Design and Analysis of Rule Induction Systems

A thesis Submitted to Birmingham University for the degree of Doctor of Philosophy

By

Khaled Eldbib

MSc in Electronics Engineering

BSc in Physics Science

College of Engineering and Physical Sciences

School of Mechanical Engineering

Birmingham University

June 2015

Abstract

Knowledge acquisition is an important area of artificial intelligence. Inductive learning is an efficient and simple method to acquire knowledge from large amounts of information. By automating knowledge acquisition, inductive learning can be expected to clear a major bottleneck in the building of expert systems.

This study begins with a survey of machine learning. The survey reveals that although inductive learning performs reasonably well in many applications, there are still a number of improvements to be made to the existing inductive learning algorithms. One of the improvements needed is the development of better methods for selecting relevant attributes, selecting seed examples, correcting errors or dealing with missing information.

The RULES family of algorithms is reviewed in this work and the drawback of the variation in their generalisation performance is investigated. This results in a new data ordering method (DOM) for the RULES family of inductive learning algorithms. Dom is based on the selection of the most representative example; the method has been tested as a pre-processing stage for many data sets and has shown promising results.

Another difficulty faced is the growing size of training data sets, which results in long algorithm execution times and less compact generated rules. In this study a new data sorting method (DSM) is developed for ordering the whole data set and reducing the training time. This is based on selecting relevant attributes and best possible examples to represent a data set. The enhanced method was also used as a pre-processing step for the RULES family.

Finally, the order in which the raw data is introduced to the RULES family algorithms considerably affects the accuracy of the generated rules. This work presents a new data grouping method (DGM) to solve this problem, which is based on clustering. This method, in the form of an algorithm, is integrated into a data mining tool and applied to a real project; as

a result, better variation in the classification percentage and a lower number of rules formed has been achieved.

Dedication

to my mother and the soul of my father.

Acknowledgements

First I would like to give thanks and praise Allah Almighty for guiding me in this endeavour. The research presented in this thesis was carried out under the supervision of Professor D. T. Pham. I would like to sincerely thank Professor Pham for his constant support during these past years. The freedom you gave me allowed my intuitions and imagination to turn into ideas. Your guidance turned these ideas into constructive work.

Many thanks to all the members of the Manufacturing Engineering Centre at the School of Engineering of Cardiff University and to the School of Mechanical Engineering, College of Engineering and Physical Sciences, Birmingham University. My gratitude goes in particular to my fellow PhD students for all our valuable discussions.

Finally, I am grateful to my wife Mrs H. M. Eldbib for her moral support and patience throughout my research. Thanks to Allah for gifting me with my twin daughters “Safa and Marwah” and my son "Elhadi". Without them, I could not have done it.

Table of Contents

List of Figures

List of Tables

Nomenclature

A_i	the i_{th} attribute in an example
A	the maximum number of entries in a leaf node of the CF tree
BS	Beam Size, the maximum number of conjunctions forming
B	the maximum number of entries in an internal node of the CF tree
C	the number of values appearing in T for a continuous attribute
cj	the centre of the jth cluster
E	the sum of the Euclidean distance between each data point and its closest cluster
Eps	a specified radius for a data point in a given cluster
k	the number of clusters in a data set
F	fuzzy set
L	the log likelihood of a mixture model
$MinPts$	the minimum number of data points within a radius Eps
nj	the number of data points in cluster j
N	the number of data points in a data set
N	the total number of instances in the training data set D.
N_a	the total number of attributes.
N_c	the total number of continuous attributes.
N_n	the total number of nominal attributes.
P_0	the probability of target class in the training data
pj	a mixture weight for cluster j
$p(x)$	mixture model

$p(x\|j)$	mixture model for cluster j
SE	Seed Example
S	Set of instances.
$\|S\|$	the number of instances in S.
S_i	the i_{th} partitions of the data set S.
T	training set of examples
Th	the noise threshold
Tr	triangular membership function
w	the beam width.
$x(j)I$	the ith data point belonging to the jth cluster the beam width.

