to be avoided, leading to the generation of more compact rule sets. The relationship between rules of different classes is maintained more efficiently, benefiting from internal information during induction.

In addition, RULES-A processes continuous attributes directly without applying any discretisation techniques. This improves the inductive learning capabilities of RULES-A compared to RULES 3+.

The pruning operation implemented in RULES-A allows more robust rule sets to be created that are capable of handling noisy data. Also, the process of removing rules covering only a small number of objects does not have any detrimental effect on the remaining rules in the rule set (the fragmentation problem discussed in Section 2.1.1).

The performance of the rule sets generated using RULES-A proves the potential of this new algorithm. Compared with C5, considered to be the best decision tree induction algorithm, the rule sets obtained using RULES-A outperform those generated by C5. In comparison with RULES 3+, the RULES-A rule sets are significantly smaller and more accurate on five of the nine benchmark data sets.

Further work is required in order to make possible the simultaneous processing of continuous and discrete attributes. Indexing techniques should also be introduced to reduce the complexity of the algorithm.

Chapter 4

Improvements to RULES-A

4.1. Preliminaries

Chapter 3 introduced RULES-A, a covering algorithm that applies the conquer-without-separation approach. In this chapter, improvements to RULES-A are discussed and their implementation in a new version of the RULES-A algorithm, RULES-A1, are described. This new algorithm can process discrete attributes directly and the pruning is carried out automatically. To facilitate the processing of discrete attributes by RULES-A1, modifications are introduced to the representation scheme of the rules. For problems with both continuous and discrete attributes, a heuristic procedure is proposed for selecting attributes that could be used to split rules. Rule pruning and continuous learning after pruning are embedded in the learning process. To verify the processing capabilities of RULES-A1, its performance is compared with that of C5 and RULES 3+ on several benchmark data sets.

The chapter also discusses further improvements of RULES-A1 that were implemented in RULES-A2. An early stopping strategy is suggested to halt learning after pruning. Also, learning from data sets with varying object orders is applied to find potentially better rule sets. The performance of RULES-A2 is compared with that of RULES-A1, C5 and RULES 3+ on several benchmark data sets.

Finally, the dependencies between the characteristics of the test data sets and the different learning methods are studied using the Tic-Tac-Toe data set. Such an analysis would help to improve a learning algorithm and apply it more effectively.

4.2. Improvements

4.2.1. Discrete Attributes

Compared with continuous attributes, discrete attributes have fewer characteristics. Their values are often limited to a specific set. Discrete attributes with many values, for examples names or transaction identifiers, are often eliminated during the pre-processing stage and are not utilised by learning algorithms. When two discrete values are compared, there are only two possible outcomes, "1" or "0" for identical or different values, respectively. To process discrete attributes, a number of changes are introduced to RULES-A.

Discrete attributes are unordered. Thus ranges cannot be used in the antecedent part of conditions formed for these attributes. A rule which covers 2 objects with different values for a particular discrete attribute will not include a condition for such an attribute. In this case, in phase 1 of RULES-A, the generated rule will include an "always-satisfied" condition for this attribute that will be omitted automatically in phase 3.

When a discrete attribute is used to split a rule, the attribute values for the objects processed up to that point should be known in order to initialise the rules that will replace it. In particular, the occurrence frequency of different values of each discrete attribute has to be stored and used later on during the splitting operation for rules initialisation.

Another issue during this operation is how to select a discrete attribute used to split a rule. If for the selected discrete attribute, the objects covered by the rule have more than 2 values, the splitting will lead to the creation of more than 2 rules. To minimise the negative effect of this operation on the generality and the efficiency of the generated rule set, a heuristic is used to select a discrete attribute of which the 2 most frequent values cover the largest percentage of objects.

The complexity of this task increases when the data set contains both continuous and discrete attributes. When splitting a continuous attribute, only two rules are always created. If a discrete attribute is split, more than 2 rules may be created. Therefore, the selection of a continuous attribute is highly preferred by the algorithm. To avoid continuous attributes always being selected to carry out the splitting operation, the following heuristics will be applied in RULES-A in the order of priority shown:

(1) Select a discrete attribute, of which the corresponding condition in the rule is "always satisfied" and of which the value for the misclassified object is different from the two values seen up to that point for this attribute. The two rules formed in this way will cover all objects of the split rule.

(2) Select a continuous attribute with the widest value range (as mentioned in chapter 3).

(3) Select a discrete attribute, of which the corresponding condition in the rule is "always satisfied" and for which the ratio between the occurrences of the two most frequent values and the total number of objects covered is the highest among all attributes. This will lead to the creation of two replacement rules that will not cover all objects of the original rule. Uncovered objects will be processed by other rules in later epochs. This heuristic prevents the creation of rules covering few objects.

One more modification to RULES-A is required because, as previously mentioned, ranges cannot be used in the antecedent part of conditions formed with discrete attributes. At the end of each epoch, the coverage of all rules is updated. If a condition of a rule for a discrete attribute is "always satisfied" and all objects processed so far have the same value for this attribute, this condition will be replaced by the appropriate "attribute-value pair".

To illustrate the operation of the new version of RULES-A, RULES-A1, the algorithm is applied to the simple data set shown in Figure 4.1 (without the attribute *Tolerance*). A step by step execution of RULES-A1 using this data set is provided in Figure 4.2. For this simple data set, 2 epochs are required to complete Phase 1 and only one division of a rule is carried out. The resultant rule set after this phase consists of 5 rules and all 3 discrete attributes are present in their conditional part. The data set is too small to form a pruning set, so that Phase 2 is not applied. Phase 3 simplifies the rules obtained and then produces the final rule set. The corresponding decision tree for this data set is shown in Figure 4.3. The resultant rule set is the most general rule set for this data set.

Id	Heat Treatment (H)	Material (M)	Tolerance	Finish (F)	Route (R)
1	Yes	Steel_3135	10	Medium	R2
2	No	Aluminium	12	Medium	R3
3	Yes	Steel_3135	8	Medium	R2
4	No	Steel_1045	14	Low	R3
5	No	Aluminium	7	High	R4
6	Yes	Steel_1045	9	Medium	R1
7	No	Aluminium	9	Medium	R3
8	No	Aluminium	10	High	R4
9	No	Aluminium	10	Low	R3
10	Yes	Steel_1045	7	Medium	R1
11	No	Steel_1045	7	Low	R3

Figure 4.1 - Training set [Pham and Dimov, 1996].

Phase 1: Initialise the rule set, RS.

Epoch 1:
Processing of object 1:
RS = { }
There is no rule yet to classify object 1.
Create Rule1: H = Yes ∩ M = Steel_3135 ∩ F = Medium ⇒ R = R2
Processing of object 2:
RS = {Rule1}
No rule can classify object 2.
Create Rule2: H = No ∩ M = Aluminium ∩ F = Medium ⇒ R = R3
Processing of object 3:
RS = {Rule1, Rule2}
Rule1 can classify object 3.
Processing of object 4:
RS = {Rule1, Rule2}
No rule can classify object 4.
Rule2 can expand to classify object 4. Modify Rule2 to:
Rule2': H = No ∩ true ∩ true ⇒ R = R3
Note: "true" is an always-satisfied condition
Processing of object 5:
RS = {Rule1, Rule2'}
Rule2' misclassifies object 5.
Conditions for attributes M and F being always-satisfied can be split.
The occurrence frequency of attribute values: (value (count))
Attribute M: Aluminium (1), Steel_1045 (1)
Attribute F: Medium (1), Low (1)
For object 5, the value of attribute M is Aluminium, so that attribute F is selected to split Rule 2'.
Create Rule3: H = No ∩ M = Aluminium ∩ F = High ⇒ R = R4
Create Rule4: H = No ∩ true ∩ F = Medium ⇒ R = R3
Create Rule5: H = No ∩ true ∩ F = Low ⇒ R = R3.
Remove Rule2' from RS.
Processing of object 6:
RS = {Rule1, Rule3, Rule4, Rule5}
No rule can classify object 6.
Create Rule6: H = Yes ∩ M = Steel_1045 ∩ F = Medium ⇒ R = R1
Processing of object 7:
RS = {Rule1, Rule3, Rule4, Rule5, Rule6}
Rule4 can classify object 7.
Processing of object 8:
RS = {Rule1, Rule3, Rule4, Rule5, Rule6}
Rule3 can classify object 8.

Figure 4.2 – A step by step execution of RULES-A1 for the training set in Figure 4.1.

Note: "true" is an always-satisfied condition

Processing of object 9:
RS = {Rule1, Rule3, Rule4, Rule5, Rule6}
Rule5 can classify object 9.
Processing of object 10:
RS = {Rule1, Rule3, Rule4, Rule5, Rule6}
Rule6 can classify object 10.
Processing of object 11:
RS = {Rule1, Rule3, Rule4, Rule5, Rule6}
Rule5 can classify object 11.
End of the data set.
By using the occurrence frequency of attribute values, all conditions of the rules are updated. Only Rule4 is changed to:
Rule4': H = No $\cap$ M = Aluminium $\cap$ F = Medium $\Rightarrow$ R = R3
End of Epoch 1.

Epoch 2:
RS = {Rule1, Rule3, Rule4', Rule5, Rule6}
All objects of the data set are classified by RS.
The rule set is unchanged.

Phase 1 stops.
The resultant rule set of Phase 1:
Rule1: H = Yes $\cap$ M = Steel_3135 $\cap$ F = Medium $\Rightarrow$ R = R2
Rule3: H = No $\cap$ M = Aluminium $\cap$ F = High $\Rightarrow$ R = R4
Rule4': H = No $\cap$ M = Aluminium $\cap$ F = Medium $\Rightarrow$ R = R3
Rule5: H = No $\cap$ true $\cap$ F = Low $\Rightarrow$ R = R3
Rule6: H = Yes $\cap$ M = Steel_1045 $\cap$ F = Medium $\Rightarrow$ R = R1

Phase 2: No changes to the rule set.

Phase 3: By simplifying the rule set from Phase 2, **the final rule set** is created:
Rule1': M = Steel_3135 $\cap$ F = Medium $\Rightarrow$ R = R2
Rule3': F = High $\Rightarrow$ R = R4
Rule4'': M = Aluminium $\cap$ F = Medium $\Rightarrow$ R = R3
Rule5': F = Low $\Rightarrow$ R = R3
Rule6': M = Steel_1045 $\cap$ F = Medium $\Rightarrow$ R = R1

Figure 4.2 (continued)

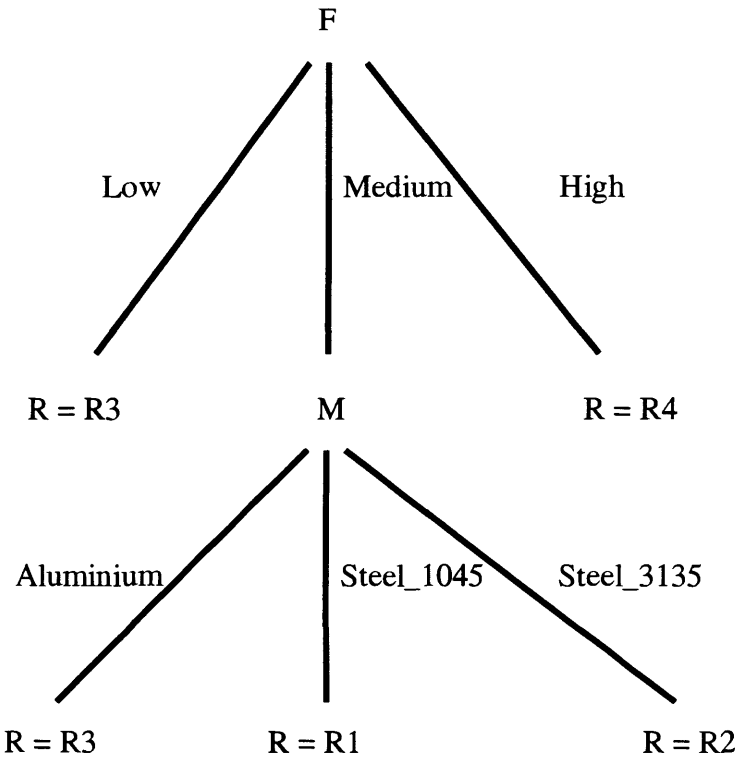


Figure 4.3 – The resultant rule set in Figure 4.2 represented as a decision tree.

4.2.2. Continuous Learning

After removing some very specific rules during the learning phase (Phase 2 of RULES-A), the coverage of the rule sets is reduced. The remaining rules could be expanded to cover more objects. Thus, by omitting the objects covered by the removed rules from the training set, the algorithm could continue to create new rules or expand existing rules. In this way, the generality of the rule set is improved. The algorithm could go through a sequence of learning and pruning iterations before converging on the final rule set.

At the end of each learning iteration and before the pruning process starts with an increased pruning threshold, the algorithm creates one candidate rule set. These candidate rule sets are stored in order to select the best of them at the end. This rule set is considered the final outcome of the algorithm. This evaluation is carried out on a validation set to measure the accuracy of the candidate rule sets. In this research, validation sets include 30% of the objects in the data set. The remaining 70% are used to form the initial training set. If two or more candidate rule sets have the same accuracy on the validation set, the rule set with the smallest number of rules is selected.

4.3. RULES-A1 Algorithm

The proposed new version of RULES-A, RULES-A1, is described in Figures 4.4 – 4.6. Compared with the original algorithm (see Figures 3.1 – 3.4), RULES-A1 has only two phases, Induction and Rule Simplification. Phases 1 and 2 of RULES-A are combined into Phase 1 of RULES-A1. The pruning of small rules is embedded in the learning phase. Phase 2 of RULES-A1 is identical to Phase 3 of RULES-A.

Phase 1 – Induction: Formation of a rule set that covers all training examples with a minimum number of rules.

Phase 2 – Rule simplification.

Figure 4.4 – The improved Rules Extraction System with Adaptivity (RULES-A1).

Step 1. Initialise the rule set (Empty rule set), $Acc_{Val} = 0$, Pruning Threshold = 0.
Step 2. Initialisation of the training set.
Step 3. One epoch:
3.1. If the training set is empty, go to step 4.
3.2. Remove randomly one object, X, from the training set.
3.3. If X is misclassified by the rule set,
divide the misclassifying rule into 2 new rules and compute their estimated capacities and coverage.
Add a new rule which can cover X, go to step 3.1.
3.4. Find a rule, R, that covers or can be expanded to cover X. If there is more than one rule, the rule with the highest evaluation measure is selected.
If X is covered by an existing rule R, update its capacity and coverage.
If R can be expanded to cover X, R is modified and its capacity updated.
Go to step 3.1.
3.5. If X cannot be covered by an existing or expanded rule, create a new rule and add it to the rule set. Go to step 3.1.
Step 4. Removal of redundant rules:
4.1. Update the coverage and capacity of all rules that are not modified in this epoch.
4.2. Remove rules in the rule set that have a capacity equal to 0.
4.3. If the rule set has undergone any changes in this epoch, go to step 2.
Step 5. Pruning:
5.1. Test the rule set on the validation set.
If its accuracy $\geq Acc_{Val}$, set Acc_{Val} to equal the accuracy of this rule set, store the rule set as the final rule set.
5.2. Increase the pruning threshold by 1.
Prune rules with capacity < the pruning threshold, remove from the training set the objects covered by the pruned rules.
Go to step 2.

Acc_{Val}: the accuracy of the rule set on the validation set.

Epoch: one iteration over the training set.

Rule capacity: the number of objects in the training set covered by the rule.

Rule coverage: the area in the object space covered by the rule.

Figure 4.5 - Phase 1 – Induction.

Input: the rule set created after phase 1.

Create a new rule set that is the same as the input rule set.

For each rule in the new rule set:

    For each rule condition:

        Check that omitting this condition from the rule does not cause

            overlapping of rules for different classes in the input rule set.

        If there is no overlapping then remove this condition.

Output: Simplified rule set.

Figure 4.6 - Phase 2 – Rule simplification.

4.4. RULES-A1 Performance

The RULES-A1 algorithm was tested on data sets that contain both continuous and discrete attributes. Only data sets with a sufficient number of objects to form validation sets (containing more than 50 objects) were selected for testing. In total, twelve data sets from the UCI Machine Learning repository [Blake et al., 1998] were used to verify the algorithm performance. The number of attributes and objects of these data sets are given in Table 4.1. A more detailed description of these data sets is given in the Appendix B.

The ten-fold cross-validation test was carried out on all selected data sets. The results obtained using RULES-A1 were compared with those of C5 and RULES 3+ (Table 4.2). The rule sets generated by RULES-A1 have a higher accuracy than both C5 and RULES 3+ on 6 of the data sets and their average accuracy is also higher.

Comparing the results of RULES-A1 and C5, the rule sets obtained by applying RULES-A1 performed better than C5 on 8 of the data sets. In particular, the performance of RULES-A1 was much better than that of C5 on 3 data sets (Abalone, Balance-Scale and Ionosphere) and only significantly worse on 1 data set (Car Evaluation).

Comparing the results of RULES-A1 and RULES 3+, the rule sets obtained by using RULES-A1 performed better than those produced by RULES 3+ on 10 of the data sets. In particular, the performance of RULES-A1 was much better than that of RULES 3+ on 6 data sets (Abalone, Australian, Cmc, Pima, Pageblocks and Yeast) and only significantly worse than that of RULES 3+ on 1 data set (Tic-Tac-Toe).

Table 4.1 – Parameters of the selected data sets.

Data set	Number of Attributes		Number of objects
	Continuous	Discrete	
Abalone	7	1	4177
Australian	14	0	690
Balance-Scale	4	0	635
Car Evaluation	6	0	1728
Cmc	9	0	1473
Credit Approval	6	9	690
Ionosphere	34	0	351
Pageblocks	10	0	5473
Pima	8	0	768
Tic-Tac-Toe	0	9	958
Yeast	8	0	1484
Wdbc	30	0	569

Table 4.2 – Results of the ten-fold cross-validation testing of RULES-A1 against C5 and RULES 3+.

Data set	RULES-A1	C5	RULES 3+
Abalone	**24.58**	20.00	18.00
Australian	**85.36**	83.50	82.93
Balance-scale	**81.12**	77.90	80.03
Car Evaluation	88.71	**92.70**	86.43
Cmc	**55.86**	54.20	48.63
Credit Approval	86.73	85.80	**87.07**
Ionosphere	**90.59**	88.30	90.03
Pageblocks	96.80	**97.40**	92.12
Pima	74.58	**76.30**	66.96
Tic-Tac-Toe	85.97	86.50	**95.53**
Yeast	**56.11**	55.20	47.32
Wdbc	93.49	94.20	**94.69**
Average accuracy	**76.66**	76.00	74.15
Number of top performances	6	3	3

4.5. Performance Improving Techniques

In this section, techniques are proposed to speed up the learning process by applying an "early stopping" strategy and improve the performance of the generated rule set by changing the order of objects in the training set.

4.5.1. Early Stopping

The learning process, Phase 1 of the improved RULES-A algorithm, takes several epochs to converge on a rule set that satisfies the user requirements (the specified pruning threshold). Figure 4.7 shows how the numbers of rules that were split and the size and accuracy of the rule set change during the learning process, from one epoch to another, on seven data sets. For these data sets, 90% of their objects were used to form the training set and the remainder for validation. Starting from an empty set, the algorithm iteratively improves the rule sets formed. Initially, the algorithm generates a relatively large number of rules and then this number is reduced significantly in the following epochs through pruning and expanding some existing rules. As a result, the performance of the generated rule set improves.

It is observed that some specific characteristics of the data sets affect the convergence speed of the algorithm. In particular, they could have an effect on the number of epochs required to converge on a particular rule set. The larger the data set size is, the longer the learning process takes. Some data sets (Balance-Scale, Car Evaluation,

Ionosphere, Tic-Tac-Toe) require a relatively small number of epochs to converge on a rule set but for others (Australian, Pima, Yeast) this takes much longer. It can be seen from Figure 4.7 that within the first 3 epochs after pruning, the size of the rule sets changes significantly and the process of rule set refinement has a major impact. In addition, the number of epochs in learning sections decreases with an increase in the pruning threshold. Thus, to reduce the overall learning time, the learning process after pruning could be stopped early. Therefore, it is proposed to limit the maximum number of epochs to a predefined parameter. In particular, in this research, this parameter is set to 6. It was found empirically that this value was sufficient to enable the rule sets to converge.

4.5.2. Changing the Order of Training Objects

The learning strategy of RULES-A1 is object-driven. The generated rule set is dependent on the order in which training objects are processed. In particular, by changing the order of objects in the training set, the rule forming process could lead to a different set of rules.

To optimise a rule set, the RULES-A1 algorithm attempts to expand some rules to cover more objects without creating overlapping areas between rules for different classes. An early-formed rule covering a relatively large area but classifying few objects in the training set may resist such an expansion. One of the solutions to avoid such "local" optima is to apply RULES-A1 on a number of training sets that contain the same objects but arranged in different orders. Then, the resultant rule sets are tested on a validation set to find the rule set with the highest accuracy.

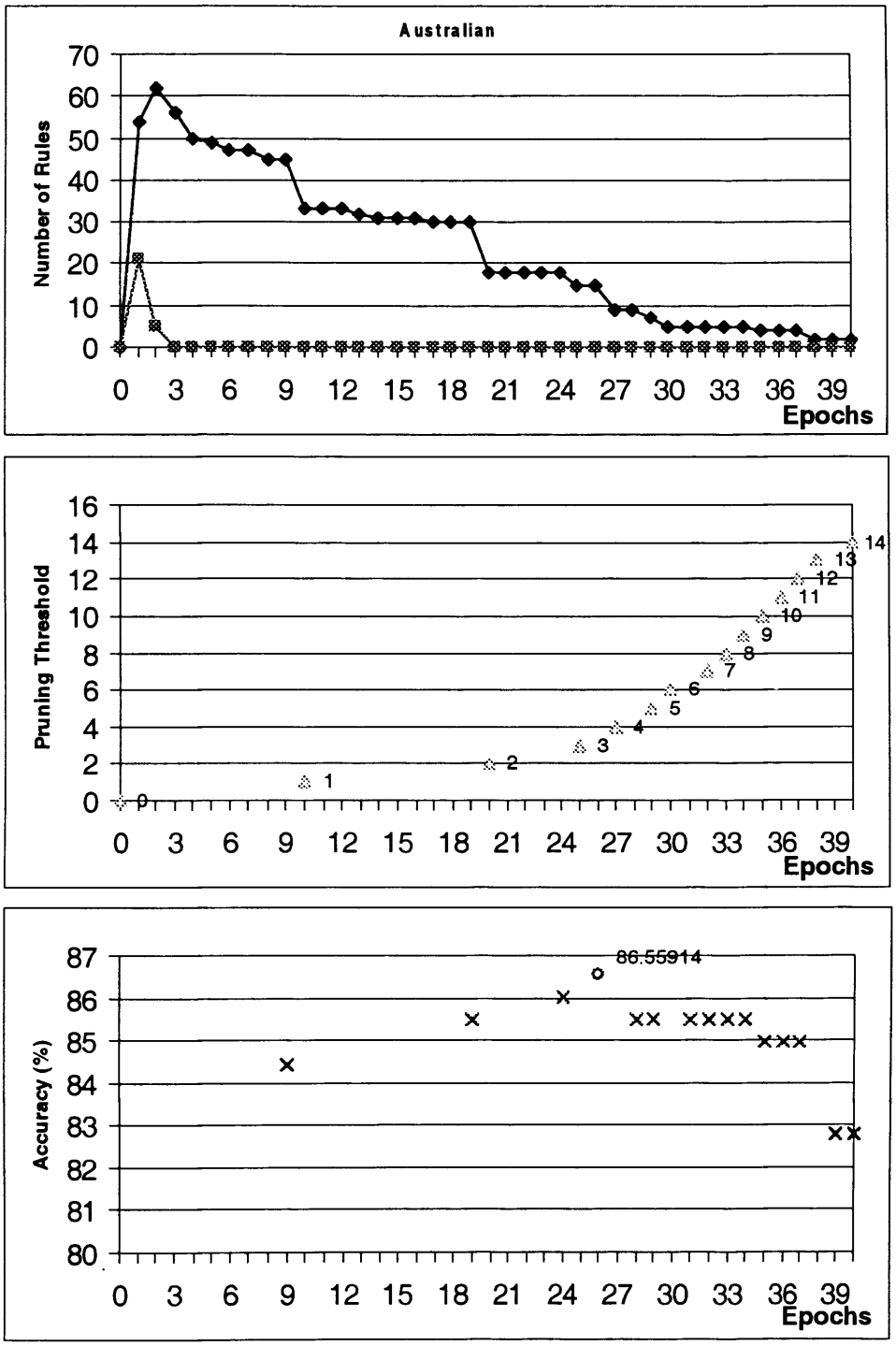


Figure 4.7 – Refinement of rule sets during the learning process.

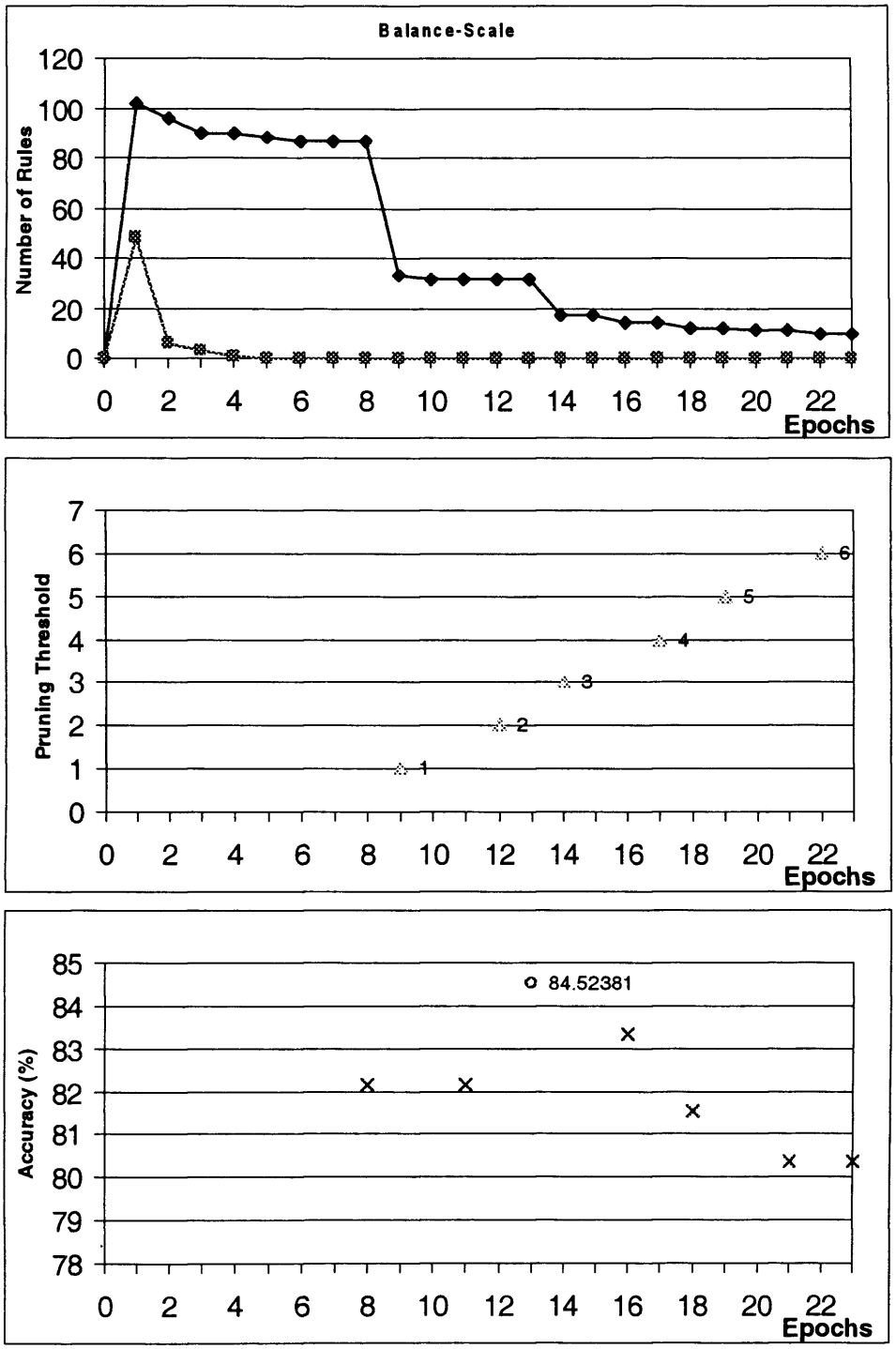
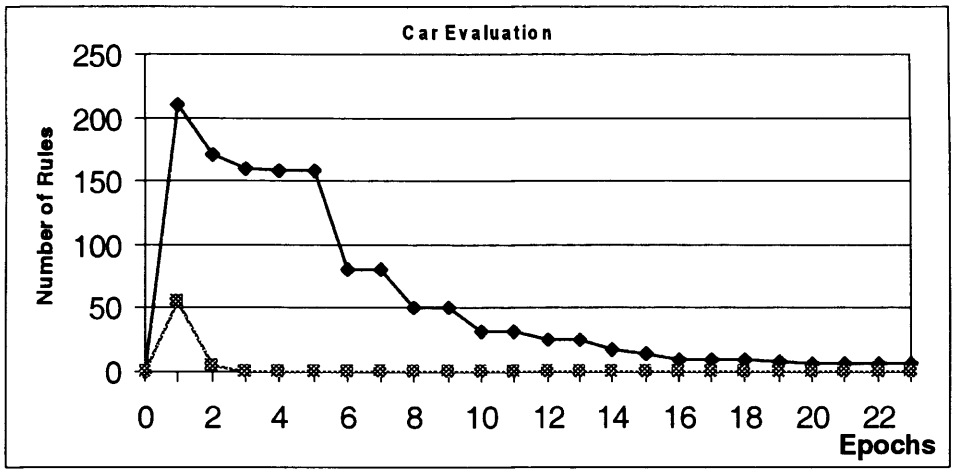
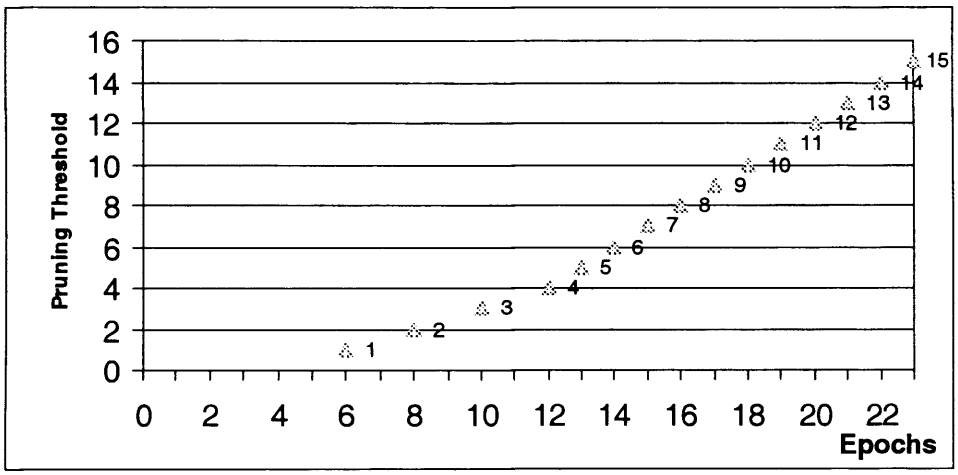