Abbreviations

AI	Artificial Intelligence
AQ	Algorithm Quasi-optimal
ARI	Adaptive Rule Induction
CART	Classification and Regression Trees
DC	Data Clustering
DM	Data Mining
DOM	Data Ordering Method
DSM	Data Sorting Method
DGM	Data Grouping Method
DGMK-means	Data Grouping Method with K-means Clustering Algorithm
ID3	Iterative Dichotomiser Version 3
KDD	Knowledge Discovery from Databases
FOIL	First Order Inductive Learner
ILP	Inductive Learning Process
ML	Machine Learning
RIPPER	Repeated Incremental Pruning to Produce Error Reduction
RULES	RULE Extraction System
RULES-3 Plus	RULE Extraction System - Version 3 Plus
RULES-5	RULE Extraction System - Version 5
RULES-5 Plus	RULE Extraction System – Version 5 Plus
RULES-F	RULE Extraction System - Version 5 Fuzzy
SRI	Scalable Rule Induction
STM	Short Term Memory

Chapter 1: Introduction

1.1 Background

For a machine to behave like a human, it requires the capacity to learn. Humans' capacity to learn has enabled the human population to interact with the real world and carry out various everyday tasks and routines. Humans have to learn how to undertake tasks and resolve problems since no one is born with pre-programmed solutions to all possible problems. Learning from experience and observation is one of the most significant features of the human learning process and is the best method of successfully acquiring and gaining knowledge. Nowadays, information systems can capture, record and store increasingly huge volumes of data on a daily basis. Data mining and machine learning techniques can be deployed to extract previously unknown and potentially useful knowledge from historical data. The knowledge can then be used by a decision maker or any other user to make informed decisions.

Data mining, also known as Knowledge Discovery in Databases (KDD), relates to the nontrivial extraction of implicit, previously unknown and potentially useful information from data stored in databases. Recent advances in the information technology industry have given organisations the ability to capture record and store vast quantities of data. As a result, an increase in the volume of such data diminishes the human understanding of it, and we become unaware of the hidden information lying within the data. The hidden information often becomes potentially useful when made explicit and taken advantage of.

A number of researchers have carried out extensive studies on machine learning for the past few decades. The aim of this study was to automate the process of knowledge acquisition by constructing expert systems. The emergence of data mining, with the application of algorithms for machine learning, led to a demand for algorithms that can effectively deal with large amounts of data. Machine learning as a process can automate the overall routine methods and procedures of inductive learning, while using coded observations from a data base that is previously stored, and without interacting directly with environment.

There are two main tasks that are commonly associated with machine learning methods: classification and clustering. The presence of an expert in a learning system is useful, and often required, when inducing classifiers from training data. Trained classifiers are deployed to arrange new objects into specified classes. A well-known method, Rule induction, is frequently used to generate classifiers. The knowledge depicted as rules is easily understood and can be verified by users without any difficulty. Moreover, the rules created through the learning process can be used directly in knowledge-based systems. On the contrary, data clustering is often deployed to find natural groups and to identify interesting distributions and patterns within the data. Clustering techniques encompass various algorithms and methods for grouping objects into clusters based on their similarities and dissimilarities. The end result of a clustering process is a scheme that allows data to be grouped in a data set, or a suggestion about regularities or dependencies in the data. Taking these characteristics into account, clustering is deployed as a pre-processing technique in this study. Representative training sets can be established by distributing portions of the data sets into homogeneous clusters prior to picking out typical examples from each cluster. For instance, the representatives for these clusters can be discovered by utilising the k-means algorithm, incremental k-means algorithm and fuzzy c-means algorithm.

1.2 Research Objectives

The main aim of this study was to develop flexible management strategies for machine learning and data mining techniques to improve their performance. To achieve this aim, the following objectives were addressed:

- To minimize the number of training rule-sets generated. (Objective 1.)
- To develop new data pre-processing techniques, focusing on feature selection and reduce the training time while handling large numbers of input data. (Objective 2.)
- To develop a pre-processing method that would improve the generalisation capability and achieve higher classification accuracy by using data clustering techniques. (Objective 3.)