Figure 4.7 (continued)





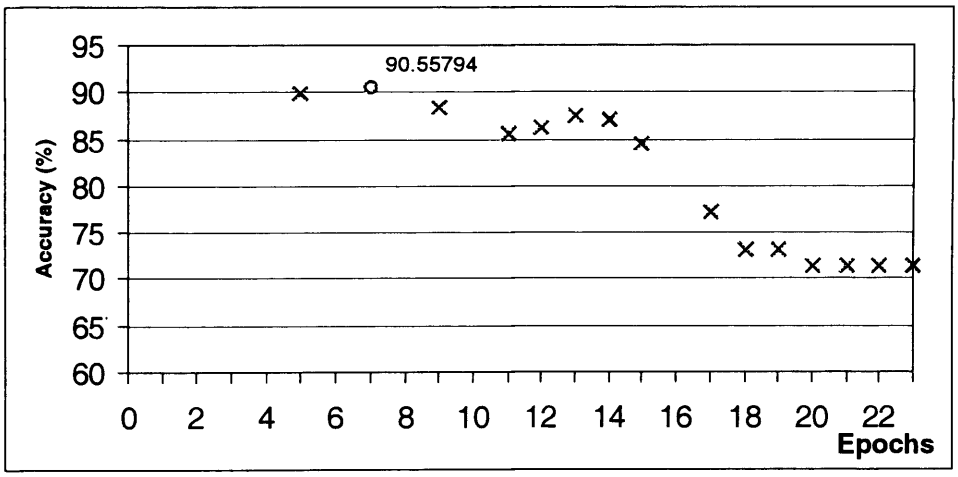
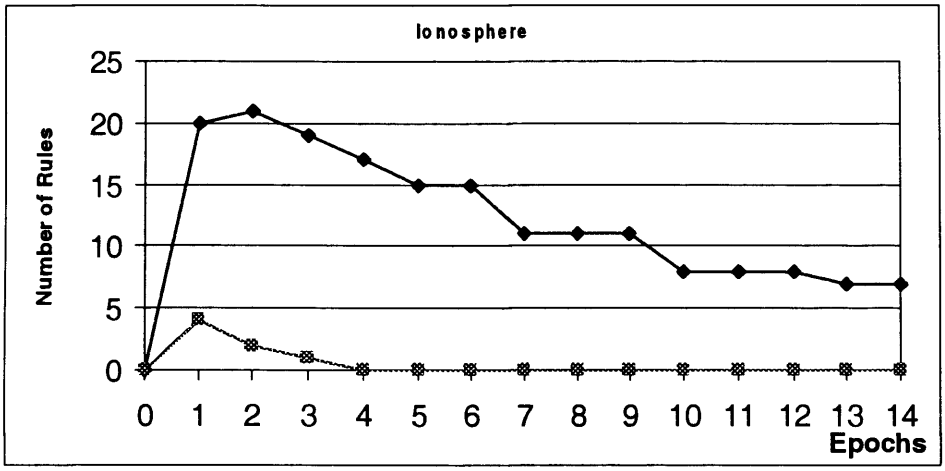
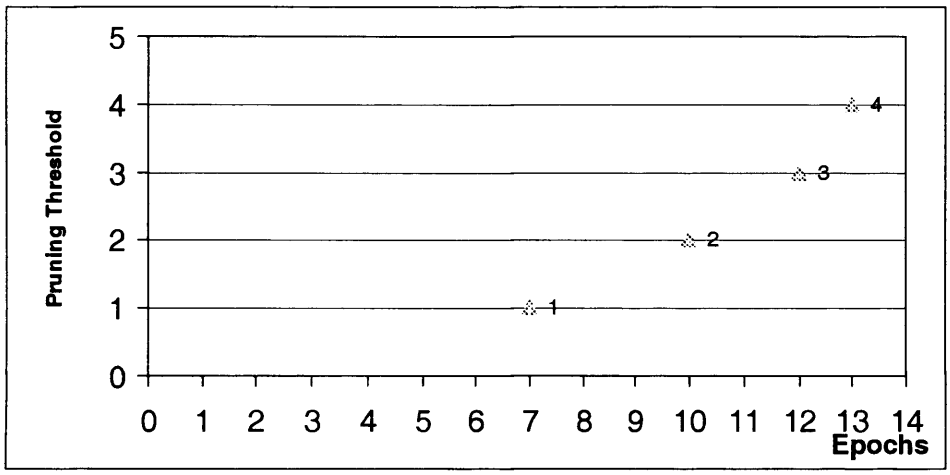


Figure 4.7 (continued)





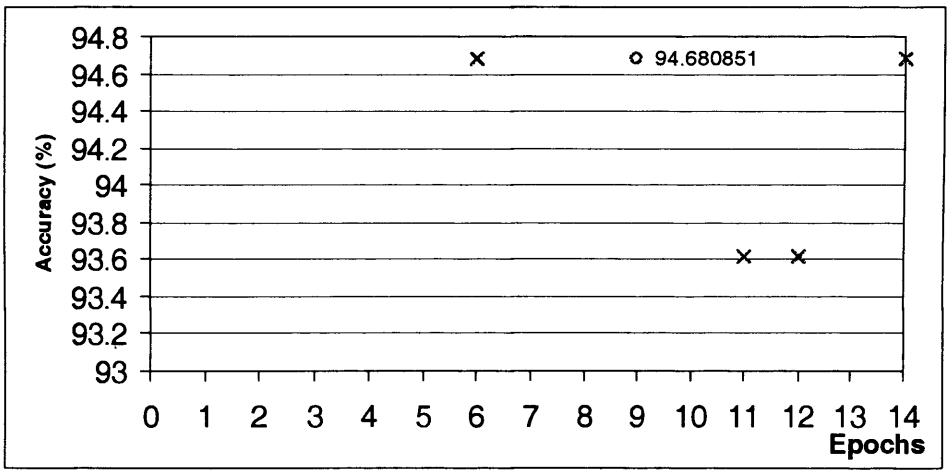
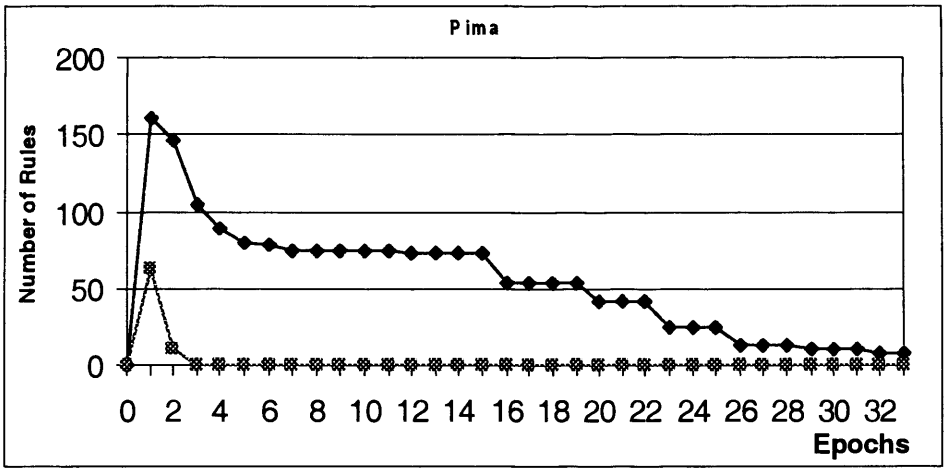
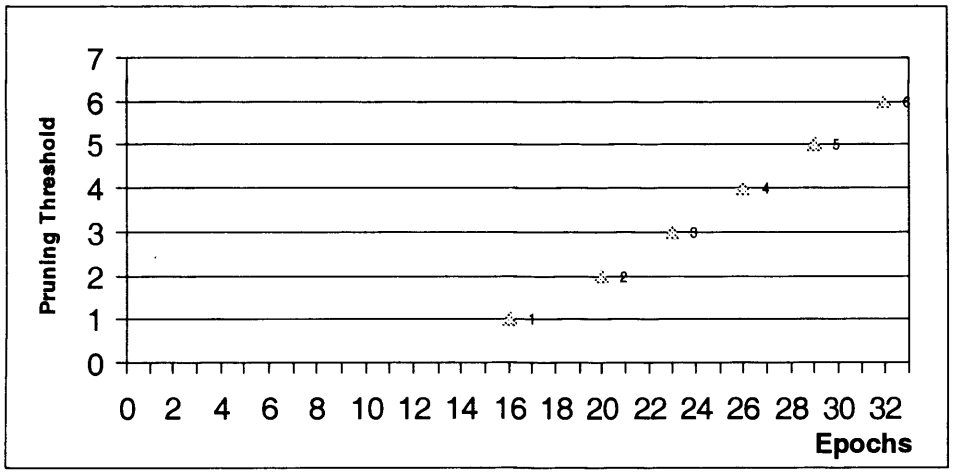


Figure 4.7 (continued)





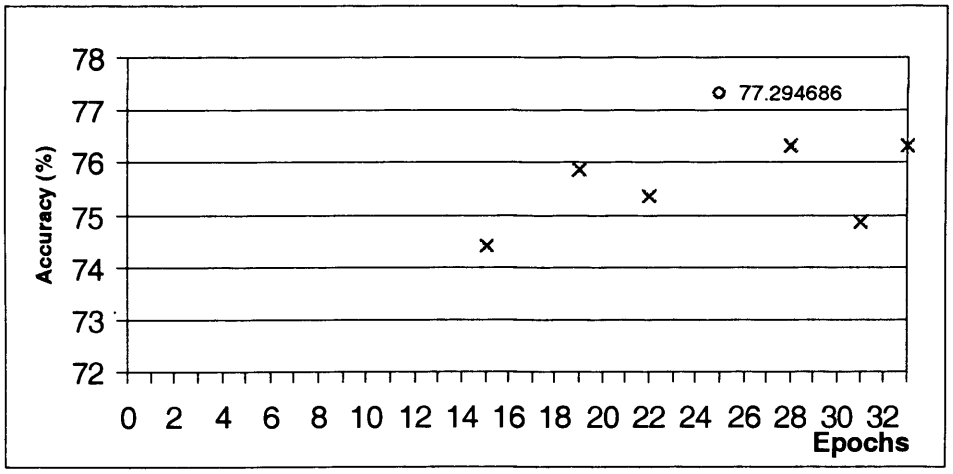
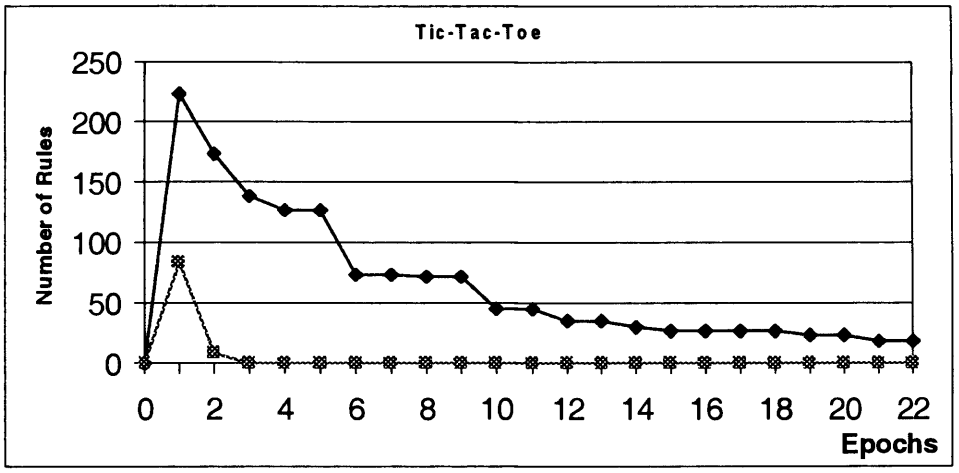
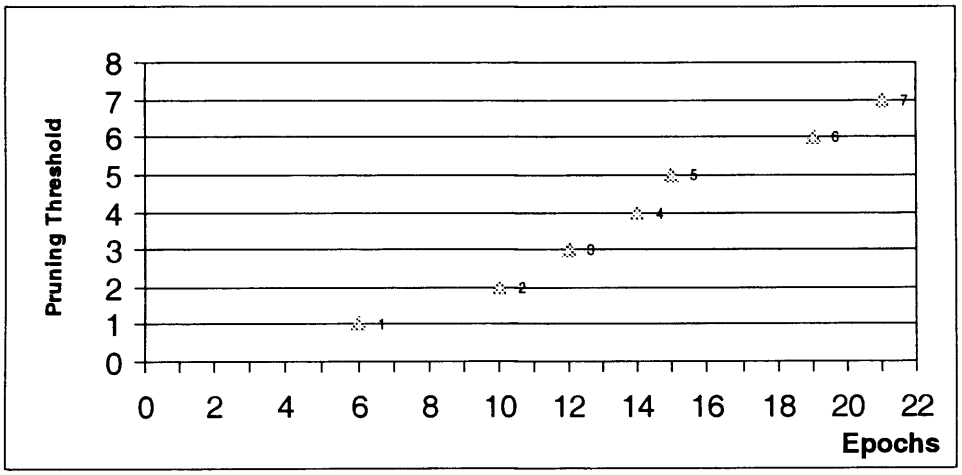


Figure 4.7 (continued)





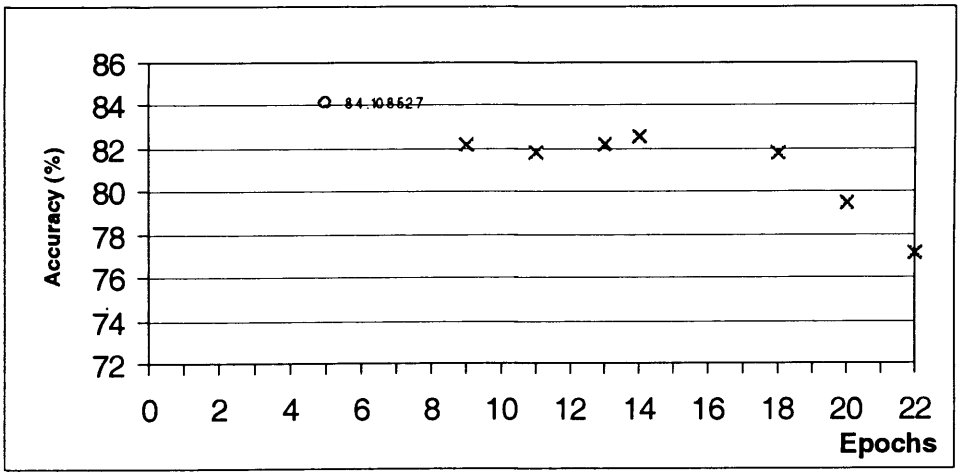
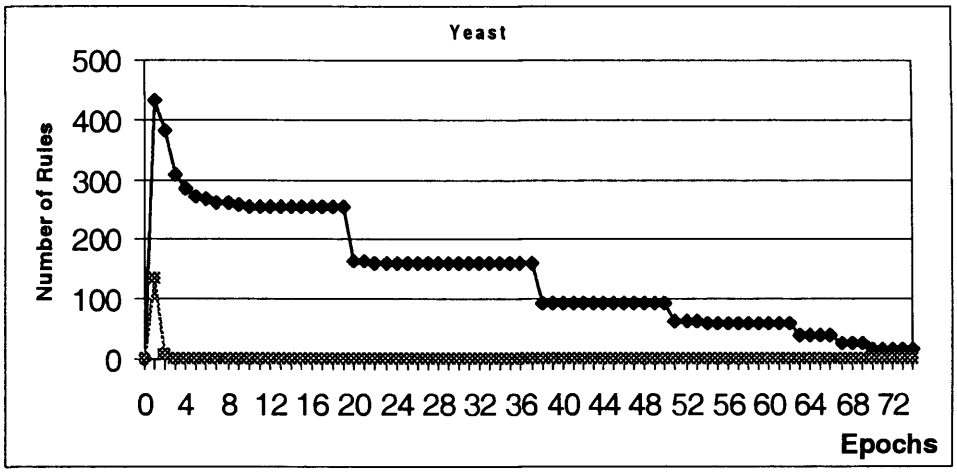
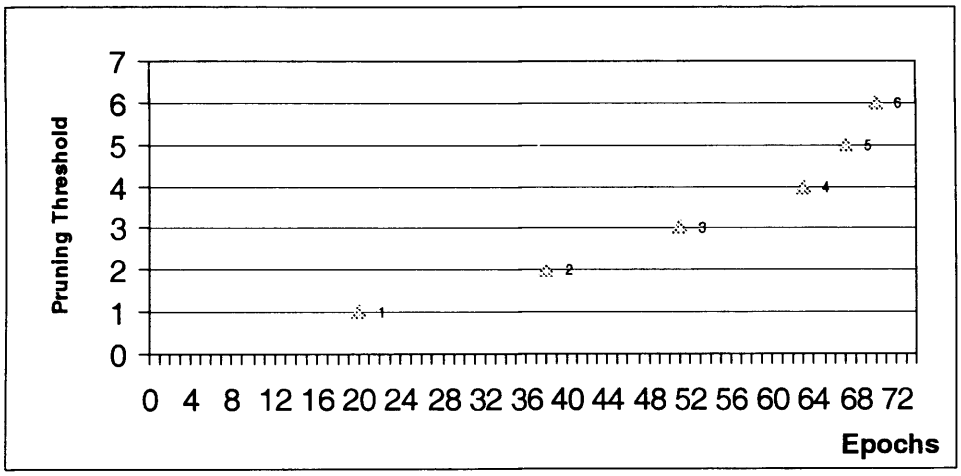


Figure 4.7 (continued)





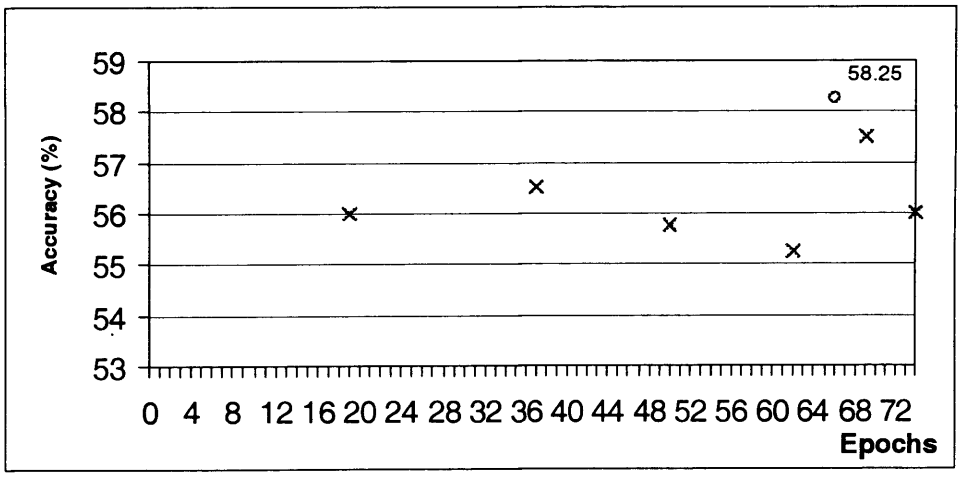


Figure 4.7 (continued)

4.5.3. Performance Analysis

Both the "early stopping" and "changing the order of training objects" strategies were implemented in the RULES-A1 algorithm. The performance of two versions of RULES-A1 is analysed. In the first version, RULES-A2, only the early stopping strategy is implemented and in the second version, RULES-A3, both strategies are applied.

Tables 4.3 and 4.4 show the required number of iterations for RULES-A1 and RULES-A2 to converge on a rule set. The early stopping strategy reduces the number of learning iterations significantly from 599 in total to 331 when the performance on all benchmark data sets is considered. In particular, on 5 data sets (Abalone, Australian, Cmc, Pima and Yeast) the number of iterations is reduced by almost 65%, from 382 to 136. Although the number of learning iterations is reduced significantly, the accuracy of the generated rule sets varies only slightly (Table 4.4). Thus, the implementation of the early stopping strategy has a direct impact on the learning process and reduces the execution time of the RULES-A1 algorithm.

The performances of RULES-A1 and RULES-A3 are compared in Table 4.5. The accuracy of the generated rule sets increases for 10 of the 12 benchmark data sets. The performance of RULES-A3 on the other 2 data sets is only marginally worse. It is worth pointing out that the performance of RULES-A3 on the Tic-Tac-Toe data set is

significantly improved. The reason for this is discussed in the following section. In general, compared with RULES-A1, RULES-A3 produces more accurate rule sets.

The results of cross-validation testing of RULES-A3 against C5 and RULES 3+ are shown in Table 4.6. Similar observations concerning the relative performances of RULES-A3, C5 and RULES 3+ can be made as in section 4.4.

Table 4.3 – Number of iterations required for RULES-A1 and RULES-A2 to converge on a rule set.

Data set	Number of iterations	
	RULES-A1	RULES-A2
Abalone	169	20
Australian	34	24
Balance-Scale	22	18
Car Evaluation	26	25
Cmc	55	29
Credit Approval	18	16
Ionosphere	14	13
Page Blocks	95	86
Pima	47	28
Tic-Tac-Toe	25	21
Yeast	77	35
Wdbc	17	15
Total number of iterations	599	331

Table 4.4 – Results of the cross-validation 10-fold testing of RULES-A2 against RULES-A1.

Data set	Accuracy	
	RULES-A1	RULES-A2
Abalone	24.58	24.76
Australian	85.36	84.67
Balance-Scale	81.12	80.97
Car Evaluation	88.71	90.47
Cmc	55.86	55.70
Credit Approval	86.73	87.00
Ionosphere	90.59	90.84
Page Blocks	96.80	96.75
Pima	74.58	74.05
Tic-Tac-Toe	85.97	86.70
Yeast	56.11	57.36
Wdbc	93.49	93.24
Average accuracy	76.66	76.88

Table 4.5 – Results of the cross-validation 10-fold testing of RULES-A3 against those of RULES-A1.

Data set	RULES-A1	RULES-A3
Abalone	24.58	**24.86**
Australian	**85.36**	85.07
Balance-Scale	81.12	**82.38**
Car Evaluation	88.71	**90.55**
Cmc	55.86	**56.41**
Credit Approval	86.73	**87.03**
Ionosphere	**90.59**	90.29
Page Blocks	96.80	**96.84**
Pima	74.58	**74.76**
Tic-Tac-Toe	85.97	**89.87**
Yeast	56.11	**56.92**
Wdbc	93.49	**93.62**
Average accuracy	76.66	**77.38**
Better rule sets	2	10

Table 4.6 – Results of the cross-validation 10-fold testing of RULES-A3 against those of C5 and RULES 3+.

Data set	RULES-A3	C5	RULES 3+
Abalone	**24.86**	20.00	18.00
Australian	**85.07**	83.50	82.93
Balance-Scale	**82.38**	77.90	80.03
Car Evaluation	90.55	**92.70**	86.43
Cmc	**56.41**	54.20	48.63
Credit Approval	87.03	85.80	**87.07**
Ionosphere	**90.29**	88.30	90.03
Page Blocks	96.84	**97.40**	92.12
Pima	74.76	**76.30**	66.96
Tic-Tac-Toe	89.87	86.50	**95.53**
Yeast	**56.92**	55.20	47.32
Wdbc	93.62	94.20	**94.69**
Average accuracy	**77.38**	76.00	74.15
Top performing rule sets	6	3	3

4.6. Tic-Tac-Toe Problem

Tic-Tac-Toe is one of the games that are terminated by a fixed-size tuple of conditions. Every ending situation of the game consists of only one tuple of 3 suitable plays. Thus, any induced rule has to have at least 3 conditions. All 16 ending principles of the game are described in Figure 4.8.

An additional problem in the Tic-Tac-Toe game is the fact that the ending principles for different classes are disjoint for the training set but highly overlapping in the domain space. Also, the ending principles (rules) for the same class may overlap when the training set is considered and at the same time overlap highiy in the domain space. Figure 4.9 illustrates such cases.

Decision trees are attribute-driven and usually the divide-and-conquer method is applied to construct them. Such trees are formed by processing one attribute at a time to divide objects in the training set into sub-groups belonging to the same class. Unfortunately, in the Tic-Tac-Toe game, the objects representing the ending principles are divided into small groups. Figure 4.10 illustrates this fragmentation. The first node of the decision tree divides P1, P3, P4, P6, P9, P11, P12 and P14 into two separate groups (those with X_5 = 'X' or X_5 = 'O'). These two groups are divided further until the decision tree contains objects belonging to the same class in each of its leaf nodes. The objects representing several ending principles, after being divided into smaller groups, can be wrongly regarded as noise and therefore to be pruned. It is obvious that there is no noise in this particular data set, so that pruning will degrade the performance of the resultant decision tree. This is the main reason for the lowest

X_1	X_2	X_3
X_4	X_5	X_6
X_7	X_8	X_9

$X_i \in \{\text{'X'}, \text{'O'}, \text{'b'}\}$

'b': blank

(a) Conventions

P1: $X_1 = \text{'X'} \cap X_2 = \text{'X'} \cap X_3 = \text{'X'} \Rightarrow \text{Winner} = \text{'X'}$

P2: $X_4 = \text{'X'} \cap X_5 = \text{'X'} \cap X_6 = \text{'X'} \Rightarrow \text{Winner} = \text{'X'}$

P3: $X_7 = \text{'X'} \cap X_8 = \text{'X'} \cap X_9 = \text{'X'} \Rightarrow \text{Winner} = \text{'X'}$

P4: $X_1 = \text{'X'} \cap X_4 = \text{'X'} \cap X_7 = \text{'X'} \Rightarrow \text{Winner} = \text{'X'}$

P5: $X_2 = \text{'X'} \cap X_5 = \text{'X'} \cap X_8 = \text{'X'} \Rightarrow \text{Winner} = \text{'X'}$

P6: $X_3 = \text{'X'} \cap X_6 = \text{'X'} \cap X_9 = \text{'X'} \Rightarrow \text{Winner} = \text{'X'}$

P7: $X_1 = \text{'X'} \cap X_5 = \text{'X'} \cap X_9 = \text{'X'} \Rightarrow \text{Winner} = \text{'X'}$

P8: $X_3 = \text{'X'} \cap X_5 = \text{'X'} \cap X_7 = \text{'X'} \Rightarrow \text{Winner} = \text{'X'}$

P9: $X_1 = \text{'O'} \cap X_2 = \text{'O'} \cap X_3 = \text{'O'} \Rightarrow \text{Winner} = \text{'O'}$

P10: $X_4 = \text{'O'} \cap X_5 = \text{'O'} \cap X_6 = \text{'O'} \Rightarrow \text{Winner} = \text{'O'}$

P11: $X_7 = \text{'O'} \cap X_8 = \text{'O'} \cap X_9 = \text{'O'} \Rightarrow \text{Winner} = \text{'O'}$

P12: $X_1 = \text{'O'} \cap X_4 = \text{'O'} \cap X_7 = \text{'O'} \Rightarrow \text{Winner} = \text{'O'}$

P13: $X_2 = \text{'O'} \cap X_5 = \text{'O'} \cap X_8 = \text{'O'} \Rightarrow \text{Winner} = \text{'O'}$

P14: $X_3 = \text{'O'} \cap X_6 = \text{'O'} \cap X_9 = \text{'O'} \Rightarrow \text{Winner} = \text{'O'}$

P15: $X_1 = \text{'O'} \cap X_5 = \text{'O'} \cap X_9 = \text{'O'} \Rightarrow \text{Winner} = \text{'O'}$

P16: $X_3 = \text{'O'} \cap X_5 = \text{'O'} \cap X_7 = \text{'O'} \Rightarrow \text{Winner} = \text{'O'}$

(b) 16 ending principles

Figure 4.8 – Conventions and 16 ending principles of the Tic-Tac-Toe game.

The 16 ending principles form the perfect set of rules for this problem.

P1 and P10, for two different classes, are disjoint in the training set (each training object is covered only by one principle) but share objects in the domain space. In fact, all objects (X_1 = 'X' $\cap$ X_2 = 'X' $\cap$ X_3 = 'X' $\cap$ X_4 = 'O' $\cap$ X_5 = 'O' $\cap$ X_6 = 'O' $\cap$ X_7 = * $\cap$ X_8 = * $\cap$ X_9 = *) are shared by P1 and P10.

P1 and P2, for the same class, are disjoint in the training set but share objects in the domain space. All objects (X_1 = 'X' $\cap$ X_2 = 'X' $\cap$ X_3 = 'X' $\cap$ X_4 = 'X' $\cap$ X_5 = 'X' $\cap$ X_6 = 'X' $\cap$ X_7 = * $\cap$ X_8 = * $\cap$ X_9 = *) are shared by P1 and P2.

P1 and P4, for the same class, share objects both in the training set and the domain space. All objects (X_1 = 'X' $\cap$ X_2 = 'X' $\cap$ X_3 = 'X' $\cap$ X_4 = 'X' $\cap$ X_5 = * $\cap$ X_6 = * $\cap$ X_7 = 'X' $\cap$ X_8 = * $\cap$ X_9 = *) are shared by P1 and P4.

Note: * could be any of the following symbols 'X', 'O', 'b'

Figure 4.9 – Examples of overlapping end principles of the Tic-Tac-Toe data set.

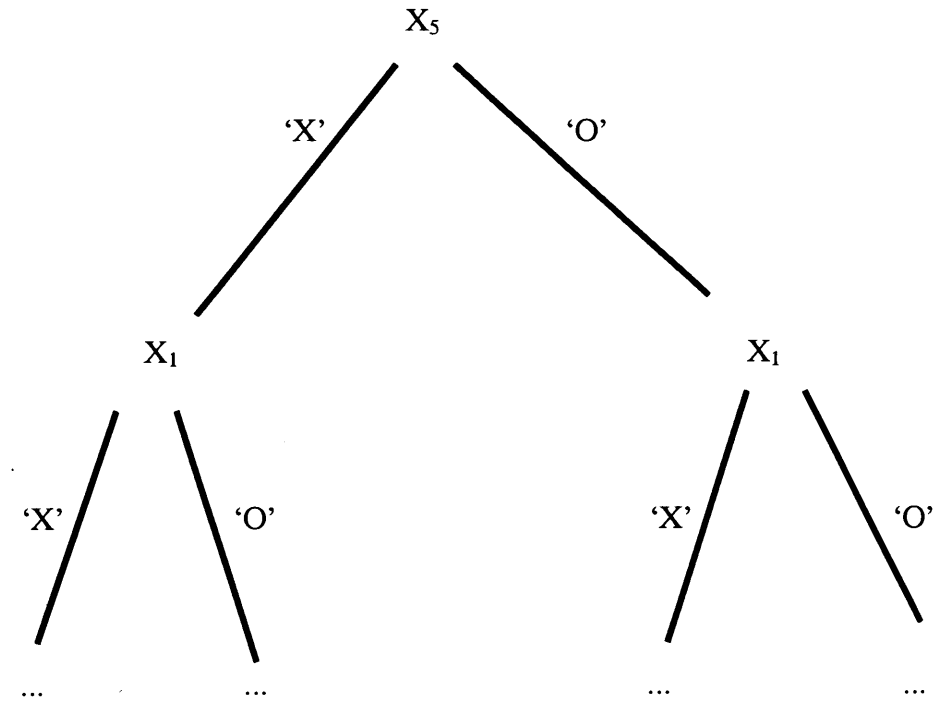


Figure 4.10 – A decision tree for the Tic-Tac-Toe data set.

classification accuracy of the decision tree formed by C5 compared with the other two methods. Such fragmentation is unavoidable when decision trees are constructed by applying divide-and-conquer strategies.

Algorithms of the original RULES family follow the separate-and-conquer-without-reduction approach (See section 2.2.2). They form rules from tuples of attribute-value pairs. The search starts by initially considering candidate rules with only one attribute-value pair, then increasing the attribute-value pairs until a rule is formed. After inducing one rule, objects covered by the formed rule are only marked instead of being removed. The evaluation of formed rules is carried out on the entire training set that includes all marked and unmarked objects, so that it is unchanged. Thus, the existing overlapping of objects in the domain space does not affect the evaluation and hence this rule forming strategy is more suitable for problems such as Tic-Tac-Toe.

To illustrate this, the RULES 3+ algorithm is applied on the Tic-Tac-Toe data set with the parameter PRSET size set to 2 and 10 respectively. This parameter specifies the size of the buffer that stores candidate rules. The algorithm is run on the training set which contains 90% of the total number of objects. With PRSET size set to 2, the algorithm has a small buffer and selects rules from a small set of candidates. As a result, it induces more than 100 rules with several rules having only one condition. With PRSET size set to 10, the algorithm has a larger buffer and can search a larger set of candidate rules. The result is a set containing 26 rules that are listed in Figure 4.11. The first 16 rules of this rule set represent the 16 ending principles and guarantee an accuracy of 100% in testing. The remaining 10 rules are fully covered by the first 16 rules. A suitable post-pruning technique can remove these redundant rules.

R1: IF { |X1 = 'X'| AND |X5 = 'X'| AND |X9 = 'X'| } THEN Winner = 'X'

R2: IF { |X3 = 'O'| AND |X5 = 'O'| AND |X7 = 'O'| } THEN Winner = 'O'

R3: IF { |X2 = 'O'| AND |X5 = 'O'| AND |X8 = 'O'| } THEN Winner = 'O'

R4: IF { |X2 = 'X'| AND |X5 = 'X'| AND |X8 = 'X'| } THEN Winner = 'X'

R5: IF { |X7 = 'X'| AND |X8 = 'X'| AND |X9 = 'X'| } THEN Winner = 'X'

R6: IF { |X3 = 'X'| AND |X6 = 'X'| AND |X9 = 'X'| } THEN Winner = 'X'

R7: IF { |X3 = 'X'| AND |X5 = 'X'| AND |X7 = 'X'| } THEN Winner = 'X'

R8: IF { |X1 = 'O'| AND |X4 = 'O'| AND |X7 = 'O'| } THEN Winner = 'O'

R9: IF { |X1 = 'X'| AND |X4 = 'X'| AND |X7 = 'X'| } THEN Winner = 'X'

R10: IF { |X4 = 'X'| AND |X5 = 'X'| AND |X6 = 'X'| } THEN Winner = 'X'

R11: IF { |X3 = 'O'| AND |X6 = 'O'| AND |X9 = 'O'| } THEN Winner = 'O'

R12: IF { |X7 = 'O'| AND |X8 = 'O'| AND |X9 = 'O'| } THEN Winner = 'O'

R13: IF { |X1 = 'X'| AND |X2 = 'X'| AND |X3 = 'X'| } THEN Winner = 'X'

R14: IF { |X1 = 'O'| AND |X2 = 'O'| AND |X3 = 'O'| } THEN Winner = 'O'

R15: IF { |X1 = 'O'| AND |X5 = 'O'| AND |X9 = 'O'| } THEN Winner = 'O'

R16: IF { |X4 = 'O'| AND |X5 = 'O'| AND |X6 = 'O'| } THEN Winner = 'O'

R17: IF { |X1 = 'O'| AND |X3 = 'O'| AND |X7 = 'X'| AND |X8 = 'O'| AND |X9 = 'X'|
} THEN Winner = 'O'

R18: IF { |X1 = 'X'| AND |X4 = 'X'| AND |X5 = 'O'| AND |X6 = 'X'| AND |X7 = 'O'|
} THEN Winner = 'O'

R19: IF { |X1 = 'X'| AND |X2 = 'O'| AND |X3 = 'X'| AND |X7 = 'O'| AND |X9 = 'O'|
} THEN Winner = 'O'

R20: IF { |X4 = 'O'| AND |X5 = 'O'| AND |X7 = 'X'| AND |X8 = 'X'| AND |X9 = 'O'|
} THEN Winner = 'O'

R21: IF { |X1 = 'X'| AND |X3 = 'O'| AND |X4 = 'O'| AND |X7 = 'X'| AND |X9 = 'O'|
} THEN Winner = 'O'

R22: IF { |X1 = 'X'| AND |X2 = 'O'| AND |X4 = 'X'| AND |X5 = 'O'| AND |X7 = 'O'|
} THEN Winner = 'O'

R23: IF { |X1 = 'O'| AND |X4 = 'X'| AND |X5 = 'O'| AND |X7 = 'X'| AND |X8 = 'O'|
} THEN Winner = 'O'

R24: IF { |X2 = 'X'| AND |X5 = 'O'| AND |X7 = 'O'| AND |X8 = 'X'| AND |X9 = 'X'|
} THEN Winner = 'O'

R25: IF { |X3 = 'O'| AND |X5 = 'O'| AND |X6 = 'X'| AND |X8 = 'O'| AND |X9 = 'X'|
} THEN Winner = 'O'

R26: IF { |X1 = 'O'| AND |X3 = 'X'| AND |X6 = 'O'| AND |X7 = 'O'| AND |X9 = 'X'|
} THEN Winner = 'O'

Figure 4.11 – The resultant rule set from RULES 3+ with PRSET = 10.

The rule forming strategy of RULES 3+ is very appropriate for the Tic-Tac-Toe data set and therefore the generated rule set has the highest accuracy of the 3 tested algorithms.

The RULES-A algorithm checks rule consistency within the domain space. Due to the overlapping of the ending principles in the domain space, if their corresponding rules are in the rule set, they are inconsistent. Thus, each ending principle is represented by more specific rules containing more than 3 conditions. This increases the complexity of the induced rule set. Therefore, the consistency checking strategy of the RULES-A family is unsuitable for the Tic-Tac-Toe data set. Further research is required to improve this strategy, especially to handle tasks similar to the Tic-Tac-Toe game.

The order of the training objects in the data set has an effect on the performance of the algorithm. Due to the existing overlapping of the ending principles in the domain space, this prevents an algorithm in the RULES-A family from expanding rules in the search for more general rules. This problem could be partially resolved by varying the order of the objects in the data set. The results obtained when RULES-A3 is applied on the Tic-Tac-Toe data set (Table 4.5) demonstrate the efficiency of this technique. The accuracy of the rule sets generated by RULES-A3 is higher when compared with RULES-A1.

In conclusion, the specific characteristics of the data sets could influence the overall performance of the generated rule sets. By introducing new techniques addressing this problem, the performance on specific types of data sets could be improved. The existing dependencies between the performance of an algorithm and the specific

characteristics of the data sets should be studied in order to determine the most suitable technique for a particular application.

4.7. Summary

This chapter discusses improvements to RULES-A. The proposed new version RULES-A1 can process data sets with both continuous and discrete attributes. It is also simpler and more convenient to apply. RULES-A1 forms the final rule sets in 2 phases compared with the 3 phases of RULES-A. The pruning is carried out automatically without requiring any interventions by users. RULES-A1 outperforms C5 and RULES 3+ on most of the 12 benchmark data sets.

Also, this chapter discusses further enhancements to RULES-A1 that increase the data processing speed of the algorithm and the accuracy of the generated rule sets. The early stopping strategy limits the maximum number of epochs. In addition, by varying the order of objects in the training set, it is possible to search for more general rule sets (containing fewer rules). As a result, the improved version of RULES-A1, RULES-A3, generates rule sets that outperform those created by the original algorithm, C5 and RULES 3+.

Chapter 5

Improvements to the K-means Algorithm

5.1. Preliminaries

Data clustering is an important data exploration technique with many applications in engineering including parts family formation in group technology and segmentation in image processing. One of the most popular data clustering methods is K-means clustering because of its simplicity and computational efficiency.

The main problem with this clustering method is its tendency to converge at a local minimum. In the first part of this chapter, the cause of this problem is explained and an existing solution involving a cluster centre jumping operation is examined. The jumping technique alleviates the problem with local minima by enabling cluster centres to move in such a radical way as to reduce the overall cluster distortion. However, the method is very sensitive to errors in estimating distortion. A clustering scheme that is also based on distortion reduction through cluster centre movement but not so sensitive to inaccuracies in distortion estimation is proposed in this chapter. The scheme, which is an incremental version of the K-means algorithm, involves adding cluster centres one by one as clusters are being formed. The chapter presents test results to demonstrate the efficacy of the proposed algorithm.

Another drawback of the popular K-means algorithm is the need for several iterations over data sets before it converges on a solution. Therefore, its application is limited to relatively small data sets. The second part of the chapter presents a scaled version of K-means that employs a buffering technique. The new algorithm, Two-Phase K-means, can robustly find a good solution in only one iteration.

5.2. Incremental K-means Algorithm

5.2.1. Conventions

For convenience, in this chapter, the information in a cluster is represented by a triple $<w, N, S>$ where w is the centre of a cluster, N is the number of objects belonging to the cluster (cluster's capacity) and S is the sum of the squared distances between the objects in the cluster and the centre of the Euclidean space. The distortion error I of a cluster is calculated using the following equation.

$$I = S - N\,(d(w, x_0))^2 \qquad (5.1)$$

where $d(w, x_0)$ is the distance between the cluster's centre w and the centre of the Euclidean space x_0.

5.2.2. Motivation

The performance of the K-means algorithm can be measured by considering the movements of the centres of the clusters. When a centre is initiated in an

inappropriate position, it cannot move to an optimum location. For example, in Figure 5.1.a, the data set is split into two disjoint regions R1 and R2 with the same uniform distribution. Suppose that the number of clusters is chosen to be 4. In this example, the hypothesis about the smooth underlying distribution [Chinrungrueng and Sequin, 1995] is not satisfied. Because of the random initialisation, after step 1 of the K-means algorithm, the centres might be located as shown in Figure 5.1.b. There is not any object in region R2 which can belong to any cluster in region R1 due to the distance between the two regions. Thus, no cluster centre in region R1 can move to region R2. Therefore, the clustering obtained by K-means (Figure 5.1.c) differs from the optimal results for this data set (Figure 5.1.d).

To overcome the problem of cluster centres being trapped in inappropriate locations, Fritzke [Fritzke, 1997] suggested a jumping operation to move the cluster centre with the least distortion error to the cluster with the most distortion error (Figure 5.2). When the centre of a cluster is taken away from an inappropriate position, the sum of distortion errors of all clusters increases by a value equal to the sum of the squared distances between objects of the removed cluster and the second nearest cluster centre. However, this calculation does not take into account the fact that the centre of the second nearest cluster centre will be moved when the objects of the removed cluster are added to it. Thus, the increase of this sum will be smaller than otherwise it might be. Moreover, in the proposed operation, the removed cluster centre will be inserted at a random position into the cluster with the largest distortion. There is no estimation of the effect of this operation on the sum of distortion errors of all clusters.

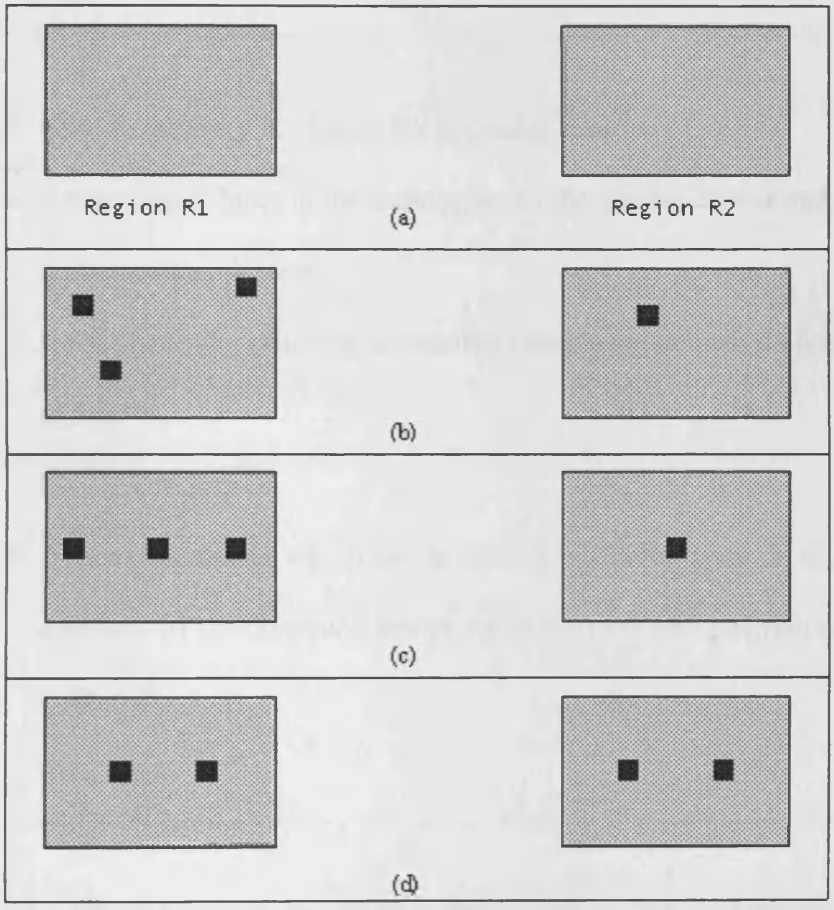


Figure 5.1 – The results of applying K-means (K=4) on two split regions.
(a) Data distribution, (b) Initialisation of clusters' centres,
(c) The result obtained by K-means, (d) The expected clustering.

Step 1: Choose arbitrary K objects for K cluster centres.

Step 2: Assign each object in the training set to the closest cluster and update the centres of the clusters.

Step 3: If the clustering criterion is satisfied (the cluster centres do not move), go to Step 4.

Else, go to Step 2.

Step 4: If there is a cluster which can be moved to a better position to reduce the total sum of the distortion errors, move it to the new position and then go to Step 2.

Else, stop.

Figure 5.2 – The modified K-means algorithm incorporating the jumping operation proposed by Fritzke [Fritzke, 1997].

Pelleg and Moore [Pelleg and Moore, 2000] proposed to start the algorithm with a small number of clusters, K, then doubled it by inserting new cluster centres in suitable positions. There are two problems with the criterion used to evaluate the performance of this operation. First, each cluster is divided independently into two without taking into account the influence of neighbouring clusters. Second, the BIC scoring, that Pelleg and Moore adopted, does not guarantee that the distortion errors of all clusters will be minimised.

In this chapter, a new criterion is proposed to assess the performance of the jumping operation suggested by Fritzke [Fritzke, 1997]. During the learning process, as mentioned already, the operation deals with the local minimum problem by removing a cluster centre from an inappropriate position and inserting it into a more promising position. The increase in the sum of distortion errors of all clusters when one cluster centre is removed and the decrease in the same sum when a new cluster centre is inserted into a new position are two parameters used to evaluate performance. Because it is infeasible to calculate the values of these parameters precisely in the general case, two procedures are described in the following section to estimate them.

5.2.3. Evaluation of Distortion of Clusters

Suppose that the centre of cluster C_i is taken out. In the worst case, all objects belonging to C_i will be allocated to the second nearest cluster C_j without affecting any other neighbouring clusters. The triples (w_i, N_i, S_i) and (w_j, N_j, S_j) characterise C_i and C_j. The triple (w_k, N_k, S_k) of the new cluster C_k is calculated from equations (5.2), (5.3)

and (5.4) and the increase in distortion ΔI in the worst case is calculated using equation (5.5).

$$N_k = N_i + N_j \tag{5.2}$$

$$w_k = \frac{1}{N_k}\sum_{t=1}^{N_k} x_t^k = \frac{1}{N_k}\left(N_i w_i + N_j w_j\right) \tag{5.3}$$

$$S_k = S_i + S_j \tag{5.4}$$

$$\begin{aligned}\Delta I &= I_k - I_i - I_j \\ &= S_k - N_k(d(w_k, x_0))^2 - \left[S_i - N_i(d(w_i, x_0))^2\right] - \left[S_j - N_j(d(w_j, x_0))^2\right] \\ &= N_i(d(w_i, x_0))^2 + N_j(d(w_j, x_0))^2 - N_k(d(w_k, x_0))^2 \\ &= \frac{N_i N_j}{N_i + N_j}(d(w_i, w_j))^2\end{aligned} \tag{5.5}$$

where: I_k, I_i and I_j, are the distortion of C_k, C_i and C_j, respectively,

and x_t^k is an object belonging to cluster C_k.

When the centre of a cluster is moved to a new position, it will cause a decrease in the sum of cluster distortion errors. This decrease cannot be calculated in the general case. In this thesis, it is assumed that a cluster C_z is a hyper-cube with a uniform distribution density p of objects belonging to it (Figure 5.3). When a new cluster centre is inserted, C_z will be split into two clusters C_{z1} and C_{z2}. The triples, (w_z, N_z, S_z), (w_{z1}, N_{z1}, S_{z1}) and (w_{z2}, N_{z2}, S_{z2}), represent these clusters. All objects of C_z are assumed to belong to C_{z1} or C_{z2}. After training, the centres of the two new clusters will be positioned as shown in Figure 5.3. Without loss of generality, the centre of C_z is considered the origin of the co-ordinate system.

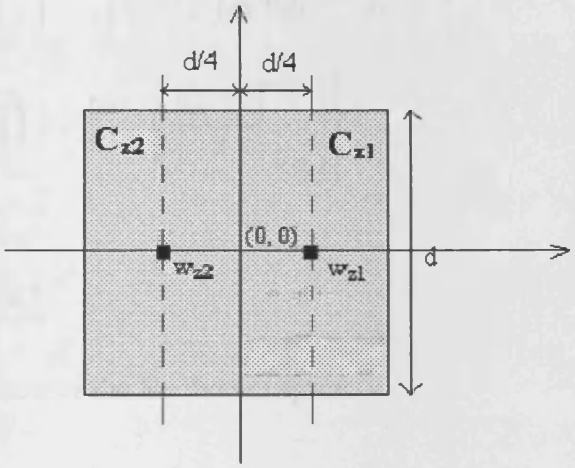


Figure 5.3 – The splitting of C_z into C_{z1} and C_{z2} after training.

The distortion error I_z of cluster C_z is calculated as follows:

$$
\begin{aligned}
I_z &= \int_{-d/2}^{d/2}\int_{-d/2}^{d/2}\cdots\int_{-d/2}^{d/2}\left(d(x,x_0)\right)^2.p.dx^{(1)}dx^{(2)}...dx^{(N_d)} \\
&= \int_{-d/2}^{d/2}\int_{-d/2}^{d/2}\cdots\int_{-d/2}^{d/2}\left(\sum_{t=1}^{N_d}(x^{(t)})^2\right).p.dx^{(1)}dx^{(2)}...dx^{(N_d)} \\
&= p.\sum_{t=1}^{N_d}\left(\int_{-d/2}^{d/2}\int_{-d/2}^{d/2}\cdots\int_{-d/2}^{d/2}\left((x^{(t)})^2\right)dx^{(1)}dx^{(2)}...dx^{(N_d)}\right) \\
&= p.\sum_{t=1}^{N_d}\left(\left(\prod_{\substack{j=1\\ j\neq t}}^{N_d}\left|x^{(j)}\right|_{-d/2}^{d/2}\right)\left(\left|\frac{(x^{(t)})^3}{3}\right|_{-d/2}^{d/2}\right)\right) \\
&= \frac{p.N_d.d^{N_d+2}}{12} \\
&= \frac{N_z.N_d.d^2}{12}
\end{aligned}
\tag{5.6}
$$

where N_d is the dimension of the Euclidean space.

Because C_z is a cube with a uniform distribution, the two new clusters C_{z1} and C_{z2} contain the same number of objects $N_{z1} = N_{z2} = N_z/2$. Using equation 5.6, the decrease in distortion errors is calculated as follows:

$$
\begin{aligned}
\Delta D &= \frac{N_{z1}N_{z2}}{N_z}\left(d(w_{z1},w_{z2})\right)^2 \\
&= \frac{N_z}{4}\left(\frac{d}{2}\right)^2 \\
&= \frac{3I_z}{4N_d}
\end{aligned}
\tag{5.7}
$$

By applying the jumping operation, the sum of the distortion errors will be changed by a value $\Delta M = \Delta I - \Delta D$. If ΔM is smaller than 0, the operation could lead to better clustering.

As the performance of the jumping operation is evaluated based on two estimated parameters in the cluster centre removal and insertion operators, this may introduce additional errors. An incremental strategy can be used to eliminate the removal of a cluster centre and the dependence on the initial positions of cluster centres. An incremental algorithm starts with the number of clusters K being set equal to 1 and increasing by 1 in each step. With each increase of K, a new cluster centre is inserted into the cluster with the most distortion and then objects are re-assigned to clusters until the centres do not move. The process is repeated until K reaches the specified number of clusters. A new improved K-means algorithm with this incremental strategy will be described in the next section. The proposed algorithm has the advantage of determining near optimal cluster centre positions.

To the author's knowledge, there is another K-means clustering algorithm [Likas et al., 2003] with a similar incremental strategy. In each step of the incremental process, that algorithm uses a local search procedure to calculate the position of the new cluster centre assuming the positions of the current cluster centres are optimal and can remain fixed. Because of the dynamic nature of clusters in a K-means operation, this calculation will not yield the optimal position for the new cluster centre for each step. The position error accumulated over the clustering process can affect the final performance of the algorithm.

5.2.4. Algorithm Description

The Incremental K-means algorithm is summarised in Figure 5.4. Phase 1 includes steps that are similar to the steps of the conventional K-means algorithm, except in its restriction on where the new cluster centre can be placed. The centres of all existing clusters do not change their positions, which makes the algorithm less dependent on the random placement of the new centres.

The complexity of the new algorithm can be assessed using the formula:

$O(K^2 * n * num\_of\_iterations)$

where n is the number of objects and *num_of_iterations* is the largest possible number of iterations in Phase 1.

Compared with the complexity $O(K*n*num\_of\_iterations)$ of the K-means algorithm, the Incremental K-means algorithm requires K times more iterations.

When there are K clusters, the new algorithm needs to run Phase 1 K times, each iteration being equivalent to one execution of the traditional K-means algorithm. Of those K times, (K-1) are considered intermediate steps that prepare the data for the next iteration. Therefore, only the last iteration of Phase 1 has to satisfy the strict end condition defined in Step 3. In this thesis, the end condition for each intermediate iteration is relaxed and tested separately.

Assign $K = 1$.

Phase 1: *Normal training*

Step 1: If $K = 1$, choose an arbitrary point for a cluster centre.

If $K > 1$, insert the centre of the new cluster in the cluster with the greatest distortion.

Step 2: Assign each object in the training set to the closest cluster and update its centre.

Step 3: If the cluster centre does not move, go to Phase 2.

Else, go to Phase 1, Step 2.

Phase 2: *Increasing the number of clusters*

If K is smaller than a specified value, increase K by 1 and go to Phase 1 – Step 1.

Else, stop.

Figure 5.4 – The Incremental K-means algorithm.

In the initial step of each run of Phase 1 of the Incremental K-means algorithm, a new cluster centre is inserted in the cluster with the largest distortion error. The insertion of the new centre affects mostly the objects belonging to this cluster.

5.2.5. Performance

Six artificial data sets and six real data sets from the UCI Repository [Blake, et al., 1998] were used to test the proposed new algorithm. The main parameters of these data sets are provided in Table 5.1. The description of these data sets is presented in the Appendix B. The object distribution of the six artificial data sets is shown in Figure 5.5.

The research carried out by Bottou and Bengio [Bottou and Bengio, 1995] and Bilmes et al. [Bilmes et al., 1997] showed that it takes on average 15 iterations for the K-means algorithm to reach a local minimum. The clustering process could be stopped by specifying termination conditions such as a predefined number of iterations and the reduction of the distortion errors in one iteration being less than a given value ε. In this work, these two termination criteria are used. In particular, the maximum number of iterations is set to 20 and ε is specified to be 10^{-7}. The algorithm stops when one of these conditions is satisfied.

Table 5.1 – Characteristics of test data sets.

(a) Real data sets.

	Balance-Scale	**Ionosphere**	**Iris**	**Pima**	**Wine**	**Zoo**
Number of attributes	4	34	4	8	13	17
Number of objects	635	351	150	768	178	101

(b) Artificial data sets

	Uniform1	**Uniform2**	**Uniform3**	**Gauss1**	**Gauss2**	**Gauss3**
Number of attributes	2	2	2	2	2	2
Number of objects	421	1084	800	848	1220	800

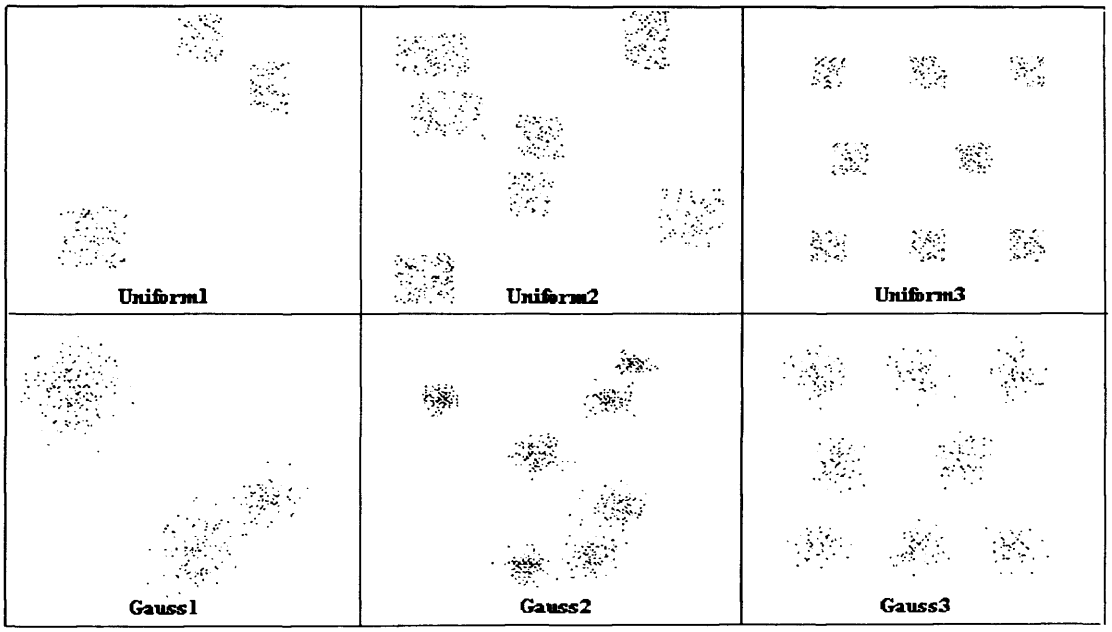


Figure 5.5 – Distribution of objects in the artificial data sets.

Due to the random nature of the K-means algorithm, it is important to conduct a large number of tests to demonstrate its performance in a statistically significant way. When the problem has R disjoint and distant regions and K clusters should be formed, an extremely large number of possibilities exist to allocate the K cluster centres to the different regions. Each particular allocation will lead to different distortion errors. Unfortunately, R is not known in real problems. Many researchers select a large K and a small number of tests, which may not lead to optimal clustering results. In this work, K was selected in the range 1 to 15 and the number of tests for each data set was taken as 500.

Figure 5.6 shows the results obtained by applying four different versions of the K-means algorithm, original K-means, K-means with the jumping operation, Incremental K-means and Incremental K-means with predefined termination conditions, to the 12 data sets. On all data sets, except the Balance-Scale data set, the K-means algorithm with the jumping operation outperforms the original K-means algorithm in spite of the fact that the results obtained are far from the optimal solution. Also, on all data sets, the Incremental K-means algorithm groups objects in clusters whose average distortion error is very close to the smallest distortion error of any of those clusters. This means that the Incremental K-means algorithm does not depend on the specific characteristics of the data sets and the value of K, and produces reliable and optimal clustering of objects.

Figure 5.7 gives the running times of the K-means algorithm, Incremental K-means algorithm and Incremental K-means algorithm with predefined termination condition. All algorithms were implemented in C++ and executed on a Pentium II 300MHz PC.