1.3 Outline of the thesis

The thesis is organised into three sections. The first section is a literature review (Chapter 2). The second section (Chapters 3 and 4) describes methods for pre-processing training examples. Finally, section three (Chapter 5) describes the clustering methods used for improving the output of inductive learning algorithms. These sections are followed by the conclusions (Chapter 6).

Chapter 2 reviews the current literature on Data Mining and Machine Learning. The chapter presents a critical review of the data mining process, induction and clustering concepts, and data engineering.

Chapters 3 study the RULES family of inductive learning algorithms. It starts with a review of all of the RULES family members highlighting the benefits and limitations of each. The chapter then presents a new data ordering method (DOM) for pre-processing the training examples. The method consists of three main steps: (i) calculating the entropy for a sub group; (ii) aggregating the entropy for each example; (iii) reordering examples according to their entropies.

Chapter 4 discusses feature selection. The chapter briefly reviews different related feature selection topics described in the literature. It then focuses on a new strategy for managing training data. This data pre-processing strategy is called the data sorting method (DSM), which also has other improved capabilities. DSM comprises three tasks: (i) changing the order of training objects, (ii) changing the order of training attributes, and (iii) changing the order of the training data set after each rule is produced.

Chapter 5 provides details of clustering methods for improving the output of inductive learning algorithms. The section focuses on enhancements aimed at improving the data grouping method (DGM). The work is based on an initial clustering stage followed by the ranking of the data in each cluster by a special density measure that considers the distribution of the patterns in the pattern space. This is similar to the formation of the rule sets formed by the Rules Family algorithms.

Chapter 6 concludes the study by outlining the main themes of the study and the main contributions together with suggestions for future research.

Chapter 2: Literature Review

2.1 Machine Learning and Data Mining

Artificial Intelligence (AI) research has long been aimed at the creation of machine intelligence. The objective is to develop a machine which acts like a human, requiring the machine to have the capacity to learn. The process of learning involves acquiring knowledge through study or through experience. It also involves committing information to memory, and receiving and responding to instructions. Another method of learning is to become aware of information through observation (Witten and Frank, 2005).

This chapter first presents a general overview of the learning process, specifically machine learning and data mining. Inductive learning is then discussed, along with its principles, types, categories, algorithms and software.

2.1.1 Machine Learning

Over the past few decades machine learning has been studied intensely by various researchers. For this research, the researcher aimed to automate the process of acquisition of knowledge by constructing expert systems. The development of data mining as an application which makes use of machine learning led to a need for algorithms that can handle large amounts of data efficiently. Machine learning is a process that can automate the overall system of inductive learning, while using coded observations from a stored data base so that direct interaction with the environment is not required. Aronis and Provost (1997) propose machine learning as a technical basis for the data mining process. They describe machine learning as an important practical consideration with a justified existence in various fields.

For example, data in large databases may be stored and regenerated for reusable purposes instead of being used for learning. Bay (2000) stated that machine learning can allow individuals to gain knowledge about a particular topic by studying or experiencing it. Aha et al. (1991) Showed that machine learning is the modification of behavioural tendencies in response to individual experiences.