Although the theoretical complexity of Incremental K-means is a function of K^2, the experiments carried out show that the running time depends linearly on K. By specifying termination conditions, the running time is reduced without sacrificing the quality of the clustering results (Figure 5.6).

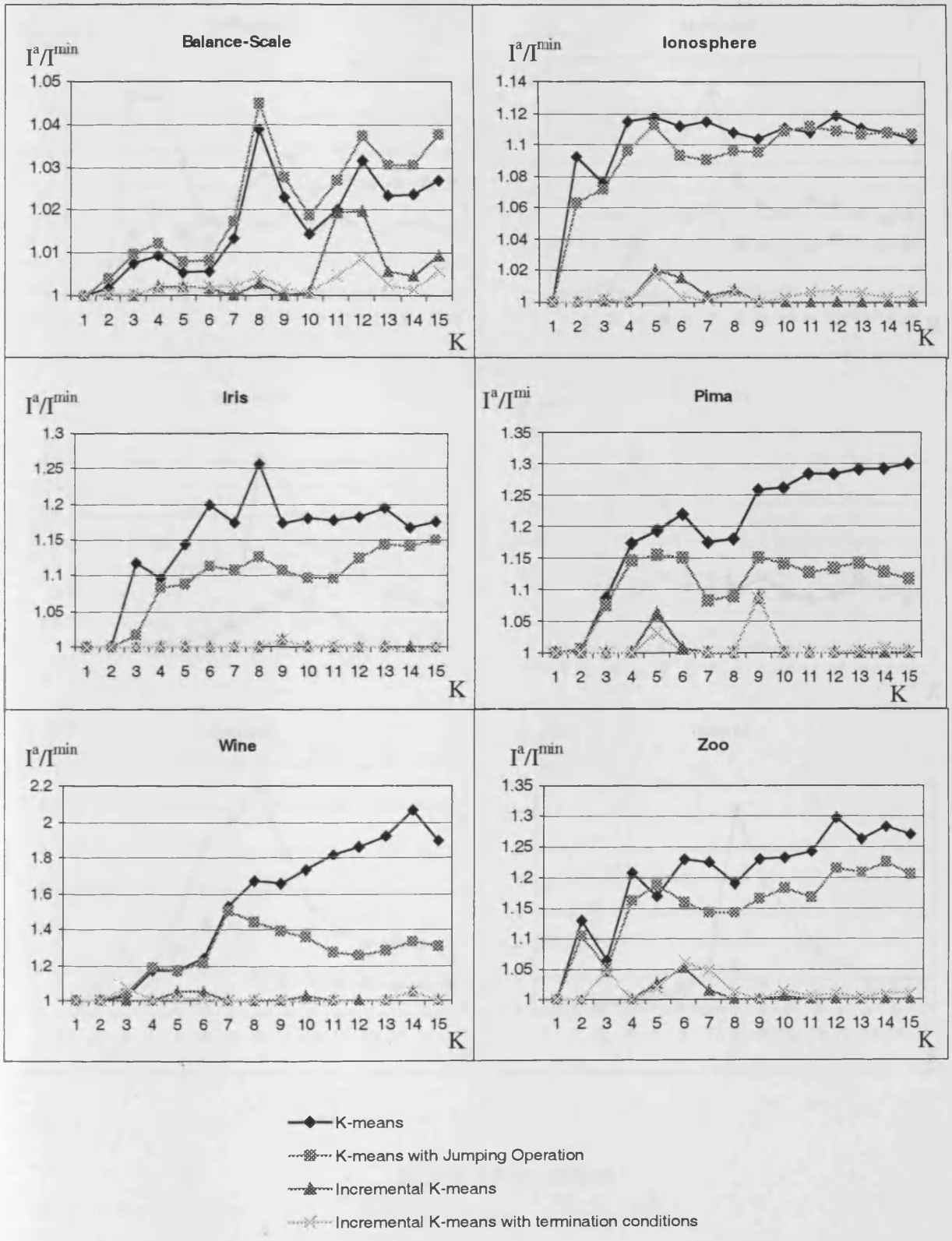


Figure 5.6 – Clustering results of K-means, K-means with jumping operation, Incremental K-means and Incremental K-means with termination conditions.

Note: I^a and I^{min} are the average and the minimum values of clusters' distortion errors.

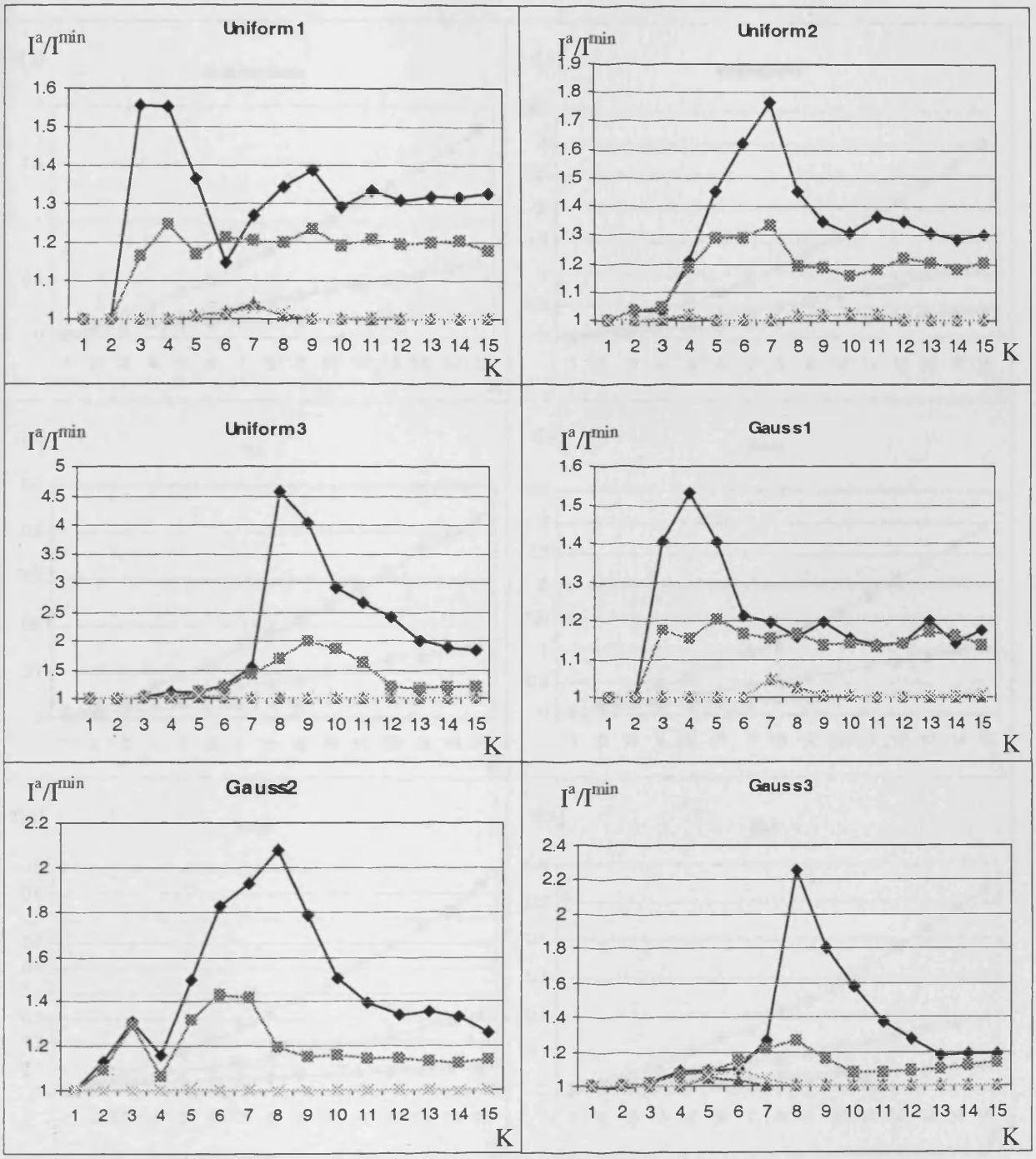


Figure 5.6 (continued)

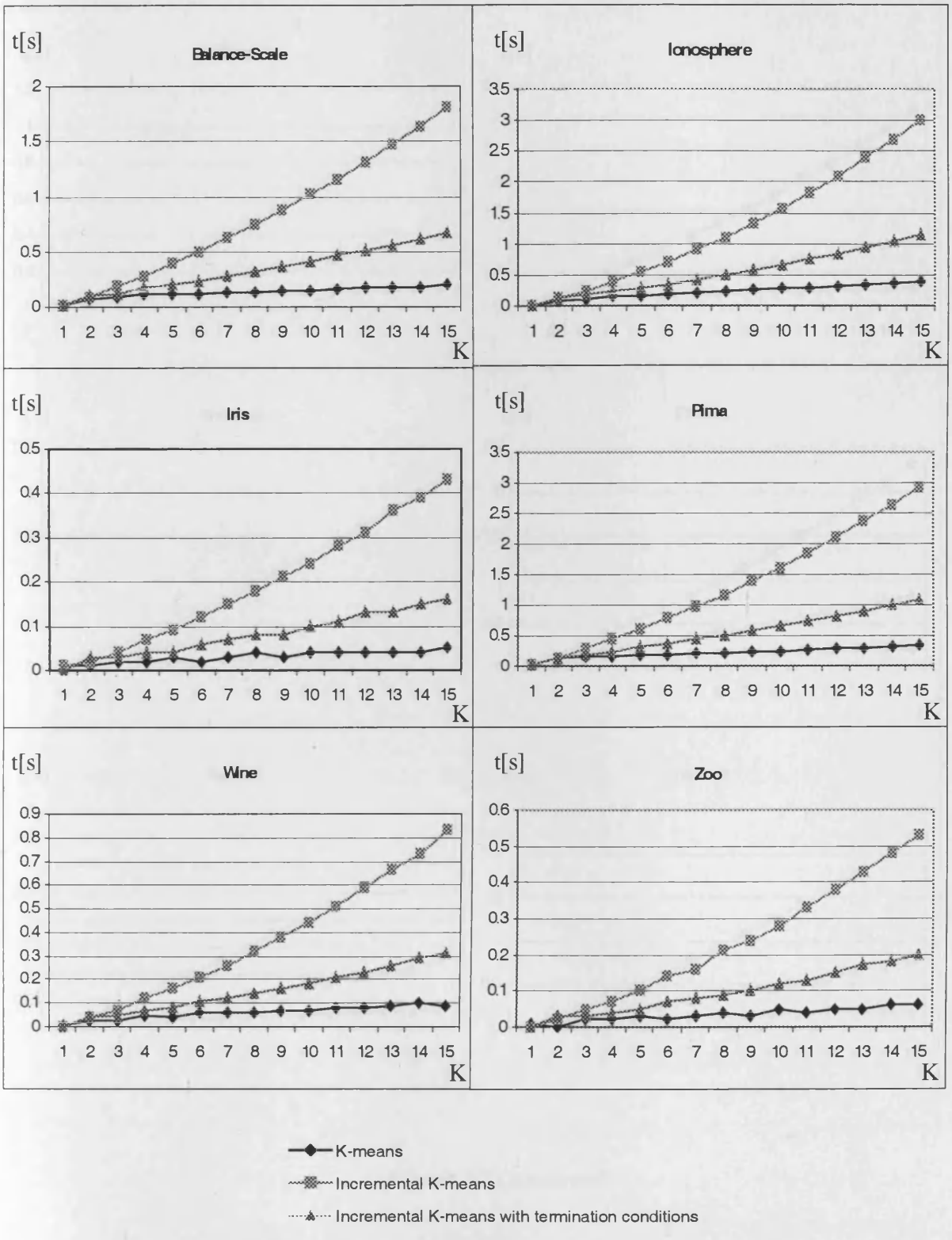


Figure 5.7 – Comparison of the running times of K-means, Incremental K-means and Incremental K-means with termination conditions.

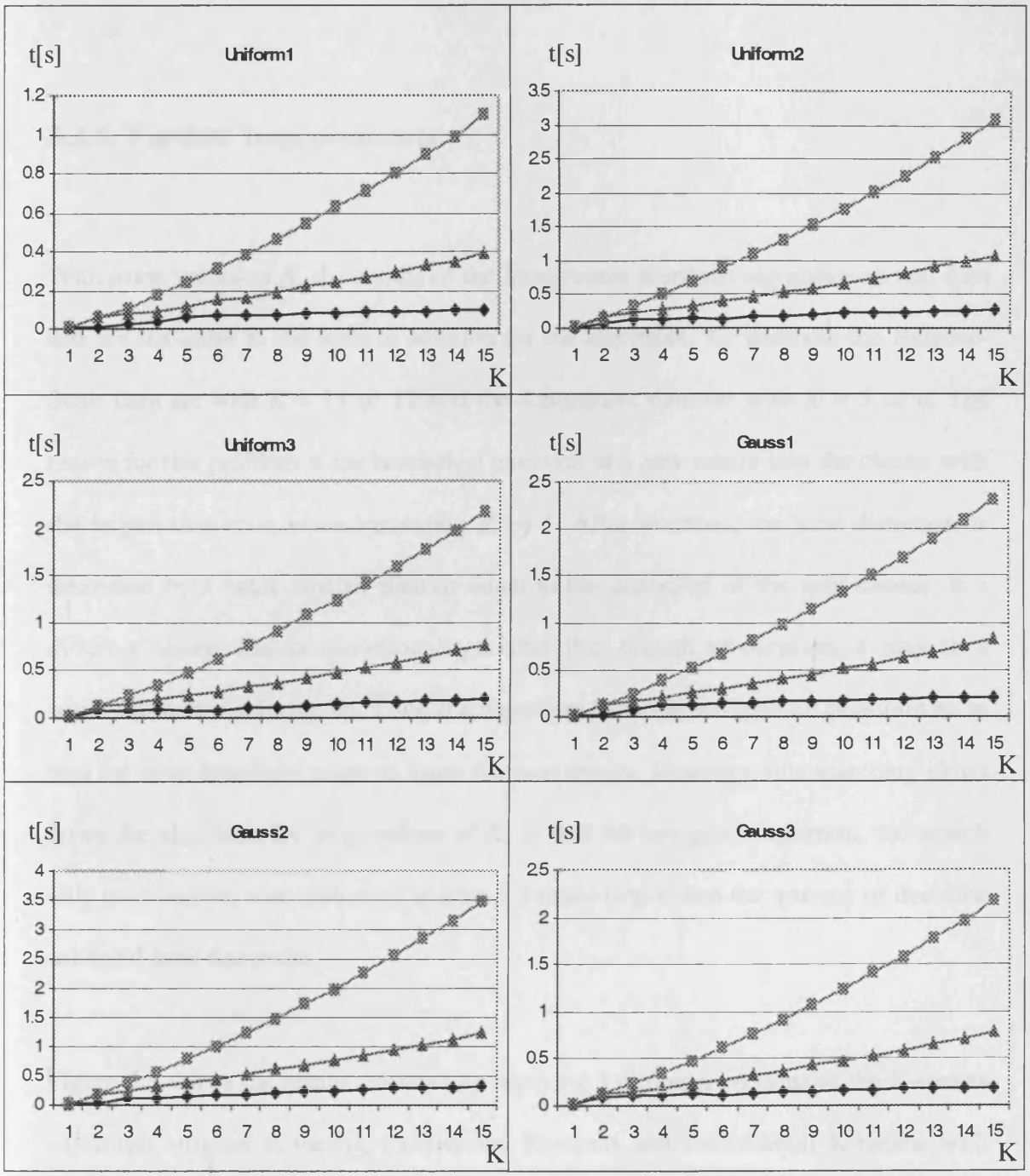


Figure 5.7 (continued)

5.2.6. Further Improvements

With some values of K, the results of the Incremental K-means algorithm on real data sets are not close to the optimal solution for the algorithm, for example the Balance-Scale data set with K = 11 or 12 and the Ionosphere data set with K = 5 or 6. The reason for this problem is the heuristical insertion of a new centre into the cluster with the largest distortion when increasing K by 1. After insertion, the total distortion is decreased by a value smaller than or equal to the distortion of the split cluster. If a different cluster has its distortion larger than this amount of decrease, it may be a better choice for splitting up. Thus, the algorithm has to investigate all possibilities to find the most beneficial place to insert the new cluster. However, this searching slows down the algorithm for large values of K, so that for any given insertion, the search only tries clusters with distortion at least 1.5 times larger than the amount of decrease achieved until that point.

Figure 5.8 shows the results obtained by applying 3 different versions of the K-means algorithm, original K-means, Incremental K-means and Incremental K-means with cluster search to the 6 real data sets. The third version has all its results close to 1 for all values of K, demonstrating that the search strategy helped it to handle cases that the plain incremental version had difficulties with.

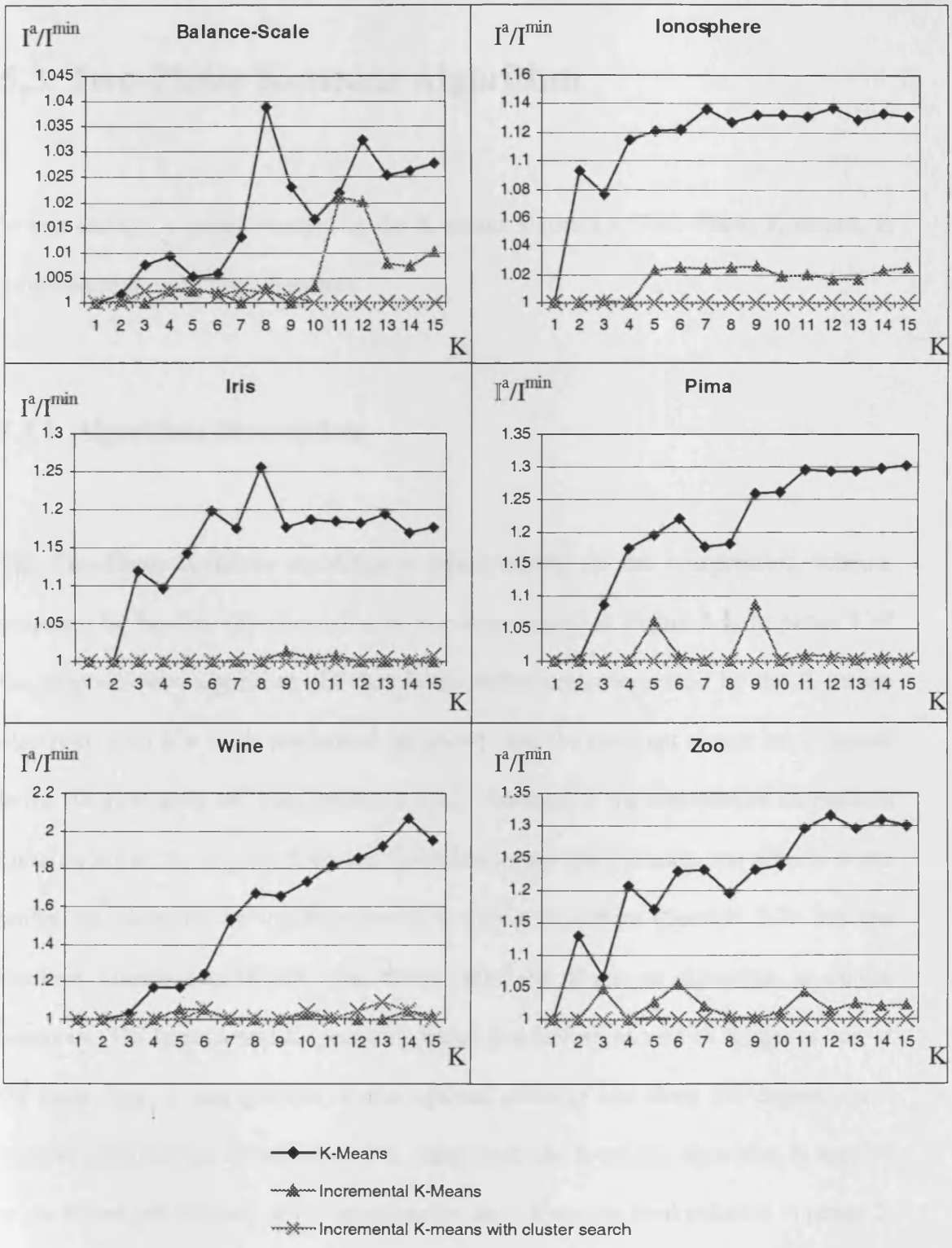


Figure 5.8 – Clustering results of K-means, Incremental K-means and Incremental K-means with cluster search.

Note: I^a and I^{min} are the average and the minimum values of the distortion errors of the clusters.

5.3. Two-Phase K-means Algorithm

In this section, a scaled version of the K-means algorithm, Two-Phase K-means, is proposed to process large data sets.

5.3.1. Algorithm Description

The Two-Phase K-means algorithm is based mainly on the compression scheme proposed by Bradley (Section 2.3.2.2). It is summarised in Figure 5.9. In phase 1 of the proposed new algorithm, the data in the buffer are compressed by the K-means algorithm with $K = K_1$ (a predefined parameter) and the resultant cluster set is stored in the "compression set" (see Section 2.3.2.2). Because of the dependence on random initialisation of the original K-means algorithm, in the first iteration, the objects in the buffer are clustered by the Incremental K-means algorithm (Section 5.2) and the resultant cluster centres are used to initialise the K-means algorithm in future iterations. The Incremental K-means algorithm is a slower version of K-means but, at the same time, it can produce a near optimal solution and does not depend on a random initialisation. When phase 1 is completed, the K-means algorithm is applied to the stored sub-clusters of the compression set to form the final solution in phase 2. Each sub-cluster is treated as a regular object weighted by its number of objects.

Phase 1:

Repeat:

- Fill the buffer with new objects from the data set.
- If this is the first iteration

  Then Apply the Incremental K-Means algorithm to the buffer

  Else Apply the original K-Means algorithm to the buffer using the clusters' positions from the previous iteration as its initialisation.
- Add the cluster set to the compression set.

Until the data set is empty.

Phase 2:

- Apply the Incremental K-Means algorithm to the compression set with each object weighted by the number of objects belonging to it from Phase 1.

Figure 5.9 - The Two-Phase K-Means algorithm.

The new scheme can reduce the negative effect on the performance of the algorithm caused by the use of a buffer that is not representative of the data set as a whole. Phase 1 finds K_1 representatives of the objects in the buffer without considering the accumulated knowledge. In the best case, when the buffer is a true image of the data, each cluster will be represented by $m * K_1 / K$ sub-clusters in the compression set where m is the number of iterations or the number of times the buffer is filled and K is the expected number of clusters. If sufficient information is stored in the compression set, Phase 2 can induce an optimal final solution. When the balancing problem occurs in the buffer, depending on the cluster's contribution to the distortion, it may contain approximately K_1 / K objects and their corresponding weights stored in the compression set. If, for a cluster, there is less stored information in any particular iteration, this may affect the follow up iterations. Thus, the problem of having a set of objects that is not representative of the data set as a whole could be resolved by forming a compression set in the context of the entire problem domain.

Phase 1 has a negative effect on the quality of the final clustering result because a large number of objects have to be represented in a buffer with a limited size (K_1 sub-clusters in the compression set). If a larger K_1 is selected, the computational time increases and more memory is required to execute the algorithm. However, at the same time, the quality of the final clustering result will improve. In practice, K_1 is selected by the user depending on the characteristics of the data.

5.3.2. Performance

Two data sets KDD98 and CoverType from the UCI KDD repository [Hettich and Bay, 1999] and the UCI Machine Learning repository [Blake et al., 1998] respectively, were used in testing. These data sets were selected for their large size. The pre-processing of these data sets is described in the following sections.

The KDD98 data set was converted into the required data format using the same software for data pre-processing as that used by Farnstrom [Farnstrom et al., 2000]. The termination condition for the original K-means algorithm depends on the average squared distance of the formed clusters and the algorithm stops before it converges to a local minimum. This condition was modified in order to terminate the algorithm when the percentage of the total distortion change was equal to a predefined threshold and the number of iterations reached a predefined limit. These two quantities were selected to be 0.001 and 50, respectively. The K-means algorithm was run with this modified termination condition 100 times. The clustering result had a standard deviation of 26975.62 from the average distortion, 3869551. This represents a standard deviation of only 0.7%. It means that the data set has no well-separated regions. Therefore, experiments on this data set would not show clearly the differences between the tested algorithms.

The Clementine software package [ISL, 1998] was used to analyse further the KDD98 data set. Most of the attributes are very sparse. 80-90% of the values are 0 and the remaining have few values. Only 7 attributes (1, 11, 13, 37, 38, 40 and 46) have

reasonable distributions. However, three pairs of attributes (11 & 13, 37 & 38 and 1 & 40) have a high positive correlation. Therefore, a new data set was formed from attributes 11, 37, 40 and 46 of the original KDD98 data set and used in the tests.

The CoverType data set has 581012 objects with 54 attributes. The last 44 attributes are binary and very sparse, so they were removed from the data set and a new set formed for the tests.

Hereafter, when KDD98 and CoverType are mentioned, this will refer to the preprocessed data sets.

Five permutations of each data set were produced. Each permutation was tested 10 times. The number of clusters K was set to 10 and 50 for the KDD98 and CoverType data sets, respectively. The parameter K_1 in the Two-Phase K-means (2PK) algorithm was set equal to K for both data sets. Parameters for Bradley's algorithm had the same values as in Farnstrom's paper [Farnstrom et al., 2000].

In addition, to analyse the performance of the algorithms when they were applied on a collection of objects in a buffer that was not representative of the entire data set, the objects in the original set were rearranged. This was done by applying the original K-means algorithm on the data in order to find a relatively good clustering of the objects. Then, the clustering result was used to reorder the objects in the data set in such a way that objects belonging to the same cluster were stored close together. This could be regarded as one of the poorest cases of object groupings in the data sets. The

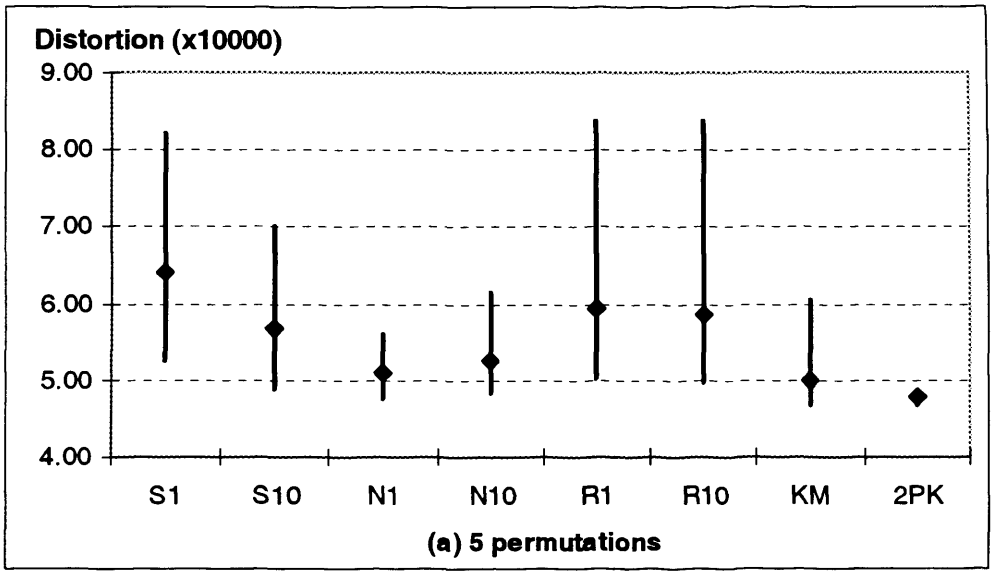
algorithms were tested on these reordered data sets under the same conditions as the five permutations.

The average cluster distortion and the average running time of all versions of the K-means algorithm when applied to the 5 permutations and the worst case of object groupings in the KDD98 and CoverType data sets are shown in Figures 5.10 to 5.13, respectively.

The average cluster distortions of the tested versions of K-means when applied to 5 permutations of the KDD98 data set are shown in Figure 5.10.a. The results of N1 and N10 (algorithms as defined in Figure 5.10 (a)) are clearly better than the results of S1 and S10, and 2PK is the most robust algorithm. 2PK consistently generates optimal solutions when applied to these permutations.

In the worst case, when objects belonging to different clusters are grouped together in the data set (Figure 5.10 (b)), the performance of R1 and R10 is affected the most. The cluster distortions for these two algorithms are much higher than those of the others. If the performances of N1 and N10 are compared against those of S1 and S10, the former are affected more. However, in general, all four versions cannot cluster the data reliably. 2PK is the most robust algorithm and the clustering results are comparable with those obtained by applying KM.

All five scaled versions of K-means show a reduction in the computational time required when compared with KM (Figure 5.11). S1 and S10 are slower than N1 and N10. 2PK is slightly slower than N1 and N10 but 12 times faster than KM.



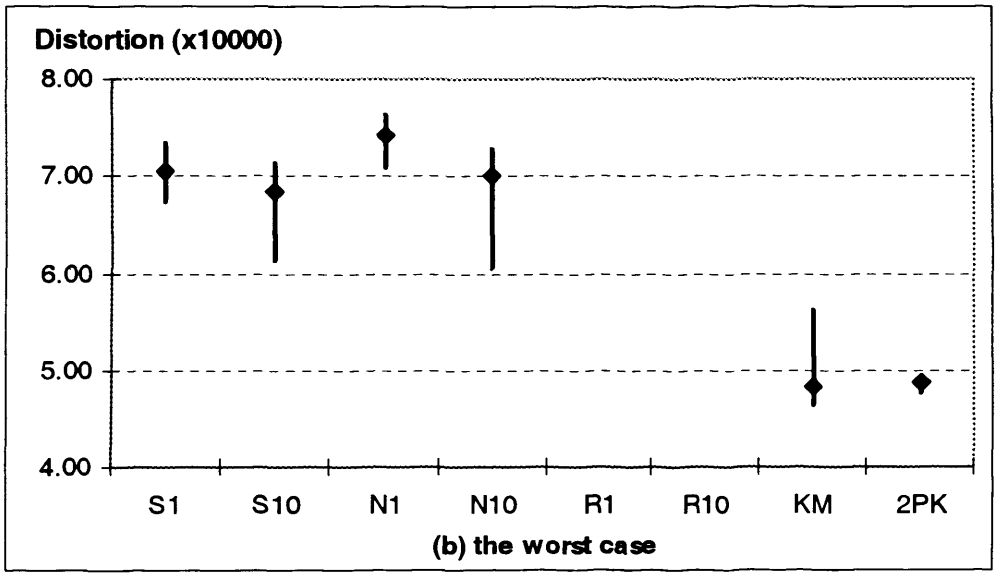


Figure 5.10 - Average cluster distortions for eight versions of K-means when applied to the KDD98 data set, (a) 5 permutations and (b) the worst case.

S1: Bradley's version of K-means with a buffer storing 1% of the data set.
S10: Bradley's version of K-means with a buffer storing 10% of the data set.
N1: Farnstrom's version of K-means with a buffer storing 1% of the data set.
N10: Farnstrom's version of K-means with a buffer storing 10% of the data set.
R1: The original K-means algorithm applied on 1% of the data set.
R10: The original K-means algorithm applied on 10% of the data set.
KM: The original K-means algorithm applied on the whole data set.
2PK: The Two-Phase K-means algorithm with a buffer storing 10% of the data set.

Note: The worst case average cluster distortions for R1 and R10 are not shown in graph (b) because they are significantly higher than those for other versions of K-means, being 53.58×10^5 and 20.45×10^5, respectively.

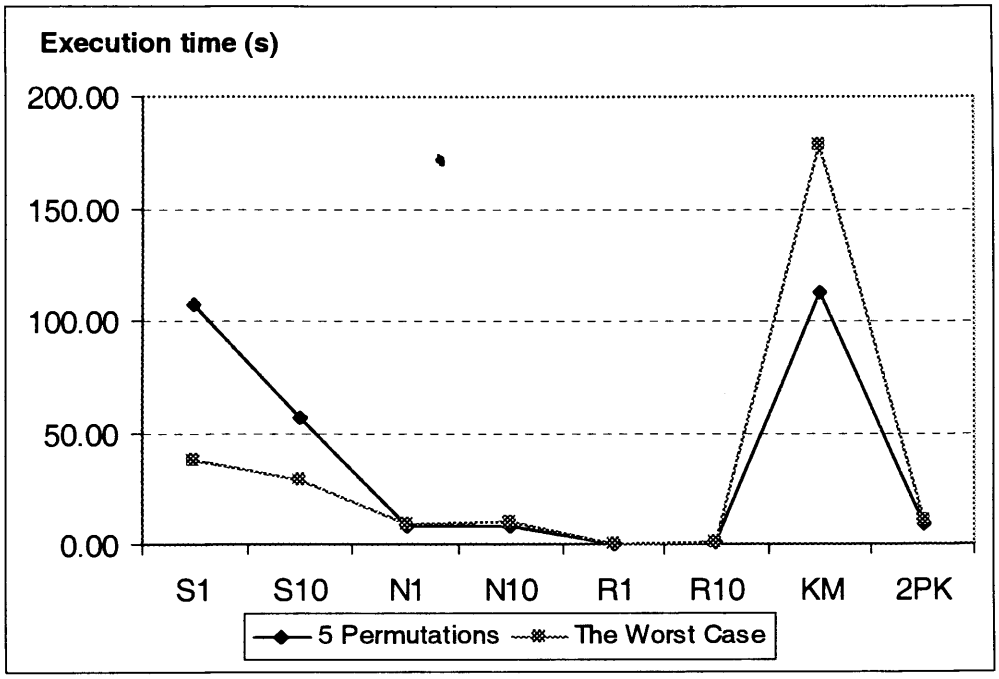


Figure 5.11 - Average execution times of tested algorithms on the KDD98 data set.

In comparing the performance of all algorithms when applied to CoverType which is a more complex data set than KDD98 and has a larger number of clusters K, the clustering results of S1 and S10 are clearly the worst (Figure 5.12.a). Surprisingly, the performance of N1 and N10 is almost the same as that of R1 and R10. At the same time, the performance of 2PK is almost the same as for KM.

In the case of the worst grouping of objects in the CoverType data set (Figure 5.12.b), the conclusions that could be made are similar to those made for the KDD data set (Figure 5.10.b). KM outperforms all other algorithms but the performance of 2PK is very close to that of KM.

S1 and S10 are the most computationally expensive algorithms as can be seen in Figure 5.13. Figure 5.13 shows that N1 and N10 are the most computationally efficient algorithms. Regarding 2PK, it is only 5 times faster than the original version KM, but generally slower when applied to the KDD98 data set. However, it is worth mentioning that the time required to execute the Incremental K-means algorithm, in the first iteration of 2PK, constitutes more than 60% of the total time. The reason for this is that K increases in steps of 1, so that the algorithm is executed approximately K times.

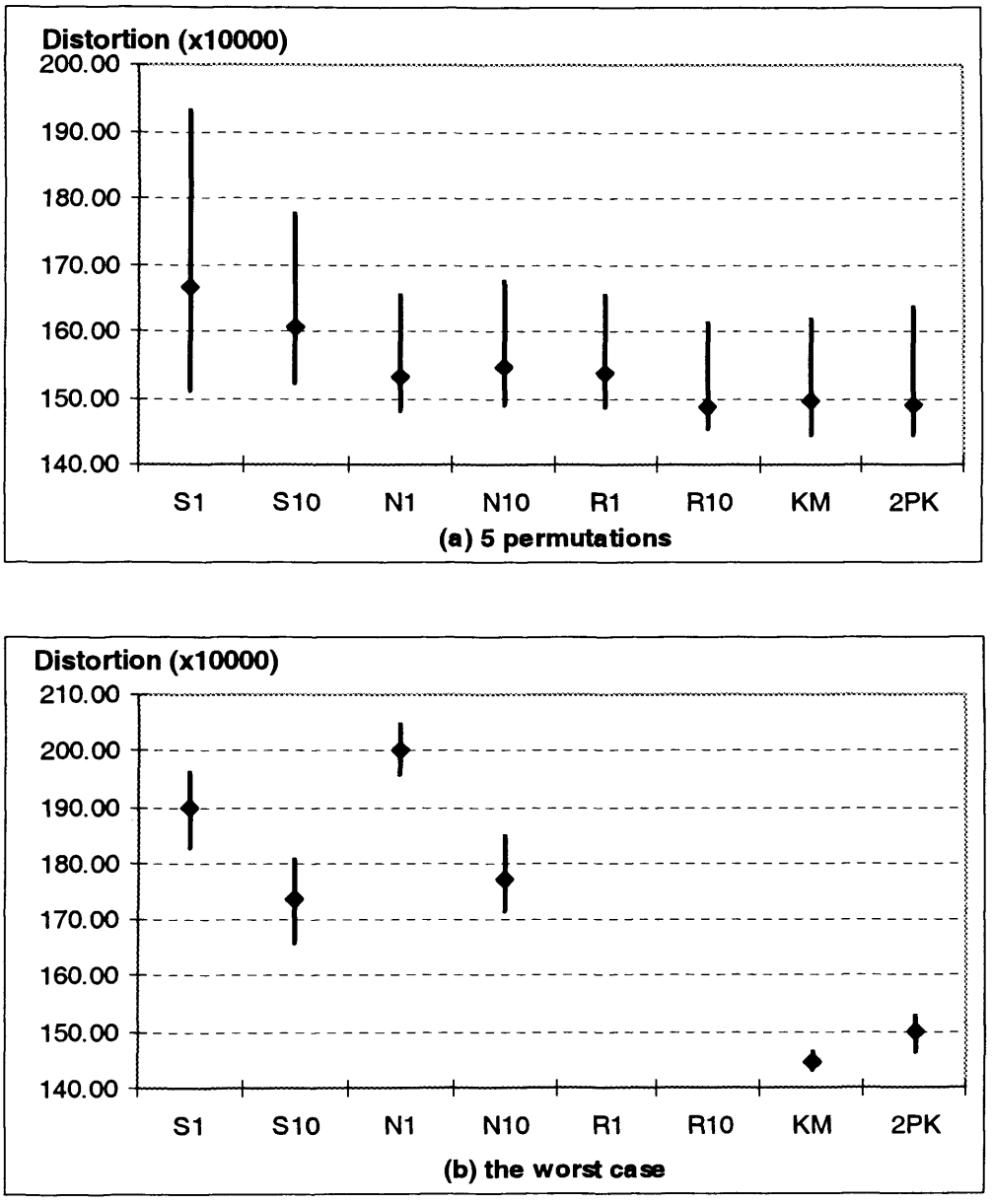


Figure 5.12 - The average cluster distortions for eight versions of K-means when applied to the CoverType data set, (a) 5 permutations and (b) the worst case.

S1: Bradley's version of K-means with a buffer storing 1% of the data set.
S10: Bradley's version of K-means with a buffer storing 10% of the data set.
N1: Farnstrom's version of K-means with a buffer storing 1% of the data set.
N10: Farnstrom's version of K-means with a buffer storing 10% of the data set.
R1: The original K-means algorithm applied on 1% of the data set.
R10: The original K-means algorithm applied on 10% of the data set.
KM: The original K-means algorithm applied on the whole data set.
2PK: The Two-Phase K-means algorithm with a buffer storing 10% of the data set.

Note: The worst case average cluster distortions for R1 and R10 are not shown in graph (b) because they are significantly higher than those for other versions of K-means, 707.37×10^5 and 314.14×10^5, respectively.

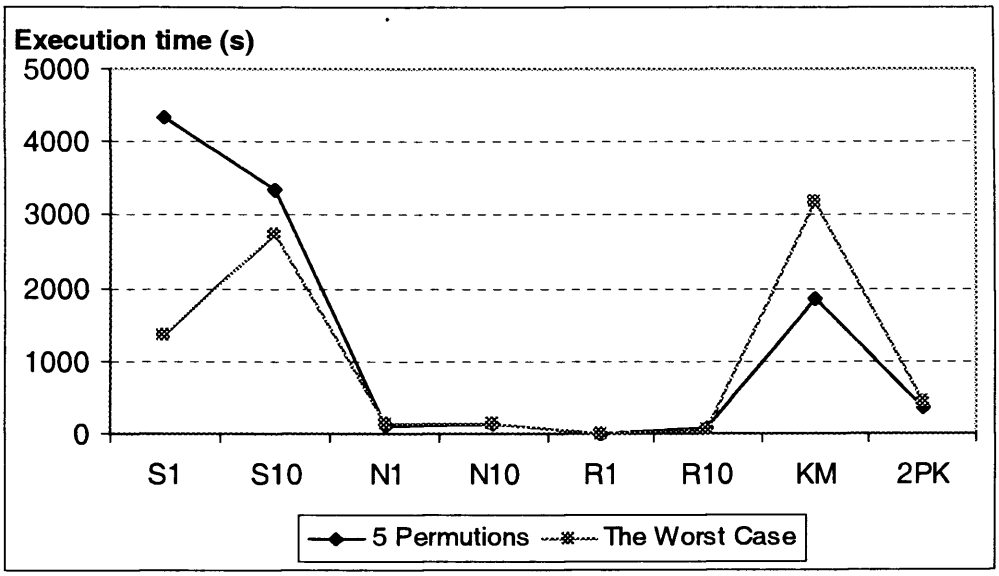


Figure 5.13 - Average execution times of tested algorithms on the CoverType data set.

5.4. Summary

This chapter has described a new clustering algorithm, Incremental K-means. The algorithm has been tested on a number of artificial and real data sets. The algorithm consistently outperforms the original K-means algorithm. The proposed search strategy decreases the dependence of the algorithm on the initialisation of cluster centres. In addition, the new algorithm only needs to be applied once to achieve almost optimal results. Further experiments will be carried out to test the new algorithm on both nominal and mixed data.

The proposed new algorithm, Two-Phase K-means, overcomes the problems associated with the application of the scaled versions of K-means on data subsets that contain collections of objects not representative of the entire data set. The algorithm consistently generates good solutions with one iteration over the data sets. It employs a simple compression strategy that is computationally more efficient than those applied in existing scaled versions of K-means.

The stepping strategy of the Incremental K-means algorithm in phase 1 of the Two-Phase K-means algorithm requires further investigation to reduce the execution time without affecting the overall performance of the algorithm.

Chapter 6

Selection of the Number of Clusters for K-means

6.1. Preliminaries

One of the deficiencies of the K-means method is the need for users to specify the number of clusters as one of the input parameters based on their domain knowledge and experience. As an exploration and analysis technique, K-means should use its clustering result regarding the data distribution to assist users in selecting the most appropriate value for this parameter. One common method for identifying this value is the use of an evaluation function based on the clustering validity.

This chapter studies existing methods for selecting the number of clusters, K, for the K-means method. Then, factors affecting the selection are analysed and an evaluation function to determine K for the K-means method is proposed.

6.2. Number of Clusters

This section reviews existing methods for selecting K for K-means and the corresponding clustering validation techniques.

6.2.1. Values of K Specified within a Range or Set

In general, to limit the effect of predefining K on the performance evaluation, a range or a set of values could be used. The values of K should vary within particular limits depending on the characteristics of the data sets. When comparing the performance of two algorithms on a given data set, the dependence between K and the data distribution may lead to results that are unfairly biased towards one method. Thus, it is important for the number of values considered to be reasonably large, to reflect the specific characteristics of the data sets and reduce this problem. At the same time, the selected values have to be significantly smaller than the number of objects in the data sets, which is the main motivation for performing data clustering.

Reported studies on K-means and its applications usually do not contain any explanation or justification for selecting particular values for K. Table 6.1 lists the numbers of clusters and objects and the corresponding data sets used in these studies. Two observations could be made when analysing the data in the table. First, a number of researchers [Bottou and Bengio, 1995; Bilmes et al., 1997; Bradley and Fayyad, 1998; Estivill-Castro and Yang, 2000] used only 1 or 2 values for K. Second, several other researchers [Al-Daoud et al., 1995; Al-Daoud et al., 1996; Hansen and Larsen, 1996; Fritzke, 1997; Pelleg and Moore, 1999] utilised relatively large K values compared with the number of objects. These two actions contravene the above-mentioned guidelines for selecting K. Therefore, the clustering results do not always correctly represent the clustering performance of the tested algorithms.

Table 6.1 - The number of clusters used in different studies of the K-means method.

Studies	Numbers of clusters *K*	Number of objects *n*	Max ratio *K/n* (%)
[Al-Daoud, et al, 1995]	32, 64, 128, 256, 512, 1024	8192	12.50
	32, 64, 128, 256, 512, 1024	29000	
	256	2048	
[Al-Daoud, et al, 1996]	600, 700, 800, 900, 1000	10000	10.00
	600, 700, 800, 900, 1000	50000	
[Alsabti, et al, 1998]	4, 16, 64, 100, 128	100000	0.13
	4, 16, 64, 100, 128	120000	
	4, 16, 64, 100, 128	256000	
[Bilmes et al., 1997]	4	564	0.70
	4	720	
	4	1000	
	4	1008	
	4	1010	
	4	1202	
	4	2000	
	4	2324	
	4	3005	
	4	4000	
	4	6272	
	4	7561	

Table 6.1 (continued)

Studies	Numbers of clusters K	Number of objects n	Max ratio K/n (%)
[Bottou and Bengio, 1995]	6	150	4.00
[Bradley and Fayyad, 1998]	10	2310	0.43
	25	12902	
[Du and Wong, 2002]	2, 4, 8	Not reported	Not reported
[Estivill-Castro and Yang, 1996]	2, 4	500	3.33
	2, 4	50000	
	2, 4	100000	
	10	300	
[Estivill-Castro, 2002]	1, 2, 3, 4	10000	0.04
[Fritzke, 1997]	10, 20, 30, 40, 50, 60, 70, 80, 90, 100	500	20.00
[Hamerly and Elkan, 2002]	100	10000	2.00
	50	2500	
[Hansen and Larsen, 1996]	7	42	16.66
	1, 2, 3, 4, 5, 6, 7	120	
[Ishioka, 2000]	2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14	250	5.60
[Kanungo, et al, 1999]	8, 20, 50, 64, 256	10000	2.56

Table 6.1 (continued)

Studies	Numbers of clusters K	Number of objects n	Max ratio K/n (%)
[Pelleg and Moore, 1999]	5000	50000	50.00
	5000	100000	
	5000	200000	
	5000	300000	
	5000	433208	
	100	100000	
	250	200000	
	1000	100000	
	1000	200000	
	1000	300000	
	1000	433208	
	40	20000	
	10, 20, 30, 40, 50, 60, 70, 80	30000	
	50, 500, 5000	10000	
	50, 500, 5000	50000	
	50, 500, 5000	100000	
	50, 500, 5000	200000	
	50, 500, 5000	300000	
	50, 500, 5000	433208	

Table 6.1 (continued)

Studies	Numbers of clusters K	Number of objects n	Max ratio K/n (%)
[Pelleg and Moore, 2000]	250	80000	10.00
	250	90000	
	250	100000	
	250	110000	
	250	120000	
	50, 100, 400	4000	
	50, 100, 400	36000	
	250	80000	
	250	90000	
	250	100000	
	250	110000	
	250	120000	
	50, 100, 150	4000	
	50, 100, 150	36000	
	50	800000	
	500	800000	
[Pena et al., 1999]	3, 4	150	6.67
	4, 5	75	
	2, 7, 10	214	

In general, the performance of any new version of the K-means algorithm could be verified by comparing it with its predecessors on the same criteria. In particular, the sum of cluster distortions is usually employed as such a performance indicator [Bottou and Bengio, 1995; Al-Daoud and Roberts, 1996; Hansen and Larsen, 1996; Pelleg and Moore, 1999; Pena et al., 1999]. Thus, the comparison is considered fair because the same model and criterion are used for the performance analysis.

6.2.2. Values of *K* Specified by the User

The K-means implementation in Data Mining or Data Analysis software packages, such as *Clementine* [ISL, 1998], *Data Engine* [MIT, 1998], *SPSS* [Kerr et al., 2002], *S-PLUS* [Insightful Corporation, 2001], requires the number of clusters to be specified by the user. To find a satisfactory clustering result, usually, a number of iterations are needed where the user executes the algorithm with different values of K. The validity of the clustering result is assessed only visually without applying any formal performance measures. With this approach, it is difficult for users to evaluate the clustering result for multi-dimensional data sets.

6.2.3. Values of *K* Determined in a Later Processing Step

When K-means is used as a pre-processing tool as shown in Figure 6.1, the number of clusters is determined by the specific requirements of the main processing algorithm [Hansen and Larsen, 1996]. No attention is paid to the effect of the clustering results

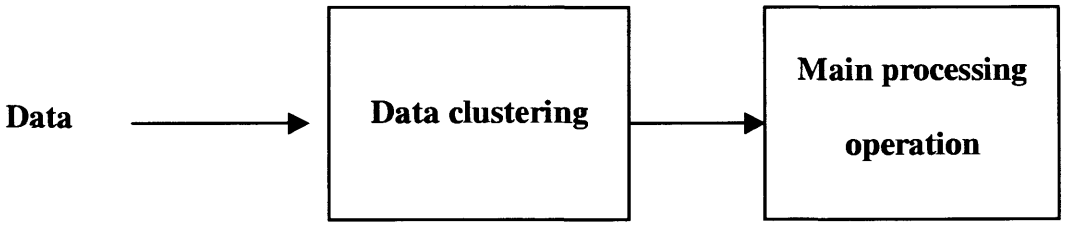


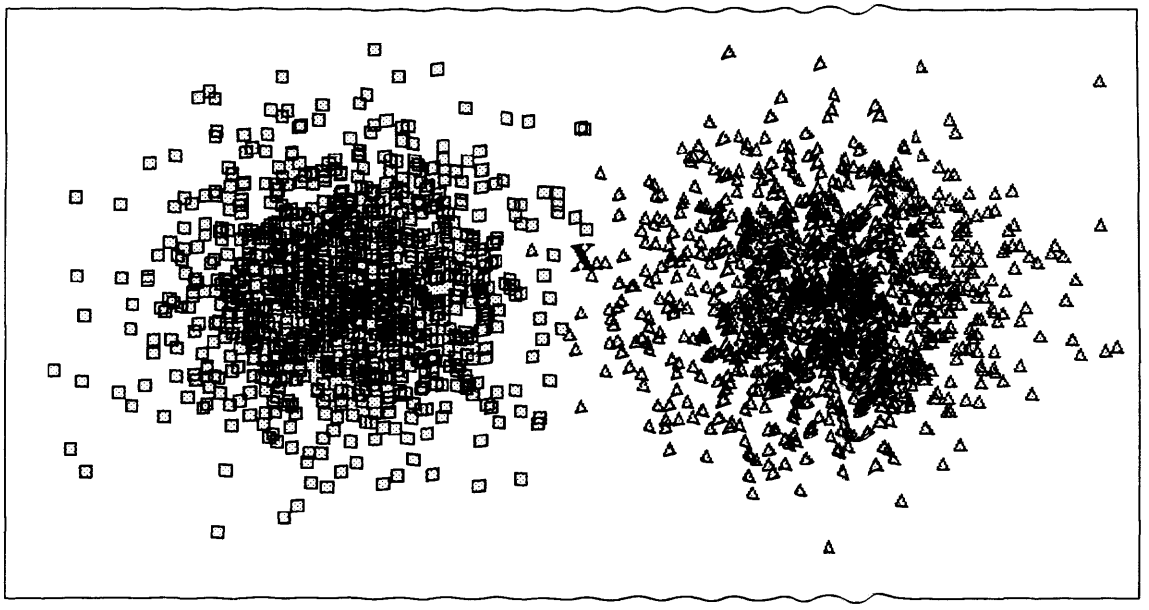
Figure 6.1 – Data clustering as a pre-processing tool.

on the performance of this algorithm. In such applications, K-means is used just as a "black box" without validation of the clustering result.

6.2.4. Values of *K* Equated to Number of Generators

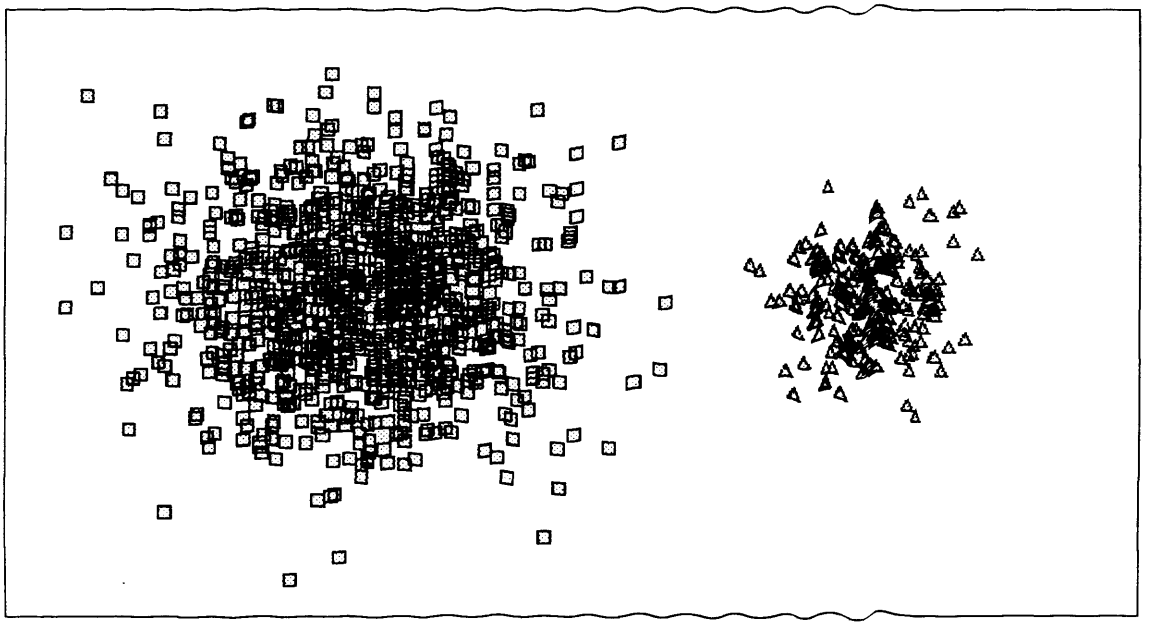
Synthetic data sets, which are used in testing algorithms, are often created by a set of normal or uniform distribution generators. Then, clustering algorithms are applied to those data sets with the number of clusters equated to the number of generators. It is assumed that any resultant cluster will cover all objects created by a particular generator. Thus, the clustering performance is judged based on the difference between objects covered by a cluster and those created by the corresponding generator. Such a difference can be measured by simply counting objects or calculating the Information Gain [Bradley and Fayyad, 1998].

There are a few deficiencies with this method. The first deficiency concerns the stability of the clustering results when there are areas in the object space that contain objects created by different generators. Figure 6.2.a illustrates such a case. The data set shown in this figure has two clusters, A and B, which cover objects generated by generators G_A and G_B respectively. Object X is in an overlapping area between clusters A and B. X has probabilities P_{G_A} and P_{G_B} of being created by G_A and G_B, respectively, and probabilities P_{C_A} and P_{C_B} of being included into clusters A and B, respectively. All four probabilities are larger than 0. Thus, there is a chance for X to be created by generator G_A but covered by cluster B, and vice versa. In such cases,



Generator G_A / Cluster A Generator G_B / Cluster B

(a)



Generator G_A / Cluster A Generator G_B / Cluster B

(b)

Figure 6.2 – The relationship between clusters can have an effect on the clustering. Two object spaces: (a) an area exists that contains objects created by two different generators (b) no overlapping areas.

Note: □ - objects generated by G_A and Δ - objects generated by G_B

the clustering results will not be perfect. The stability of the clustering results depends on these four probabilities. With an increase in the overlapping areas in the object space, the stability of the clustering results decreases.

The difference between the characteristics of the generators also has an effect on the clustering results. In Figure 6.2 (b) where the number of objects of cluster *A* is five times larger than that of cluster *B*, the smaller cluster *B* might be regarded as noise and all objects might be grouped into one cluster. Such a clustering outcome would disagree with that obtained by visual inspection.

Unfortunately, this method of selecting *K* cannot be applied to practical problems. The data distribution in practical problems is unknown and also the number of generators cannot be specified.

6.2.5. Values of *K* Determined by Statistical Measures

There are several statistical measures available for selecting *K*. These measures are often applied in combination with probabilistic clustering approaches. They are calculated with certain assumptions about the underlying distribution of the data. The Bayesian Information Criterion (BIC) or Akeike's Information Criterion (AIC) [Pelleg and Moore, 2000; Ishioka, 2000] are calculated on data sets which are constructed by a set of Gaussian distributions. The measures applied by Hardy [Hardy, 1996] are based on the assumption that the data set fits the Poisson distribution. Monte-Carlo techniques, which are associated with the *Null Hypothesis*,

are used for assessing the clustering results and also for determining the number of clusters [Theodoridis and Koutroubas, 1999; Halkidi et al., 2002].

There have been comparisons between probabilistic and partitioning clustering [Bradley and Fayyad, 1998]. Expectation-Maximisation (EM) is often recognised as a typical method for probabilistic clustering. Similarly, K-means is considered a typical method for partitioning clustering. Although, EM and K-means share some common ideas, they are based on different hypotheses, models and criteria. Probabilistic clustering methods do not take into account the distortion inside a cluster, so that a cluster created by applying such methods may not correspond to a cluster in partitioning clustering, and vice versa. Therefore, statistical measures used in probabilistic methods are not applicable in the K-means method. In addition, the assumptions about the underlying distribution cannot be verified on real data sets and therefore cannot be used to obtain statistical measures.

6.2.6. Values of *K* Equated to the Number of Classes

With this method, the number of clusters is equated to the number of classes in the data sets. A data clustering algorithm can be used as a classifier by applying it to data sets from which the class attribute is omitted and then the clustering results are assessed using the omitted class information [Kotari and Pitts, 1999; Cai, 2001]. The outcome of the assessment is fedback to the clustering algorithm to improve its performance. In this way, the clustering can be considered supervised.

With this method of determining the number of clusters, the assumption is made that the data clustering method could form clusters, each of which would consist of only objects belonging to one class. Unfortunately, most real problems do not satisfy this assumption.

6.2.7. Values of *K* Determined through Visualisation

Visual verification is applied widely because of its simplicity and explanation capabilities. Visual examples are often used to illustrate the drawbacks of an algorithm or present the expected clustering results [Bilmes et al., 1997; Cai, 2001].

The assessment of a clustering result using visualisation techniques depends heavily on their implicit nature. The clustering models utilised by some clustering methods may not be appropriate for particular data sets. The data sets in Figure 6.3 are illustrations of such cases. The application of visualisation techniques implies a data distribution continuity in the expected clusters. If K-means is applied to such data sets, there is not any cluster that could satisfy the clustering model of K-means and at the same time corresponds to a particular object grouping in the illustrated data sets. Therefore, K-means cannot produce the expected clustering results. This suggests that the K-means approach is unsuitable for such data sets.

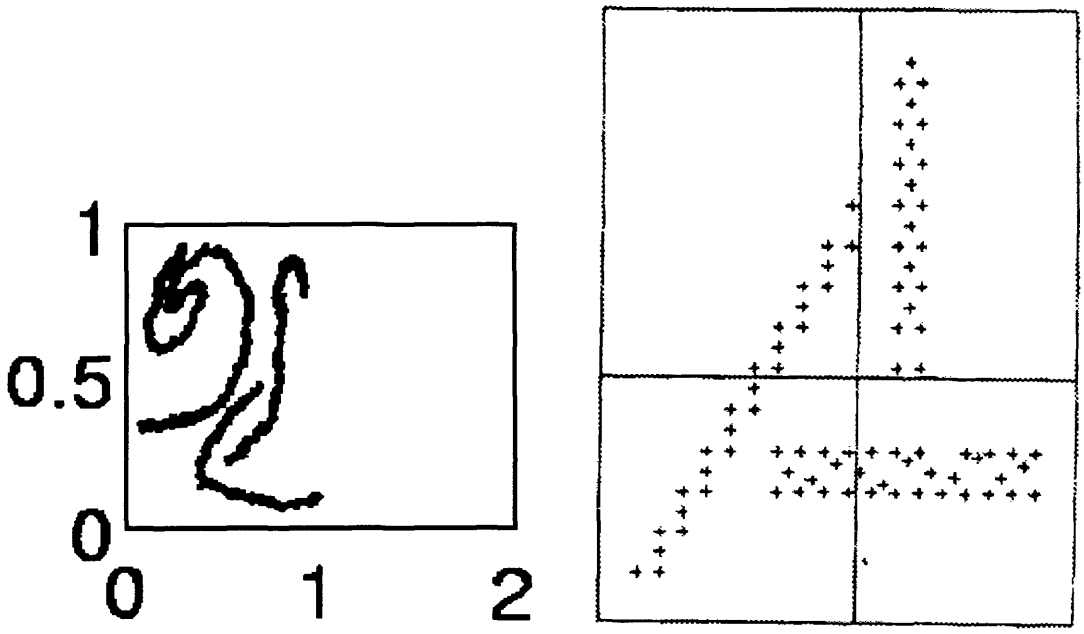
The characteristics of the data sets in Figure 6.3 (position, shape, size and object distribution) are implicitly defined. This makes the validation of the clustering results difficult. Any slight changes in the data characteristics may lead to different

outcomes. The data set in Figure 6.3 (b) is an illustration of such a case. Another example is the series of data sets in Figure 6.4. Although two clusters are easily identifiable in the data set in Figure 6.4 (a), the numbers of clusters in the data sets in Figures 6.4 (b) and 6.4 (c) depend on the distance between the rings and the object density of each ring. Usually such parameters are not explicitly defined when a visual test is carried out.