Machine learning is the scientific process that deals with the programming of systems to enable them to learn from experiences and improve automatically (An and Cercone 1999). Machine learning does not refer to “learning by heart”, but rather to the recognition of complex patterns in data, to enable subsequent intelligent decisions to be made. The main difficulty in this approach is how to describe the set of all possible decisions that can be made based on the inputs. To overcome this weakness, algorithms based on computational and statistical principles have been developed by various researchers to enable the discovery of knowledge regarding specific data (Bayardo ,1998). Wagstaff (2012) said that the machine learning research field includes various distinct approaches. These include logic, probability theory, searching, reinforcement learning, control theory and statistics. These methods were developed for various applications such as forecasting, data mining, robotics, expert systems and pattern recognition. According to Aronis and Provost (1997) various computer algorithms are studied in machine learning to enable a goal to be achieved. For instance a goal may be to make accurate predictions, to effectively complete a task, or to behave intelligently. Aha et al. (1991) suggested that learning should be based on observations or on data mining. Generally machine learning is the process that enables a machine to perform better in the future based on knowledge of past experiences. The main focus of machine learning research is the development of automatic methods that can enhance the ability of machines to learn from past experiences (Aha et al., 1991). The main aim of machine learning is to develop learning algorithms that can enable machines to assimilate information and enhance their working capabilities without the assistance of a human being. Wagstaff (2012) defined machine learning as a sub-area of artificial intelligence. It is not possible to build a system that can facilitate intelligence and vision without the use of a learning process. Wagstaff (2012) suggest that building such a system is very difficult. Wagstaff (2012) further stated that we cannot deem a system to be intelligent if it is not capable of learning, as learning is the main criterion for intelligence.

According to Bay (2000), machine learning research interacts with other fields such as physics, computer science, and especially statistics. The principle goal of machine learning is to develop algorithms that have practical value, and these algorithms must be efficient so that they can enhance a machine's capabilities by enabling it to learn from previous experiences (Barber, 2012). The data required by learning algorithms has its own importance, as it is essential for learning algorithms to adapt based on previous experiences. Wagstaff (2012) stated that machine learning is a system that identifies different patterns automatically from data or past experiences. Bay (2000) said that machine learning is being used in diverse fields such as spam filters, internet searches, recommender systems, fraud detection, credit scoring, ad placement, stock trading and drug designing.

Machine learning requires learning algorithms that can be easily applied to solve different problems. Figure 2.1 shows a typical machine learning problem model.

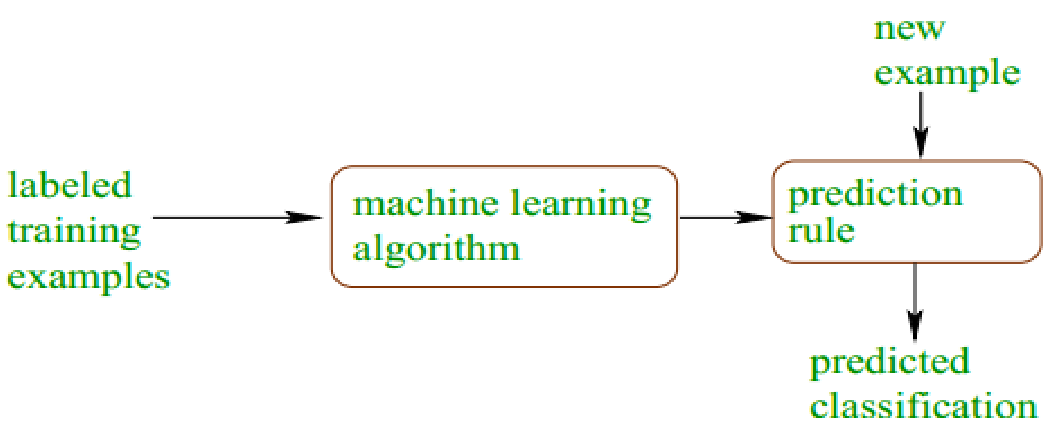


Figure 2.1: The Machine Learning framework.

Wagstaff (2012) described three basic components that can help an individual to select an appropriate learning algorithm for a particular job, as there are thousands of algorithms, with thousands more being publisher every year (Wagstaff, 2012). These three basic components are representation, evaluation and optimisation. i.e:

Learning = representation+ evaluation+ optimisation. (2.1)

Bishop (2006) claimed that a classifier should represent formal language that can be understood by the computer. It is necessary to acknowledge to the appropriate representation of the inputs (which features are to be used). The evaluation component is also known as the objective function. It differentiates good classifiers from bad classifiers. Bay (2000) said that the selection of the best optimisation technique is necessary for the efficiency of the learner. This also helps to determine the produced classifier.