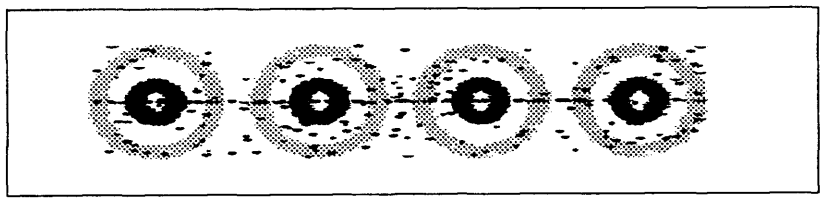
In spite of the above-mentioned deficiencies, visualisation of the results is still a useful method of selecting K and validating the clustering results when the data sets do not violate the assumptions of the clustering model. In addition, this method is recommended in cases where the expected results could be identified explicitly.

6.2.8. Values of K Determined Using a Neighbourhood Measure

A neighbourhood measure could be added to the cost function of K-means to determine K [Kothari and Pitts, 1999]. Although this technique has showed promising results for a few data sets, it needs to prove its potential in practical applications. Because the cost function has to be modified, this technique cannot be applied to the original K-means method.



(a) (b)



(c)

Figure 6.3 – Inappropriate data sets for the K-means approach.

Data sets with: (a) 4 clusters [Bilmes et al., 1997]; (b) 3 clusters [Hardy, 1996];

(c) 8 clusters [Cai, 2001]

Note: The number of clusters in each data set was specified by the respective authors.

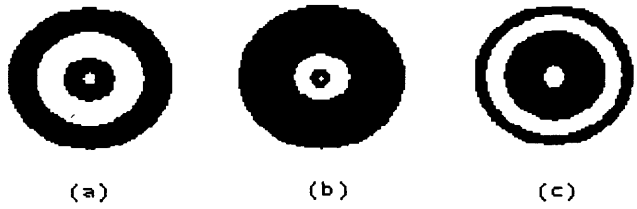


Figure 6.4 – Variations of the two-ring data set.

6.3. Factors Affecting the Selection of *K*

A function *f(K)* for evaluating the clustering result could be used to select the number of clusters. Factors that such a function should take into account are discussed in this section.

6.3.1. Approach Bias

The evaluation function should be related closely to the clustering criteria. As mentioned previously, such a relation could prevent adverse effects on the validation process. In particular, in the K-means method, the criterion is the minimisation of the distortion of clusters, so that the evaluation function should take this parameter into account.

6.3.2. Level of Detail

From the scale-space theory in vision research [Lindberg, 1994], observers that could see relatively low levels of detail would get only a general view of an object. By increasing the level of detail, they could obtain *more knowledge* about the observed object but at the same time, the amount of information that they have to process increases significantly. Because of resource limitations, a high level of detail is used only to examine parts of the object.

Such an approach could be applied in clustering. A data set with n objects could be grouped into any number of clusters between 1 and n, which would correspond to the lowest and the highest level of detail, respectively. By specifying different K values, it is possible to assess the results of grouping objects into various numbers of clusters. From this evaluation, more than one K value could be recommended to users, but the final selection is made by them.

6.3.3. Internal Distribution versus Global Impact

Clustering is used to find irregularities in the data distribution and identify regions in which objects are concentrated. However, not every region with a high concentration of objects is considered a cluster. For a region to be identified as a cluster, it is important to analyse not only its internal distribution, but also its interdependency with other object groupings in the data set.

In K-means, the distortion of a cluster is a function of the data population and the distance between objects and the cluster centre. Each cluster is represented by its distortion and its impact on the entire data set is assessed by its contribution to the sum of all distortions. Thus, such information is important in assessing whether a particular region in the object space could be considered a cluster.

6.3.4. Constraints for *f(K)*

The robustness of *f(K)* is very important. Because the performance of this function is judged based on the result of the clustering algorithm, it is important for this result to vary as little as possible when *K* remains unchanged. One of the main deficiencies of K-means is its dependence on randomness. Thus, the algorithm should yield consistent results so that its performance can be used as a variable in the evaluation function. One of the versions of K-means, the Incremental K-means method, which was presented in chapter 5, satisfies this requirement so that a measure for assessing its performance can be used as a variable in *f(K)*.

The role of *f(K)* is to identify trends in the data distribution and therefore it is important to keep it independent of the number of objects. The number of clusters *K* is assumed to be much smaller than the number of objects *N*. When *K* increases, the robustness of *f(K)* should increase also. If a minimum or maximum value of *f(K)* is used to select *K*, such a value of *K* could be considered optimum due to the function robustness at stationary points.

6.4. Number of Clusters for K-means

As mentioned previously, cluster analysis is used to find irregularities in the data distribution. When the data distribution is uniform, there is not any irregularity. Therefore, data sets with uniform distribution could be used to calibrate and verify the clustering performance. This approach was applied by Tibshirani [Tibshirani et al.,

2000]. A data set of the same dimension as the actual data set and with a uniform distribution was generated. The clustering performance on this artificial data set was then compared with the result obtained for the actual data set. A measure known as the "*Gap*" statistic [Tibshirani et al., 2000] was employed to assess performance. In this work, instead of generating an artificial data set, the clustering performance for the artificial data set was estimated. Also, instead of the *Gap* statistic, a new and more discriminatory measure was employed for evaluating the clustering result in this chapter.

When the K-means algorithm is applied to data with a uniform distribution and K is increased by 1, the clusters are likely to change and in the new positions, the partitions will be again approximately equal in size and their distortions similar to one another. The evaluations carried out in chapter 5 showed that when a new cluster is inserted into a cluster (K=1) with a hyper-cuboid shape and a uniform distribution, the decrease in the sum of distortions is proportional to the original sum of distortions. This conclusion was found to be correct for clustering results obtained with relatively small values of K. In such a case, the sum of distortions after the increase in the number of clusters could be estimated from the current value.

The evaluation function $f(K)$ is defined using Equations 6.1 and 6.2, where S_K is the sum of the cluster distortions when the number of clusters is K, N_d is the number of data set attributes and α_K is a weight factor. The term $\alpha_K S_{K-1}$ in Equation 6.1 is an estimation of S_K based on S_{K-1} made with the assumption that the data has a uniform distribution. The value of $f(K)$ is the ratio of the real and estimated distortions and is close to 1 when the data distribution is uniform. When there are areas of concentration

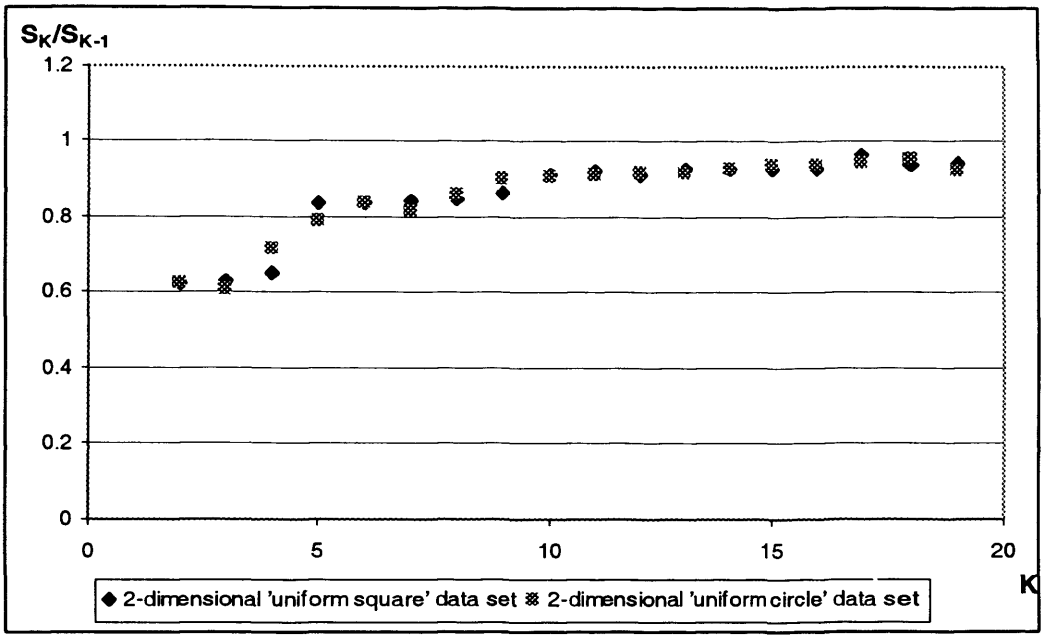
in the data distribution, S_K will be less than the estimated value, so that $f(K)$ decreases. The smaller $f(K)$ is, the more concentrated the data distribution. Thus, values of K that yield small $f(K)$ can be regarded as giving well-defined clusters.

The weight factor α_K, given in Equation 6.2, is a positive number less than or equal to 1, and is applied to reduce the effect of dimensions. With K=2, α_K is computed using Equation 6.2 (a). This equation is derived from Equation 5.7 which shows that the decrease in distortion is inversely proportional to the number of dimensions N_d (see Section 5.2.3).

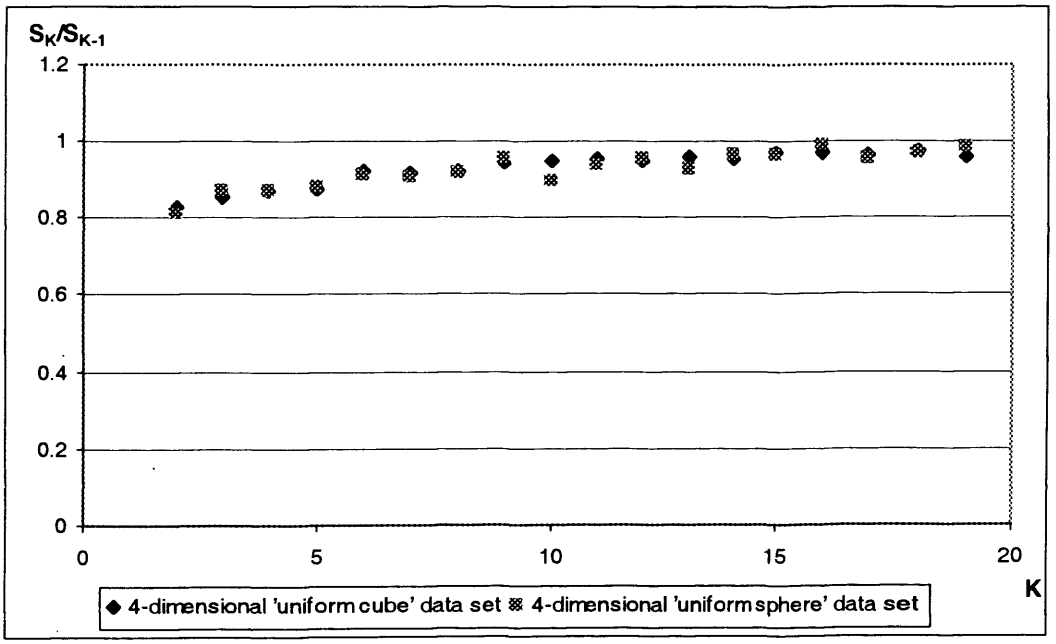
As K increases above 2, the decrease in the sum of distortions reduces (the ratio S_K/S_{K-1} approaches 1) as can be seen in Figure 6.5. This figure shows the values of S_K/S_{K-1} computed for different K when the clustering algorithm is applied to data sets of different dimensions and with uniform distributions. With such data sets, $f(K)$ is expected to be equal to 1 and α_K should be chosen to equate $f(K)$ to 1. From Equation 6.1, α_K should therefore be S_K/S_{K-1} and thus obtainable from Figure 6.5. However, for computational simplicity, the recursion Equation 6.2 (b) has been derived from the data represented in Figure 6.5 to calculate α_K. Figure 6.6 shows that the values of α_K obtained from Equation 6.2 (b) fit the plots in Figure 6.5 closely.

$$f(K) = \begin{cases} 1 & \text{if } K = 1 \\ \dfrac{S_K}{\alpha_K S_{K-1}} & \text{if } S_{K-1} \neq 0, \forall K > 1 \\ 1 & \text{if } S_{K-1} = 0, \forall K > 1 \end{cases} \tag{6.1}$$

$$\alpha_K = \begin{cases} \left(1 - \dfrac{3}{4N_d}\right) & \text{if } K = 2 \text{ and } N_d > 1 \quad (a) \\ \alpha_{K-1} + \dfrac{1-\alpha_{K-1}}{6} & \text{if } K > 2 \text{ and } N_d > 1 \quad (b) \end{cases} \tag{6.2}$$



(a)

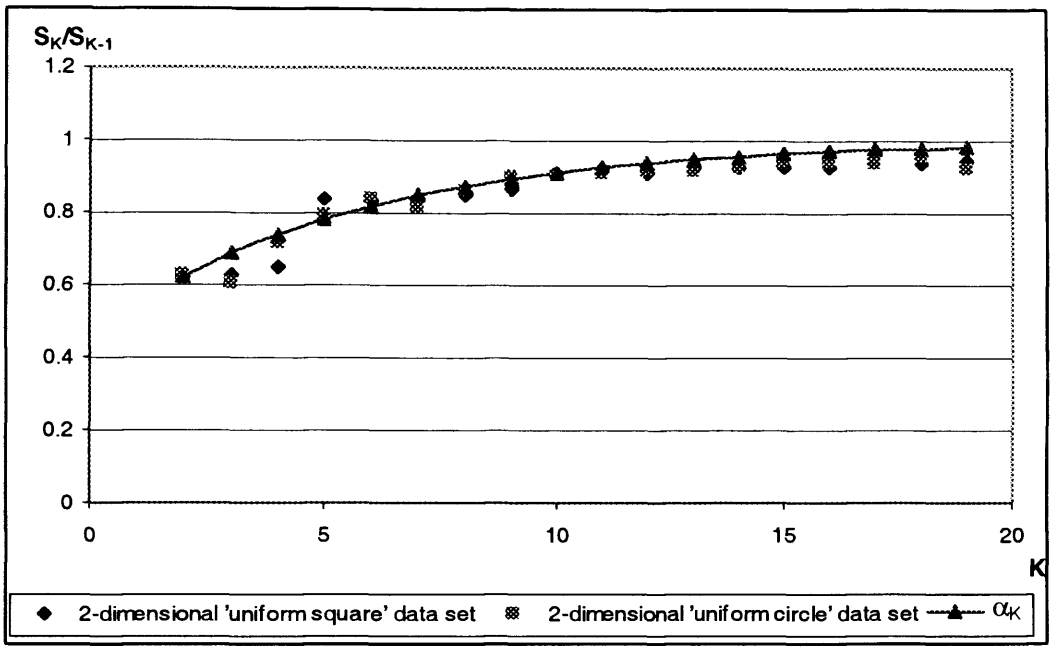


(b)

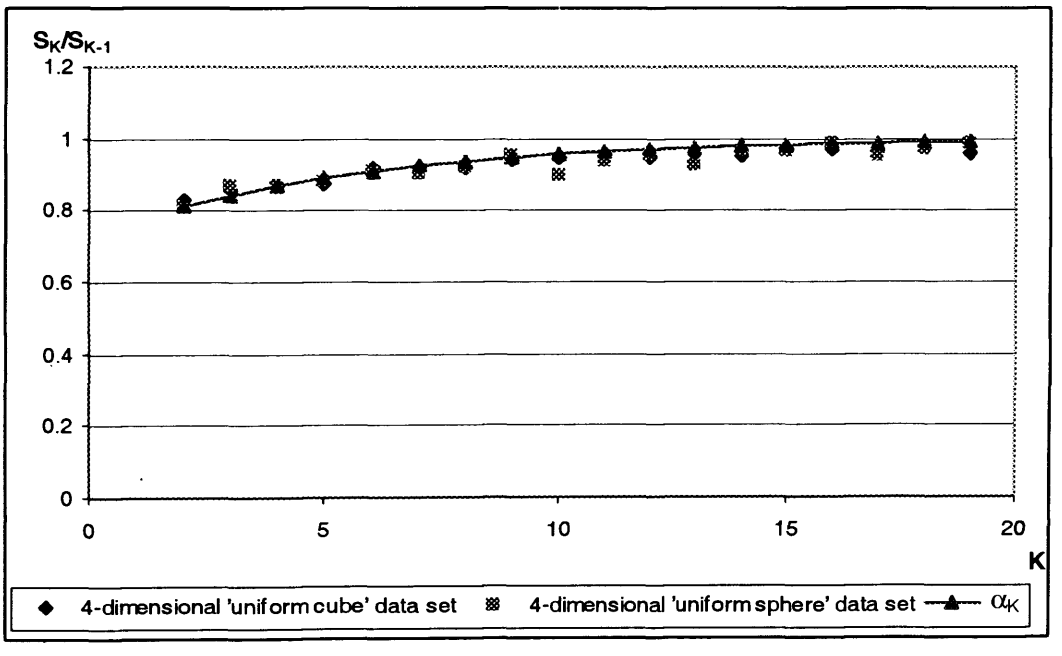
Figure 6.5 – The ratio S_K/S_{K-1} for data sets having uniform distributions

(a) 2-dimensional 'square' and 'circle', and

(b) 4-dimensional 'cube' and 'sphere'.



(a)



(b)

Figure 6.6 – Comparison of the values of α_K calculated using Equation 6.2 (b) and the ratio S_K/S_{K-1}.

The proposed function $f(K)$ satisfies the constraints mentioned in the previous section. The robustness of $f(K)$ will be verified experimentally in the next section. When the number of objects is doubled or tripled but their distributions are unchanged, the resultant clusters remain in the same position, S_K and S_{K-1} are doubled or tripled correspondingly, so that $f(K)$ stays constant. Generally, $f(K)$ is independent of the number of objects of the data.

To reduce the effect of the differences in the ranges of the attributes, data is standardised or normalised before the clustering starts. However, it should be noted that, when the data has well-separated groups of objects, the shape of such regions in the problem space has an effect on the evaluation function. In such cases, the scaling techniques do not influence the local object distribution, because scaling applies to the whole data set.

6.5. Performance

The evaluation function $f(K)$ is tested by a series of experiments on the artificially generated data sets shown in Figure 6.7. All data is standardised before the Incremental K-means method is applied with K ranging from 1 to 19. $f(K)$ is calculated based on the total distortion of the clusters.

In Figures 6.7 (a), 6.7 (b) and 6.7 (c), all objects belong to a single region with a uniform distribution. The graph in Figure 6.7 (a) shows that $f(K)$ reflects well the clustering result on this data set with a uniform distribution because $f(K)$ is

approximately constant and equal to 1 for all K. When K=4 and K=3 in Figures 6.7 (a) and 6.7 (b), respectively, $f(K)$ reaches minimum values. This could be attributed to the shape of the areas defined by the objects belonging to these data sets. However, the minimum values of $f(K)$ do not differ significantly from the average value for any strong recommendations to be made to the user. By comparing the values of $f(K)$ in Figures 6.7 (a) and 6.7 (c), it can be seen that α_K reduces the effect of the data set dimensions on the evaluation function.

For the data set in Figure 6.7 (d), again, all objects are concentrated in a single region with a normal distribution. The $f(K)$ plot for this data set suggests correctly that when K = 1, the clustering result is the most suitable for this data set.

The data sets in Figures 6.7 (e) and 6.7 (f) are created by 2 generators having normal distributions. In Figure 6.7 (e), the two generators have an overlapping region but in Figure 6.7 (f), they are well separated. Note that the value for $f(2)$ in the latter figure is much smaller than that in the former.

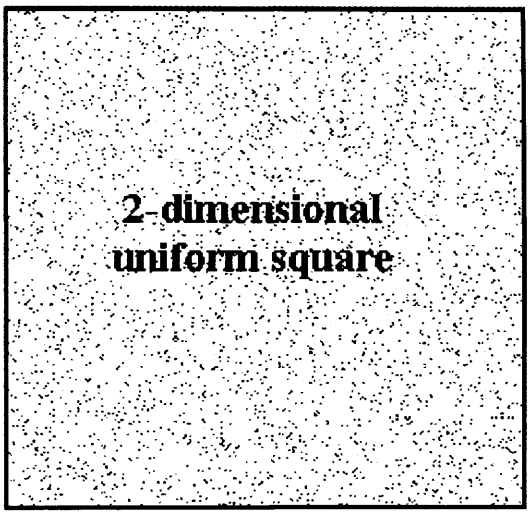
The data sets in Figures 6.7 (g) and 6.7 (h) have three recognisable regions. From the corresponding graphs, $f(K)$ suggests correct values of K for clustering these data sets.

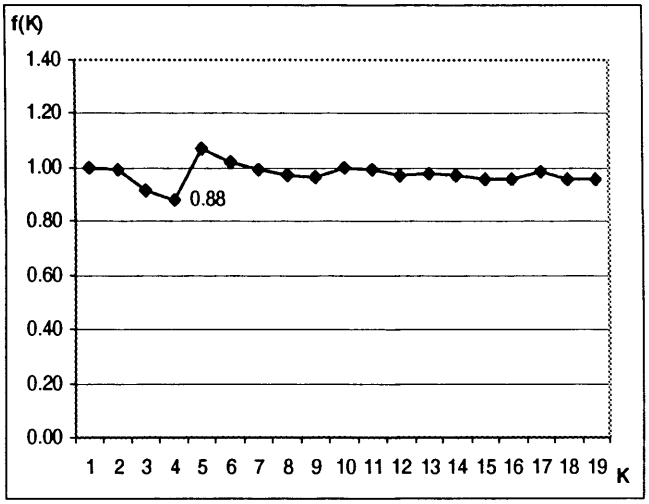
Three different generators that create object groupings with a normal distribution are used to form the data set in Figure 6.7 (i). In this case, $f(K)$ suggests the values 2 or 3, for K. Because two of these three generators create object groupings that overlap, $f(2)$ is smaller than $f(3)$. This means that the data has only two clearly defined regions, but K=3 could also be used to cluster the objects.

Figures 6.7 (j) and 6.7 (k) illustrate how the level of detail could affect the selection of K. $f(K)$ reaches minimum values at K= 2 and 4 respectively. In such cases, users could select the most appropriate value of K based on their specific requirements. A more complex case is shown in Figure 6.7 (l) where there are possible K values of 4 or 8. The selection of a particular K will depend on the requirements of the specific application for which the clustering is carried out.

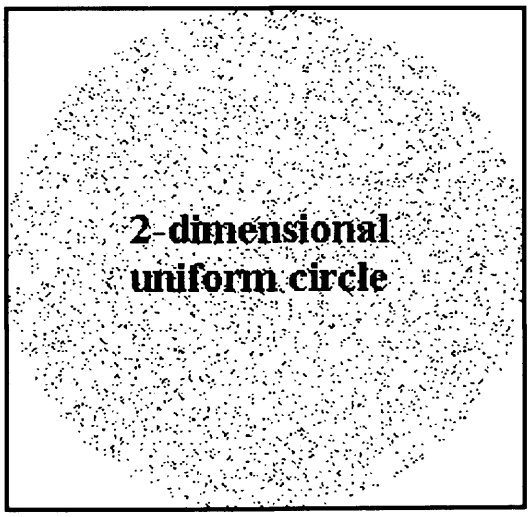
The data sets in Figures 6.7 (m), 6.7 (n) and 6.7 (o) have well-defined regions in the object space, each of which has a different distribution, location and number of objects. If the minimum value of $f(K)$ is used to cluster the objects, K will be different from the number of generators utilised to create them or the number of object groupings that could be identified visually. Therefore, from analysing $f(K)$, only recommendations could be made and the decision as to which particular value should be adopted has to be taken by the user.

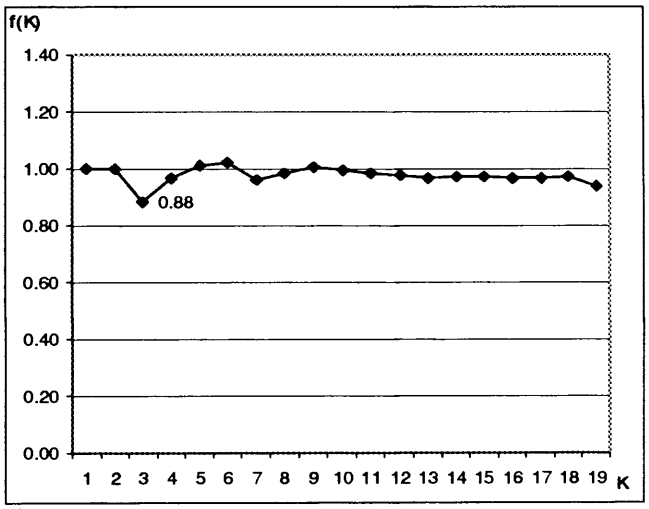
From the graphs in Figures 6.7, a conclusion could be made that any K with corresponding $f(K) < 0.85$ could be recommended for clustering. If there is not a value with corresponding $f(K) < 0.85$, K=1 is selected.



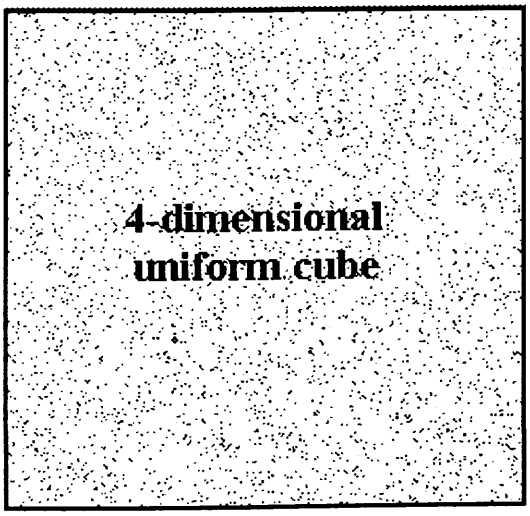


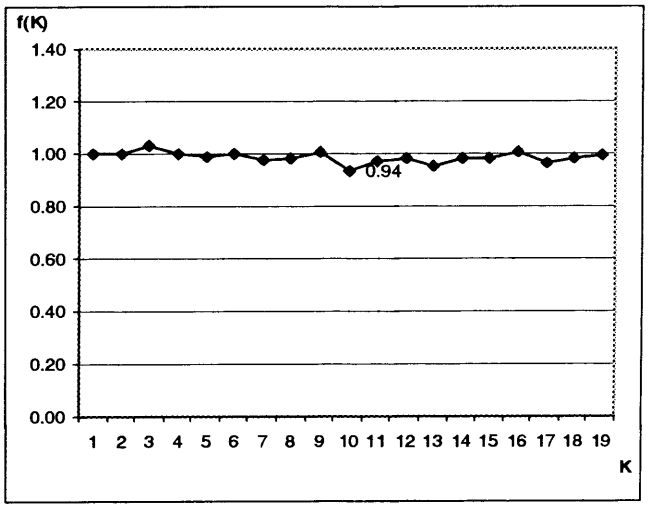
(a)





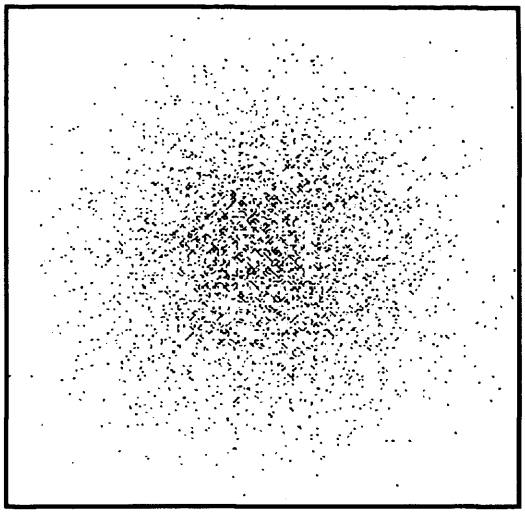
(b)

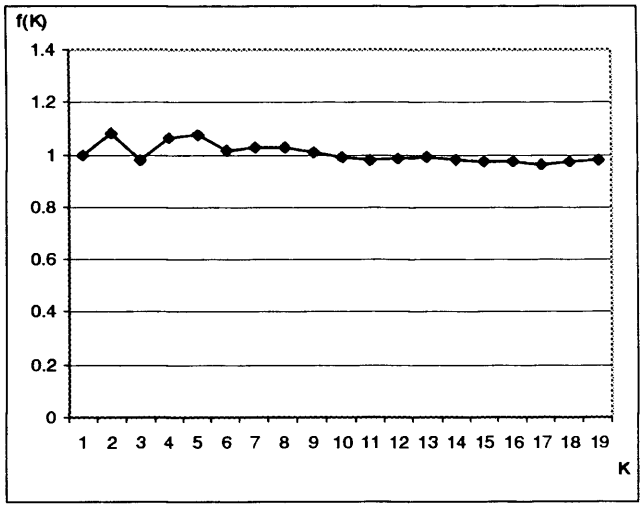




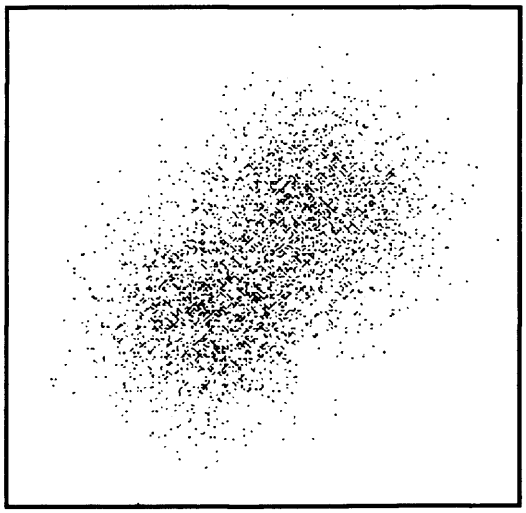
(c)

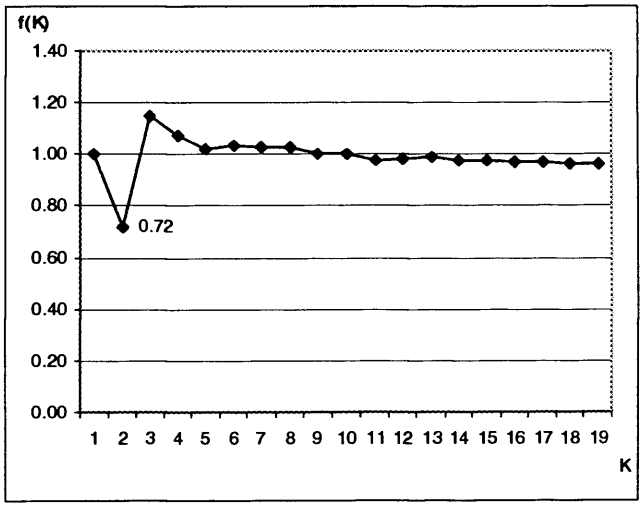
Figure 6.7 – Data sets and their corresponding *f(K)*.



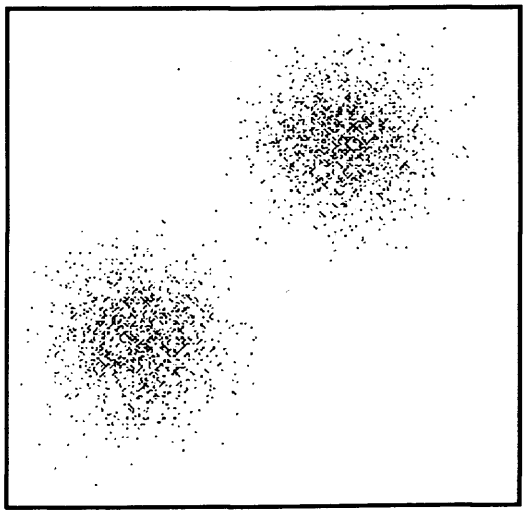


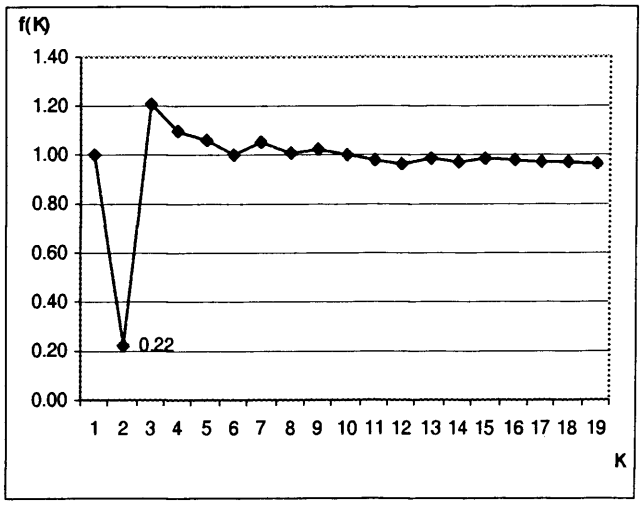
(d)





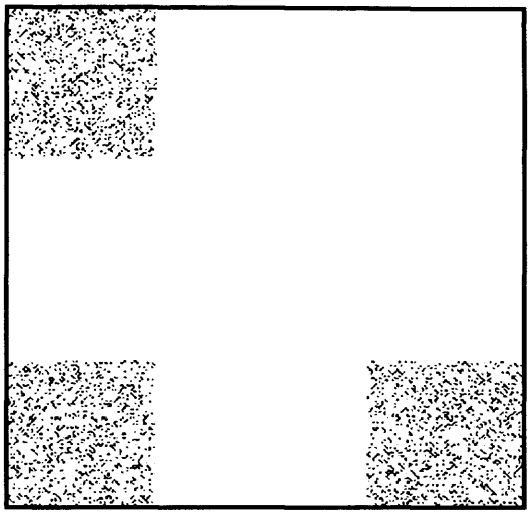
(e)

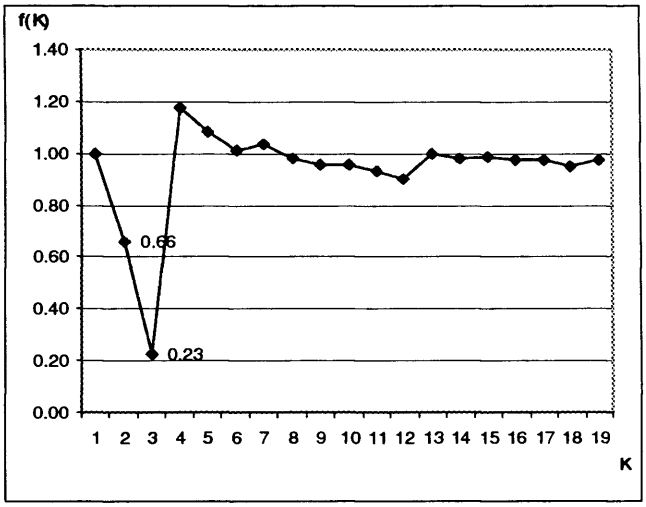




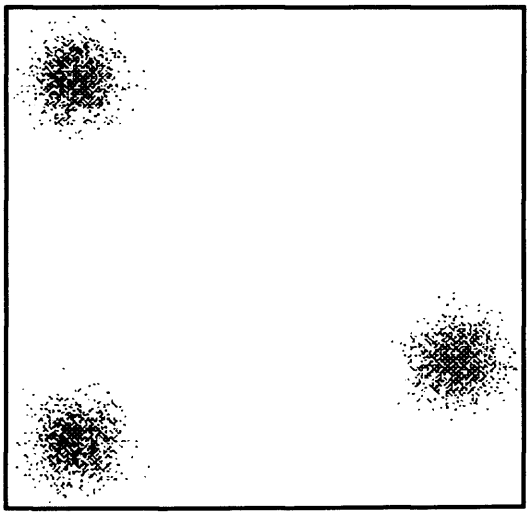
(f)

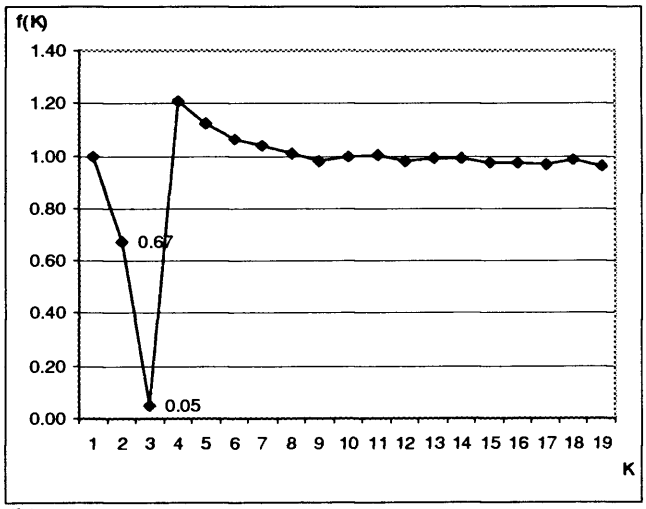
Figure 6.7 (continued)



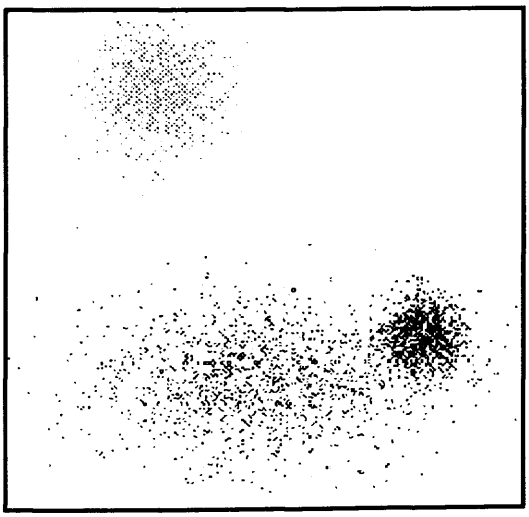


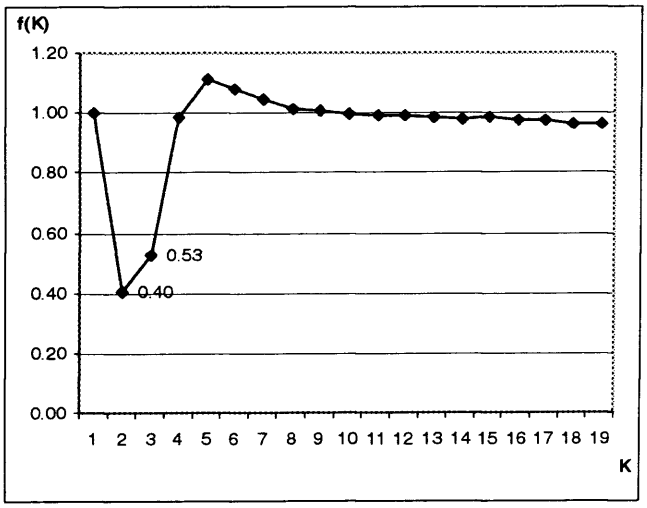
(g)





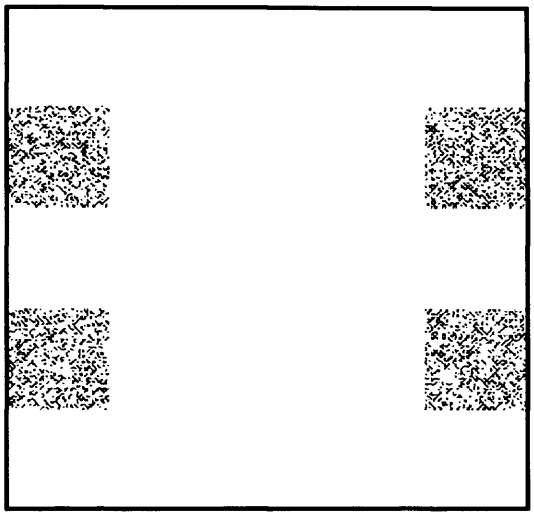
(h)

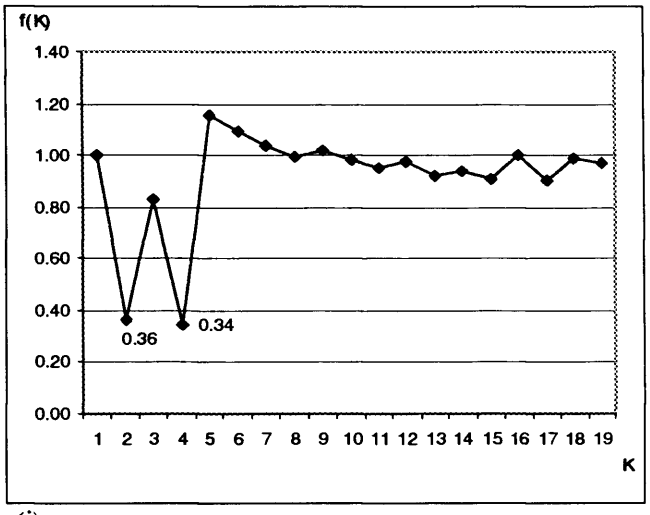




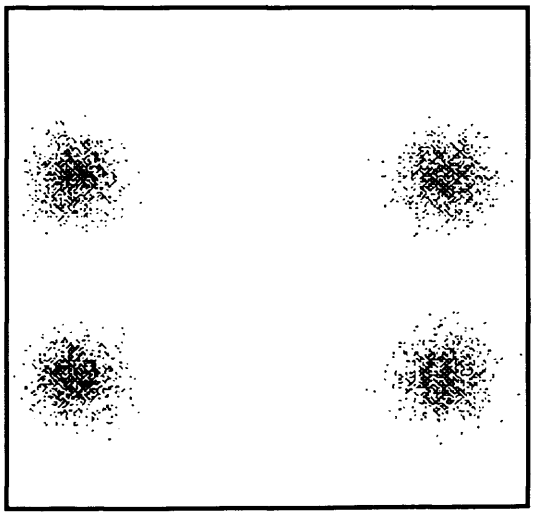
(i)

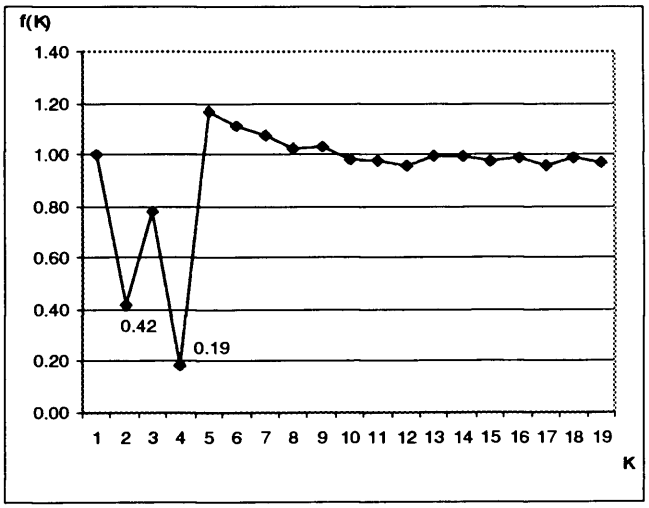
Figure 6.7 (continued)



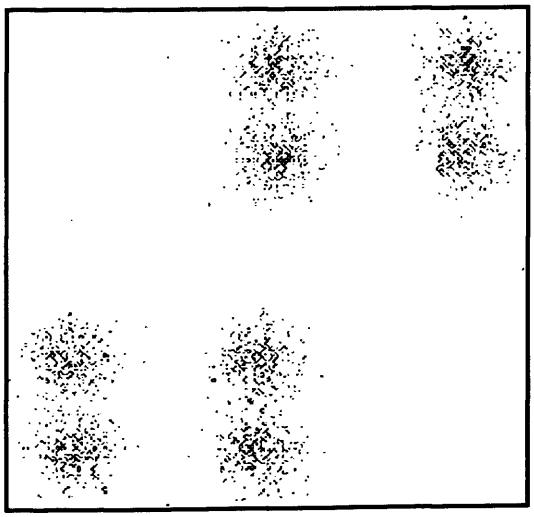


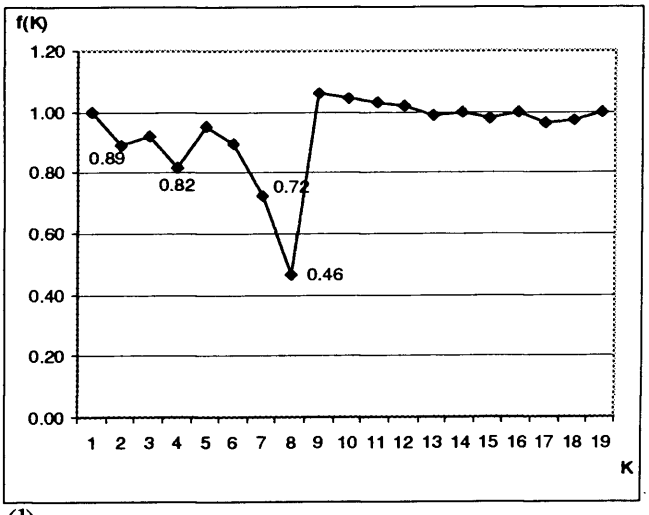
(j)





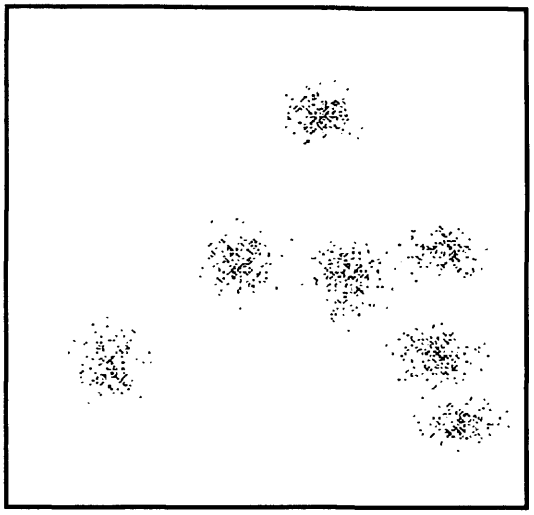
(k)

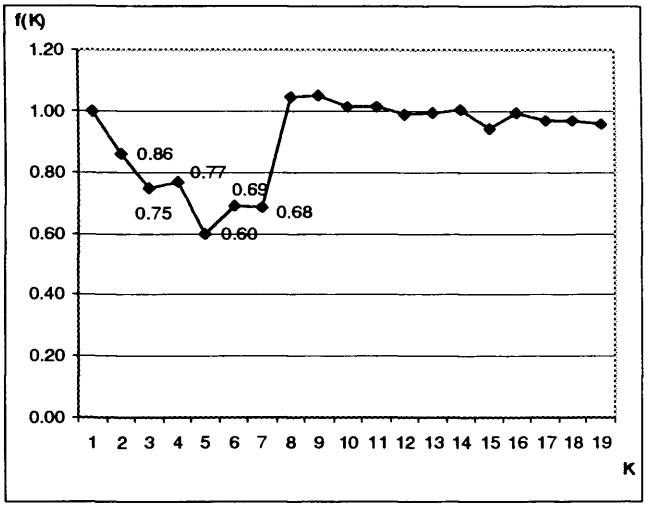




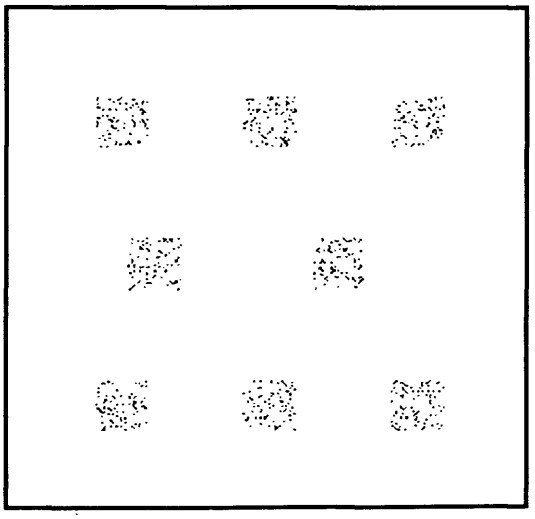
(l)

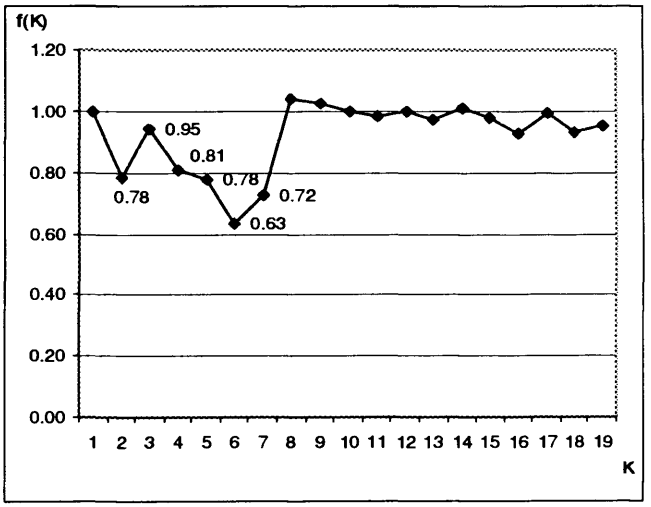
Figure 6.7 (continued)



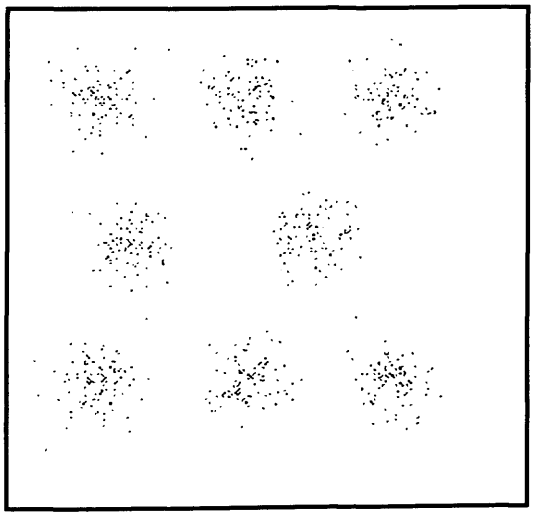


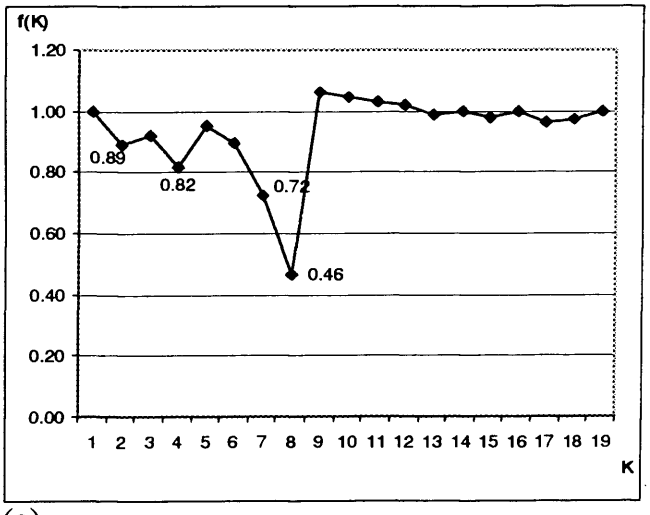
(m)





(n)





(o)

Figure 6.7 (continued)

The proposed function $f(K)$ is also applied to 12 benchmarking data sets from the UCI Machine Learning Repository [Blake et al., 1998]. Figure 6.8 shows how the value of $f(K)$ varies with K. If a threshold of 0.85 is selected for $f(K)$ (from the study on the artificial data sets), the numbers of clusters that will be recommended for each of these data sets are given as in Table 6.2. $K = 1$ means that the data distribution is very close to the standard uniform distribution. The values recommended using $f(K)$ are very small because of the high correlation between the attributes of these data sets, very similar to that shown in Figure 6.7 (e). This can be verified by examining two attributes at a time and plotting the data sets in 2-D.

The above experimental study on 15 artificial and 12 benchmark data sets has proved the robustness of $f(K)$. The evaluation function converges to 1 when K increases above 9.

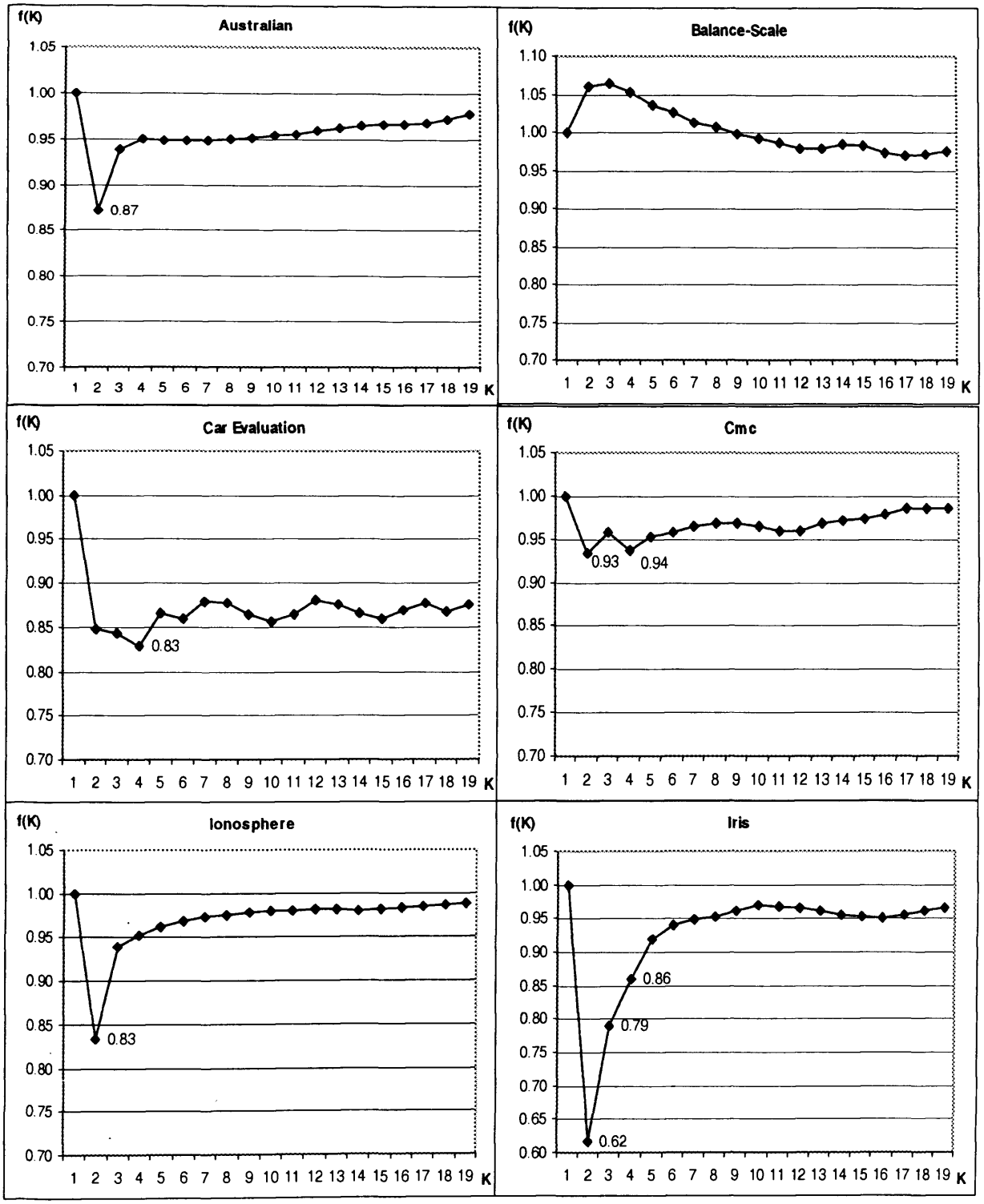


Figure 6.8 – f(K) for the 12 benchmark data sets.

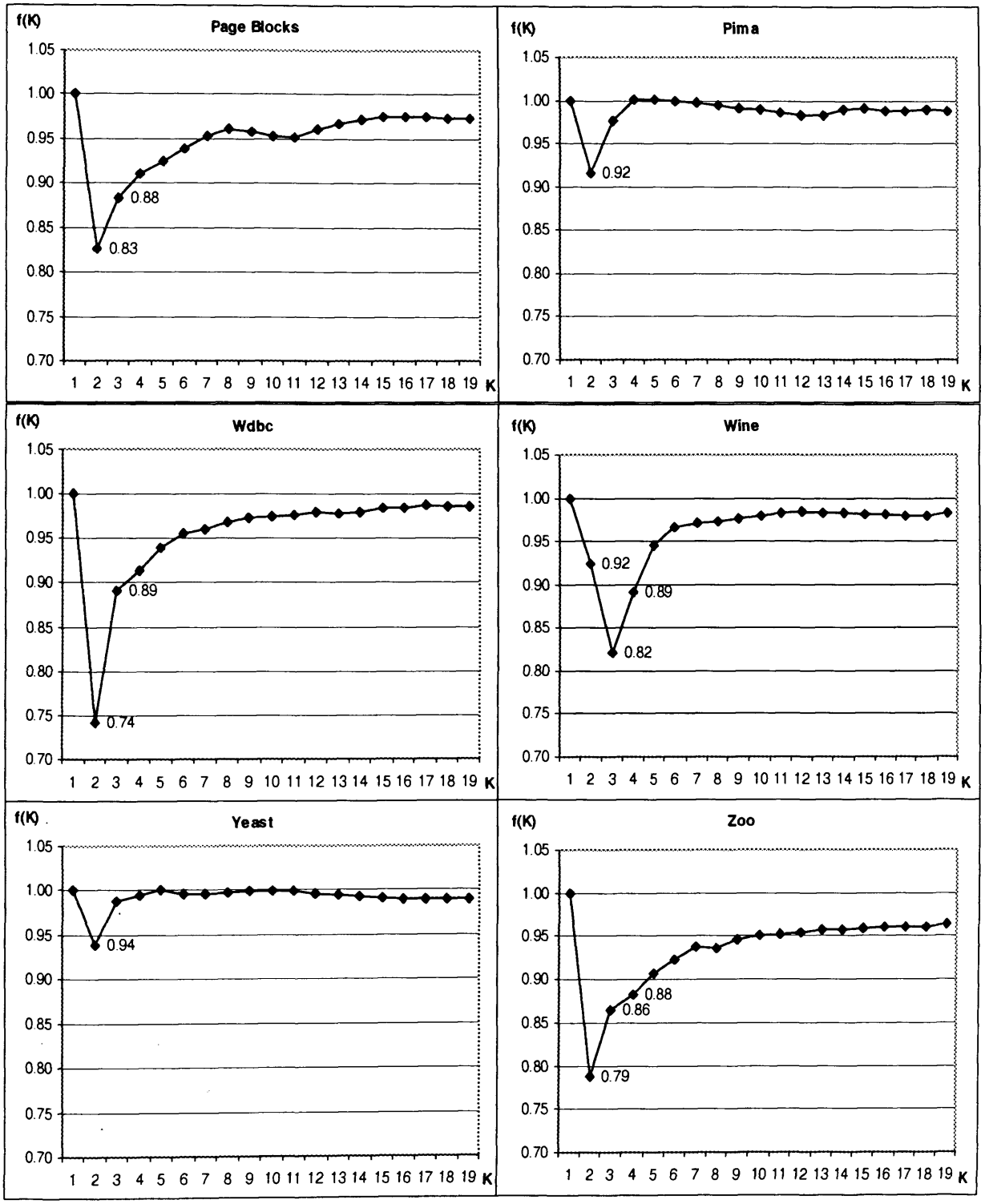


Figure 6.8 (continued)

Table 6.2 – The recommended number of clusters based on f(K).

Data sets	**Proposed number of clusters**
Australian	1
Balance-Scale	1
Car Evaluation	2, 3, 4
Cmc	1
Ionosphere	2
Iris	2, 3
Page Blocks	2
Pima	1
Wdbc	2
Wine	3
Yeast	1
Zoo	2

6.6. Summary

Existing methods of selecting the number of clusters for K-means have a number of deficiencies. Also, methods for assessing the clustering results do not provide sufficient information on the performance of the clustering algorithm.

The main factors affecting the selection of K have been studied. By taking into account these factors in defining the selection method, the selection of K will be tied to the K-means clustering model. Multiple values of K could be recommended to users for cases when different clustering results could be obtained with various required levels of detail. The constraints applied to such a selection method have also been discussed.

A new method to select the number of clusters for the K-means algorithm has been proposed in the chapter. The new method closely relates to the approach of K-means. The performance of the method has been investigated experimentally on 15 artificial and 12 benchmark data sets. Further research is required to verify the capability of this method when applied to data sets with complex object distributions.

Chapter 7

Conclusions and Future Work

This chapter concludes the thesis. The chapter focuses on the main contributions of the research and provides suggestions for future work.

7.1. Conclusions

In chapter 2, constraints and limits on the learning approaches of existing rule induction and data clustering algorithms are studied. The flexibility of these approaches is also discussed.

The first part of this thesis concentrates on contributions to rule induction research. Chapter 3 introduces a new covering method, called RULES-A, which forms the whole rule set simultaneously instead of one rule at a time. RULES-A has the ability to process continuous attributes directly. The adaptive capability of RULES-A enables efficient management of the relationships between the rules in the rule set during the induction process.

In chapter 4, an improved version of RULES-A, called RULES-A1, is introduced. RULES-A1 forms the final rule sets in 2 phases compared with the 3 phases of

RULES-A. The proposed new version can process data sets with both continuous and discrete attributes. Rule pruning is carried out automatically without requiring intervention by users. The strategies of stopping early and varying the order of objects in the training set are applied in enhanced versions of RULES-A1 to increase the data processing speed of the algorithm and the accuracy of the generated rule sets.

The final outcome of this investigation into new rule induction techniques is a family of powerful covering algorithms that outperform C5 and RULES 3+ in classification tasks on benchmark data sets.

The second part of the thesis concentrates on improvement of the K-means algorithm. In chapter 5, the Incremental K-means algorithm is introduced. The algorithm has a flexible strategy for managing the number of clusters. The proposed search strategy decreases the dependence of the algorithm on the initialisation of cluster centres. In addition, the new algorithm only needs to be applied once to achieve almost optimal results. The algorithm consistently outperforms the original K-means algorithm.

A second new K-means algorithm, called Two-Phase K-means, is proposed in chapter 5. Using a buffer, the algorithm consistently generates near optimal solutions with one iteration over large data sets. It employs a simple compression strategy that is computationally more efficient than those applied in other scaled versions of K-means.

Chapter 6 shows that existing methods of selecting the number of clusters are unsuitable for the K-means clustering approach. Current clustering validation methods

cannot show the potential of K-means and limit its capabilities. Factors affecting the selection of the number of clusters are studied thoroughly. A new method to select the number of clusters for the K-means algorithm is proposed. The new proposed method has demonstrated its effectiveness in a series of visual examples and for real data sets.

As a result of the investigation into new clustering methods, new K-means algorithms have been devised that overcome all recognised deficiencies of the K-means algorithm.

7.2. Future Research Directions

A number of aspects of the algorithms developed in this thesis could be improved.

Indexing techniques, such as those based on k-d trees, can be introduced to RULES-A to reduce the complexity of the algorithm in finding classification rules for an object or check the consistency for rules in the rule set.

Although the technique of varying the order of objects in the training set can reduce the dependency of RULES-A on randomness, using a buffer with a suitable strategy to seed candidate rules may improve the quality of formed rule sets.

Indexing techniques can also be applied to reduce the complexity of the Incremental K-means algorithm. The stepping strategy of this algorithm requires further research

to reduce the execution time without affecting the overall performance of the Incremental K-means algorithm and the Two-Phase K-means algorithm.

Because of the different characteristics of discrete and continuous attributes, the flexible management strategy of the Incremental K-means algorithm needs further study using data sets with both types of attributes.

The selection method in chapter 6 needs further investigation with practical data sets having more complicated distributions.

Appendix A

Complexity Estimation of RULES-A

Given

a is the number of attributes

k is the number of rules

e is the number of objects in the training set

p is the number of objects in the pruning set

r is the number of iterations over the training set and specified experimentally

t is the pruning threshold and a predefined parameter

The complexity of RULES-A (Figures 3.1 – 3.4) is estimated in the following sections.

Phases 1:

Step 0

Step 1

Step 2

2.1

2.2. *e* times steps 2.3, 2.4 or 2.5

2.3. Check *k* rules, each rule has *a* attributes

2.4. Check *k* rules; each rule (having *a* attributes) is checked for its consistency with *k-1* rules

2.5.

Step 3

3.1. Update k rules, each rule has a attributes

3.2. Check k rules

3.3. Execute r times steps 3.1 and 3.2

The combined complexity of Phase 1 is $O(aek^2r)$.

Phase 2:

Step 1. Calculate accuracy by checking p objects on k rules; each object or rule has a attributes.

Step 2. Check the capacity of k rules.

Calculate accuracy by checking p objects on k rules; each object or rule has a attributes.

Step 3. Execute t times steps 1 and 2.

The combined complexity of Phase 2 is $O(akpt)$.

Phase 3:

Check k rules

For each rule, check its a attributes

For each attribute, check the overlapping of the rule with the other k-1 rules

The combined complexity of Phase 3 is $O(a^2k^2)$.

The following observations can be seen experimentally:

$a << e$

$e \sim p$

$k << e$

$t << e$

$t < k$

Therefore,

$$a^2k^2 < akpt < aek^2r$$

The three phases are run consecutively, so that the complexity of the RULES-A algorithm is considered to be the largest of these three, $O(aek^2r)$.

Appendix B

Data Sets

Many of the data sets used in this thesis are from the UCI repository of machine learning databases [Blake et al., 1998] and KDD databases [Hettich and Bay, 1999]. These databases were contributed by many researchers, mostly from the field of machine learning and data mining, and collected by the machine learning group at the University of California, Irvine. These data sets are described briefly below.

Balance-Scale data set: This data set was generated to model experimental psychological results. It contains 3 classes (balance scale tips to the right, tips to the left, or is central), 4 numerical attributes and 625 examples.

Abalone data set: The data is used to predict the age of abalone from physical measurements. There are a total of 4177 instances in the data, and each is described by 7 continuous and 1 discrete attributes.

Australian data set: This data set is the modified version of the Credit Approval data set. Attribute 4 is removed. All discrete values were mapped to numerical values.

Car Evaluation data set: This data set was used to evaluate cars according to the features that describe their price, technical characteristics and safety. It contains examples in 4 classes (unacceptable, acceptable, good and very good), 6 continuous attributes and 1728 examples.

Cmc (Contraceptive Method Choice) data set: This data set was used to predict the current contraceptive method choice of a woman based on her demographic and socio-economic characteristics. It contains examples in 3 classes (no use, long-term methods, or short-term methods), 9 numerical attributes and 1473 examples.

Credit Approval data set: This data set concerns credit card applications. It contains examples in 2 classes (-, +), 6 numerical attributes, 9 discrete attributes and 690 examples.

Ionosphere data set: This data set was used to classify radar returns from the ionosphere. It contains examples of 2 classes (g, b), 34 numerical attributes and 351 examples.

Pageblocks data set: This data set was used to classify all the blocks of the page layout of a document that has been detected by a segmentation process. It contains examples in classes (text, horizontal line, vertical line, graphic, picture), 10 numerical and 5473 examples.

Pima Indian Diabetes data set: This data set consists of records on diabetes patients. It contains examples in 2 classes (+, -), 8 numerical attributes and 768 examples.

Tic-tac-toe data set: This data encodes the complete set of possible board configurations at the end of tic-tac-toe games with a 3x3 board, where the player "X"

is assumed to have played first. It contains examples in 2 classes (positive, negative), 9 discrete attributes and 958 examples.

Yeast data set: This data set was used to predict the cellular localisation sites of proteins. It contains examples in 9 classes (CYT, NUC, MIT, ME3, ME2, ME1, EXC, VAC, POX, ERL), 8 numerical attributes and 1484 examples.

Wisconsin Breast Cancer database: Each data point represents data for one breast cancer case. There are three different data sets in this database. The **Wdbc** data set (New diagnostic) is used in this thesis. This data set contains examples in 2 classes (benign or malignant), 31 continuous attributes and 569 examples.

Glass2 data set: This data set was used in a study of glass for a criminological investigation. It contains examples in 2 classes (float_processed, non_float_ processed), 10 numerical attributes and 214 examples.

Iris data set: This is the most widely used data set in the literature. The data set contains 3 classes of 50 instances each, where each class refers to a type of iris flower. Each instance is described by four continuous attributes, namely, sepal length, sepal width, petal length and petal width.

Heart data set: This data set was used in a study of heart disease. It contains examples in 2 classes (absent, present), 13 numerical attributes and 270 examples.

Wine data set: This data set was used in a chemical analysis to determine the origin of wines. It contains examples in 3 classes (1, 2, 3), 13 numerical attributes and 178 examples.

Zoo data set: This data set was used to classify the group of animals. It contains examples in 7 classes (1, 2, 3, 4, 5, 6, 7), 17 numerical attributes and 101 examples.

Diabetes is identical with Pima.

References

Al-Daoud, M. B., Venkateswarlu, N. B. and Roberts, S. A., 1995. Fast K-MEANS Clustering Algorithms. University of Leeds, School of Computer Studies, Report 95.18, June 1995.

Al-Daoud, M. B., Venkateswarlu, N. B. and Roberts, S. A., 1996. New methods for the initialisation of clusters. Pattern Recognition Letters 17, pp. 451-455.

Alsabti, K., Ranka, S. and Singh, V., 1998. An Efficient K-means Clustering Algorithm. Proceedings of the First Workshop on High-Performance Data Mining, Orlando, Florida. ftp://ftp.cise.ufl.edu/pub/faculty/ranka/Proceedings.

Berkhin, P., 2001. Survey of Clustering Data Mining Techniques. Research paper. Accrue Software, Inc. http://www.accrue.com.

Bermejo, S. and Cabestany, J., 2002. The effect of finite sample size on on-line K-means. Neurocomputing 48 (2002), pp. 511-539.

Bilmes, J., Vahdat, A., Hsu, W. and Im, E. J., 1997. Empirical Observations of Probabilistic Heuristics for the Clustering Problem. Technical Report TR-97-018, International Computer Science Institute, Berkeley, CA

Bisbal, J. and Grimson, J., 2001. Database sampling with functional dependencies. Information and Software Technology, Vol. 43, Issue 10, pp. 607-615.

Blake, C., Keogh, E. and Merz, C. J., 1998. UCI Repository of Machine Learning Databases, Irvine, CA. Department of Information and Computer Science, University of California Irvine, CA, USA.

Bose, I. and Mahapatra, R. K., 2001. Business data mining – a machine learning perspective. Information & Management 39, pp. 211-225.

Bottou, L. and Bengio, Y., 1995. Convergence properties of the K-means algorithm. Advances in Neural Information Processing Systems, Vol. 7, MIT Press, Cambridge, MA, pp. 585-592.

Bradley, S. and Fayyad, U. M., 1998. Refining initial points for K-means clustering. J. Shavlik, editor, Proceedings of the Fifteenth International Conference on Machine Learning (ICML '98), Morgan Kaufmann, San Francisco, CA, pp. 91-99.

Bradley, P., Fayyad, U. M., and Reina, C., 1998a. Scaling clustering algorithms to large databases. Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining, NY, pp. 9-15.

Bradley, P., Mangasarian, O. And Street, W., 1998b. Feature selection via mathematical programming. Journal of Computing, 10, pp. 209-217.

Brunk, C. A. and Pazzani, M. J., 1991. An investigation of noise-tolerant relational concept learning algorithms. Proceedings of the 8th International Workshop on Machine Learning, Illinois, pp. 389-393.

Cai, Z., 2001. Technical aspects of data mining. PhD Thesis. Cardiff University, Cardiff, UK.

Castro, V. E. and Yang, J., 2000. A Fast and Robust General Purpose Clustering Algorithm. Fourth European Workshop on Principles of Knowledge Discovery in Databases and Data Mining 2000 (PKDD00), Lyon, France, pp. 208-218.

Cerquides J. and López de M. R., 1997. Proposal and Empirical Comparison of a Parallelizable Distance Based Discretisation Method, Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining, Newport Beach, CA., pp. 139-142.

Chapman P., Clinton J., Kerber R., Khabaza T., Reinartz T., Shearer C. and Wirth R., 2000. CRISP-DM 1.0 Process and User Guide, http://www.crisp-dm.org.

Cheeseman, P. and Stutz, J. 1996. Bayesian Classification (AutoClass): Theory and Results. In Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy , R. (Eds.) Advances in Knowledge Discovery and Data Mining, AAAI Press/MIT Press, pp. 61-83.

Chinrungrueng, C. and Sequin, C. H., 1995. Optimal Adaptive K-means Algorithm with Dynamic Adjustment of Learning Rate. IEEE Transactions of Neural Networks, Vol. 6, No. 1, January 1995, pp. 157-169.

Clark, P. and Boswell, R., 1991. Rule induction with CN2: some recent improvements. Machine Learning - EWSL-91. Proceedings of the European Working Session on Learning, Porto, Portugal, pp. 151-163.

Clark, P. and Niblett, T., 1989. The CN2 Induction Algorithm. Machine Learning, Vol. 3, part 4, pp. 261-283.

Cohen, W. W., 1995. Fast effective rule induction. Proceedings of the 12th International Conference on Machine Learning, Tahoe City, CA, USA, pp. 115-123.

Cook, G. D. and Robinson, A. J., 1995. Utterance clustering for large vocabulary continuous speech recognition. Proceedings of the European Conference on Speech Communication and Technology, Madrid, Spain, Vol. 1, pp. 219-22.

Domingos, P., 1994. The RISE System: Conquering Without Separating. Proceedings of the Sixth IEEE International Conference on Tools with Artificial Intelligence, New Orleans, LA, IEEE Computer Society Press, pp. 704-707.

Domingos, P., 1996a. Linear-Time Rule Induction. Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, Portland, Ohio, pp. 96-101.

Domingos, P., 1996b. From Instances to Rules: A Comparison of Biases. Proceedings of the Third International Workshop on Multistrategy Learning, Harpers Ferry, WV, AAAI Press, pp. 147-154.

Domingos, P., 1996c. Unifying Instance-Based and Rule-Based Induction. Machine Learning, 24, pp. 141-168.

Domingos, P., 1998. Occam's Two Razors: The Sharp and the Blunt. Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining, New York City, NJ, pp. 37-43.

Dougherty, J., Kohavi, R., and Sahami, M., 1995. Supervised and Unsupervised Discretization of Continuous Features. Proceedings of the Twelfth International

Conference on Machine Learning, San Francisco, CA, Morgan Kaufmann, pp. 194-202.

Du, Q. and Wong, T.-W., 2002. Numerical studies of MacQueen's k-means algorithm for computing the centroidal voronoi tessellations. An International Journal of Computers & Mathematics with Applications 44 (2002), pp. 511-523.

Duda, R. O., Hart P. E. and Stork, D. G., 2001. Pattern Classification. Second Edition, John Wiley & Sons, Inc, pp. 517-599.

Epter, S., Krishnamoorthy, M. and Zaki, M., 1999. Clusterability Detection and Initial Seed Selection in Large Data Sets. Technical Report 99-6. Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY.

Estivill-Castro, V., 2002. Why so many clustering algorithms. SIGKDD Explorations, ACM SIGKDD, Vol. 4, Issue 1, pp. 65-75.

Farnstrom, F., Lewis, J. and Elkan, C., 2000. Scalability for Clustering Algorithms Revisited, SIGKDD Explorations, ACM SIGKDD, Vol. 2, Issue 1, pages 51-57.

Fayyad U. M. and Irani K. B., 1993. Multi-interval Discretisation of Continuous-valued Attributes for Classification Learning. Proceedings of the 13th International Joint Conference on Artificial Intelligence, Chambery, France, pp. 1022-1027.

Fayyad U. M., Piatetsky-Shapiro G., Smyth P. and Uthurusamy, R., (1996), Advances in Knowledge Discovery and Data Mining, AAAI Press, Menlo Park, CA.

Fedorov, V. V., Herzberg, A. M. and Leonov, S. L., 2003. Component-wise dimension reduction. Journal of Statistical Planning and Inference, In Press, Corrected Proof.

Forsyth, R., 1989. Machine learning: principles and techniques. Chapman and Hall, London.

Fraley, C. and Raftery, A. 1999. MCLUST: Software for model-based cluster and discriminant analysis. Technical Report 342, Department of Statistics, University of Washington.

Fritzke, B., 1997. The LBG-U method for vector quantization – an improvement over LBG inspired from neural networks. Neural Processing Letters 5, No. 1, pp. 35-45.

Fürnkranz, J. and Widmer, G., 1994. Incremental Reduced Error Pruning. In W. Cohen and H. Hirsh (eds.), Proceedings of the 11th International Conference on Machine Learning (ML-94), New Brunswick, NJ, Morgan Kaufmann, pp. 70-77.

Furnkranz, J., 1997. Pruning Algorithms for Rule Learning. Machine Learning, Vol. 27(2), pp.139-171.

Furnkranz, J., 1999. Separate-and-Conquer Rule Learning. Artificial Intelligence Review, Vol. 13(1), pp.3-54.

Gee, C. J., Fabella, A. B., Fernandes, I. B., Turetsky, I. B., Gur, C. R., and Gur, E., 1999. New experimental results in atlas-based brain morphometry. Proceedings of SPIE Medical Imaging 1999: Image Processing, K. M. Hanson, ed., Bellingham, WA, pp. 604-611.

Grabmeier, J. and Rudolph, A., 2002. Techniques of Clustering Algorithms in Data Mining. Data Mining and Knowledge Discovery, 6, Kluwer Academic Publishers, Netherlands, pp. 303-360.

Grossman, R. L., Kamath, C., Kegelmeyer, P., Kumar, V. and Namburu, R. R., 2001. Data mining for scientific and engineering applications, Kluwer Academic, London.

Halkidi, M., Batistakis, Y. and Vazirgiannis, M., 2002. Cluster Validity Methods: Part I. SIGMOD Record, Vol. 31, Number 2.

Hamerly, G. and Elkan, C., 2002. Alternatives to the k-means algorithm that find better clusterings. Proceedings of the Eleventh International Conference on Information and Knowledge Management (CIKM 02), McLean, VA, pp. 600-607.

Han, J. and Kamber, M., 2000. Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers, NJ.

Hansen, L. K. and Larsen, J., 1996. Unsupervised Learning and Generalisation. Proceedings of the IEEE International Conference on Neural Network, Washington DC, June 1996, pp. 25-30.

Hardy, A., 1996. On the number of clusters. Computational Statistics & Data Analysis 23 (1996), pp. 83-96.

Hettich, S. and Bay, S. D., 1999. The UCI KDD Archive [http://kdd.ics.uci.edu]. Irvine, CA. University of California, Department of Information and Computer Science.

Horch, A. and Isaksson, A. J., 2001. Assessment of the sampling rate in control systems. Control Engineering Practice, Vol. 9, Issue 5, pp. 533-544.

Hunt, E. B., Marin, J., and Stone, P. J., 1966. Experiments in induction. Academic Press, New York, NY, USA.

Insightful Corporation, 2001. S-PLUS 6 for Windows Guide to Statistics, Vol. 2, Insightful Corporation, Seattle, WA. http://www.insightful.com/DocumentsLive/23/44/statman2.pdf

Ishioka, T., Extended K-means with an Efficient Estimation of the number of Clusters, 2000. Proceedings of Second International Conference of Intelligent Data Engineering and Automated Learning (IDEAL 2000), Hong Kong, China, December 2000, pp. 17-22.

ISL: Integral Solutions Ltd., 1998. SPSS Clementine Data Mining System. User Guide Version 5, Basingstoke, Hampshire, UK

Jain A. K. and Dubes R. C., 1988. Algorithms for Clustering Data, Prentice Hall. Englewood.Cliffs, New Jersey.

Jain, A.K., Murty M.N., and Flynn P.J., 1999. Data Clustering: A Review. ACM Computing Surveys, Vol 31, No. 3, pp. 264-323.

Josien, K. and Liao T. W., 2002. Simultaneous grouping of parts and machines with an integrated fuzzy clustering method. Fuzzy Sets and Systems, Vol. 126, Issue 1, pp. 1-21.

Kanungo, T., Mount, D. M., Netanyahu, N., Piatko, C., Silverman, R., and Wu, A., 2002. The Efficient k-Means Clustering Algorithm: Analysis and Implementation. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 24, No. 7, pp. 881-892.

Kaufman, K.A. and Michalski, R. S., 1999. Learning in an Inconsistent World: Rule Selection in AQ18. Reports of the Machine Learning and Inference Laboratory, MLI 99-2, George Mason University, Fairfax, VA, May, 1999.

Kaufman, L. and Rousseeuw P. J., 1990. Finding groups in data: an introduction to cluster analysis, Wiley, New York.

Kerr, A., Hall, H. K. and Kozub, S., 2002. Doing statistics with SPSS. SAGE, London.

Kohavi, R. and John, G., 1998. The wrapper approach. In: Liu, H. And Motoda, H. (Eds.), Feature Extraction, Construction and Selection: A Data Mining Perspective. Springer, Berlin.

Kosmala, A., Rottland, J., and Rigoll., G., 1997. Improved Online Handwriting Recognition Using Context Dependent Hidden Markov Models. Proceeding of the International Conference on Document Analysis and Recognition, Ulm, pp. 641-644.

Kothari, R. and Pitts, D., 1999. On finding the number of clusters. Pattern Recognition Letters 20, pp. 405-416.

Kovesi, B., Boucher, J.-M. and Saoudi, S., 2001. Stochastic K-means algorithm for vector quantisation. Pattern Recognition Letters 22, pp. 603-610.

Langley, P., 1996. Elements of machine learning. Morgan Kaufmann, San Francisco, CA.

Lee, M. D. and Corlett, E. Y., 2003. Sequential sampling models of human text classification, *Cognitive Science,* In Press, Corrected Proofs.

Likas, A., Vlassis, N. And Verbeek, J. J., 2003. The global k-means clustering algorithm. Pattern Recognition 36 (2003), pp. 451-461.

Lindeberg, T., 1994. Scale-space theory in computer vision. Kluwer Academic, Boston.

Liu, H. and Setiono, R., 1995. Chi2: Feature selection and discretization of numeric attributes. Proceedings of 7th IEEE International Conference on Tools with Artificial Intelligence, Washington D.C., November 1995, pp. 388-391.

Liu, H., 1996. Efficient Rule Induction from Noisy Data. Expert Systems with Applications: An International Journal, Pergamon, Vol. 10, No. 2, pp. 275-280, 1996.

Liu, H., 1998. A Family of Efficient Rule Generators. Encyclopaedia of Computer Science and Technology, Marcel Dekker, New York, Allen Kent and James G. Williams (eds.), Vol. 39, pp. 15 - 28.

Lozano, S., Dobado, D., Larraneta, J. and Onieva, L., 2002. Modified fuzzy C-means algorithm for cellular manufacturing. Fuzzy Sets and Systems, Vol. 126, Issue 1, pp. 23-32.

MacQueen, J. B., 1967. Some methods for classification and analysis of multivariate observations. Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability, Vol. I: Statistics, University of California Press, Berkeley and Los Angeles, CA, pp. 281-297.

McGovern, 1998. acQuire-macros: An Algorithm for Automatically Learning Macro-actions. Neural Information Processing System Conference 11 (NIPS'98), Workshop on Abstraction and Hierarchy in Reinforcement Learning, http://www-anw.cs.umass.edu/~amy/pubs.html.

Michalski, R. S., 1977. Variable-valued logic and its applications to pattern recognition and machine learning. Computer Science and Multiple-Value Logic. Theory and Application. David C. Rine Editor, pp. 506-534.

Michalski, R. S., Mozetic, I., Hong, J., and Lavrac, N., 1986. The multipurpose incremental learning systems AQ15 and its testing application to three medical domains. Proceedings of the 5th National Conference on Artificial Intelligence, AAAI, Philadelphia. Morgan Kaufmann, PA, pp. 1041-1045.

Michalski, R. S., Bratko, I. and Kubat, M., 1998. Machine Learning and Data Mining: Methods and Applications, Wiley, Chichester, West Sussex, UK.

Michie, D., Spiegelhalter, D.J. and Taylor, C.C., 1994. Machine learning, neural and statistical classification, Prentice Hall, Englewood Cliffs, NJ.

MIT, 1998. DataEngine 3.0 - Intelligent data analysis - an easy job. MIT- - Management Intelligenter Technologien GmbH, Germany, http://www.mitgmbh.de.

Mitchell, T., Machine Learning and Data Mining, 1999. Communications of the ACM, Vol. 42, No. 11, November 1999, pp. 30--36.

Ng, R. and Han, J., 1994. Efficient and Effective Clustering Method for Spatial Data Mining, Proceedings of 1994 International Conference on Very Large Data Bases (VLDB'94), Santiago, Chile, September 1994, pp. 144-155.

Pagallo, G. & Haussler, D., 1990. Boolean feature discovery in empirical learning. Machine Learning, Vol. 5, pp. 71-99.

Paliouras, G. and Bree, D. S., 1995. The effect of numeric features on the scalability of inductive learning programs. Proceedings of the European Conference in Machine Learning. Springer-Verlag, Berlin, pp. 218-231.

Pao, Y.-H., 1989. Adaptive Pattern Recognition and Neural Networks. Addison-Wesley, New York, NY.

Patane, G. and Russo, M., 2001. The enhanced LBG algorithm. Neural Networks 14 (2001), pp. 1219-1237.

Pelleg, D. and Moore, A., 1999. Accelerating Exact K-means Algorithms with Geometric Reasoning. Proceedings of the Conference on Knowledge Discovery in Databases 1999 (KDD99), San Diego, CA, pp. 277-281.

Pelleg, D. and Moore, A., 2000. X-means: Extending K-means with Efficient Estimation of the Number of Clusters. Proceedings of the Seventeenth International Conference on Machine Learning (ICML2000), Stanford, CA, pp. 727-734.

Pena, J. M., Lazano, J. A. and Larranaga, P., 1999. An empirical comparison of four initialisation methods for the K-means algorithm. Pattern Recognition Letters, Vol. 20, pp. 1027-1040.

Pham, D. T., and Aksoy, M. S., 1995a. RULES: a simple rule extraction system. Expert Systems with Applications, Vol. 8, no. 1, pp. 59-65.

Pham, D. T., and Aksoy, M. S., 1995b. A New Algorithm for Inductive Learning. Journal of Systems Engineering, Vol. 5, no. 2, pp.115–112.

Pham, D. T., and Dimov, S. S., 1996. The RULES-3 Plus inductive learning algorithm. Proceeding of the 3rd World Congress on Expert Systems, Seoul, Korea, Vol. 2, pp. 917-924.

Pham, D. T., and Dimov, S. S., 1997. An Algorithm for Incremental Inductive Learning. Proceeding of the Institution of Mechanical Engineers, Vol. 211, part B, pp. 239–249.

Pham, D. T., and Dimov, S. S., and Salem, Z., 2000. Technique for selecting examples in inductive learning. Proceeding of the European Symposium on Intelligent Techniques (ESIT 2000), Aachen, Germany, pp. 119-127.

Quinlan J. R., 1986. Induction of Decision Trees, Machine Learning 1, pp. 81-106.

Quinlan, J. R., 1993. C4.5: Programs for Machine Learning. Morgan Kauffman, San Mateo, California.

Romesburg, H. C., 1990. Cluster Analysis for Researchers, Krieger Publishing, Malabar, Florida.

RuleQuest, 2000. RuleQuest Research Pty Ltd, 30 Athena Avenue, St Ives NSW 2075, Australia. www.rulequest.com.

Schafer, J.B., Konstan, J. and Riedl, J., 2001. E-Commerce Recommendation Applications, Data Mining and Knowledge Discovery, 5, pp. 115-153, Kluwer Academic Publisher, Netherlands.

Theiler, J. and Gisler, G., 1997. A contiguity-enhanced K-means clustering algorithm for unsupervised multispectral image segmentation. Proceedings of Algorithms, Devices, and Systems for Optical Information Processing, San Diego, CA, Vol. 3159, pp. 108-118.

Theodoridis, S. and Koutroubas, K., 1998. Pattern Recognition. Academic Press, London.

Tibshirani, R., Walther, G. and Hastie, T., 2000. Estimating the number of clusters in a dataset via the Gap statistic. Technical Report 208, Dept. of Statistics, Stanford University.

Tolun, M. R. and Abu-Soud, S. M., 1998. ILA: An Inductive Learning Algorithm for Rule Extraction. Expert Systems with Applications, Vol. 14(3), 361-370, April 1998.

Tolun, M. R., Sever, H., Uludag, M., and Abu-Soud, S. M., 1999. ILA-2: An Inductive Learning Algorithm for Knowledge Discovery. Cybernetics and Systems, Vol. 30, part. 7, pp.609-628.

Ventura, D. and Martinez, T., 1995a. An Empirical Comparison of Discretization Methods. Proceedings of the Tenth International Symposium on Computer and Information Sciences, Kusadasi, Turkey, pp. 443-450, 1995.

Ventura, D., 1995b. On Discretisation as a Preprocessing Step for Supervised Learning Models. Master's Thesis, Computer Science Department, Brigham Young University, 1995.

Wallace, C. and Dowe, D. 1994. Intrinsic classification by MML ñ the Snob program. Proceedings of the 7th Australian Joint Conference on Artificial Intelligence, UNE, World Scientific Publishing Co., Armidale, Australia, pp. 37-44.

Wilson, D. R. and Martinez, T. R., 1997. Improved Heterogeneous Distance Functions. Journal of Artificial Intelligence Research, Vol. 6, pp. 1-34.

Winters, N. and Victor, J. S., 2002. Information Sampling for vision-based robot navigation. Robotics and Autonomous Systems, Vol. 41, Issues 2-3, pp. 145-159.