A machine learning task is shown in Figure 2.2. The figure shows a collection of two dimensional data. Different cluster colours represent different class labels. In this figure a classification is dividing the boundary between the different class of clusters.

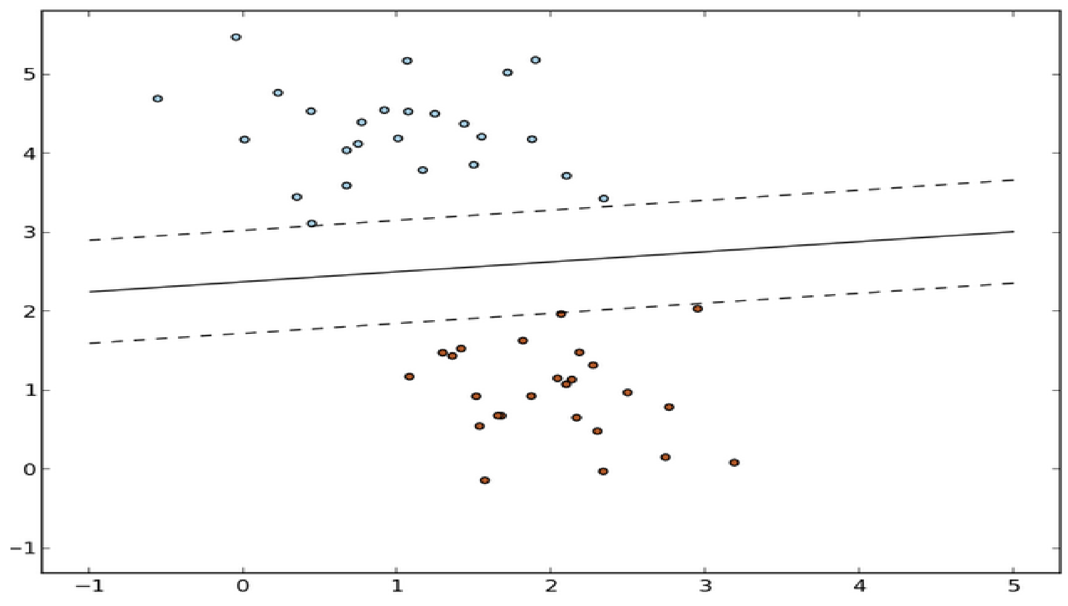


Figure 2.2: Example of a Linear Decision Boundary.

2.1.2 Data mining

Cabena et al. (1997) defined data mining as the process of extracting previously unknown, valid and actionable information from large data sets and then using the information to make crucial business decisions.

According to Berry and Linnoff (2004), data mining is the process in which predictive information from large databases is extracted. It is an advanced technique enabling businesses to emphasise the most important information within their data warehouses. Bramer (2007) stated that data mining is a tool that predicts future behaviours and trends. A data mining tool may help businesses to find answers to the most time consuming and difficult questions of the organisation. In order to enhance the value of existing information sources, businesses can rapidly implement data mining techniques, further helping them to integrate new systems and products. Calderon et al. (2003) explained data warehouses as platforms that contain all the data of the organisation under one place, and in a centralised form for the deployment to users. This can help business organisations to accomplish all tasks from simple reporting to complicated analysis, support decisions and the attainment of objectives. Han and Kamber (2006) described data warehouses as the depository of information that business organisations require if they are to succeed in the information age. Larose (2004) stated that the data warehouse is the most powerful strategic tool for businesses in the information age. It helps business organisations to compete across time it also helps business organisations in developing strategies to evaluate the strategic insights of employees of the organisation in various fields.

Markov and Larose (2007) said that data mining is a process that analyses the activities of business organisations historically, stores statistical data in a warehouse, and reveals hidden patterns and trends. Business organisations use different data mining techniques for performing different tasks of the business. For example, market analysis may be used for identification of new product bundles and prevention of reduction in customers. In today’s competitive business environment, organisations are collecting raw data about their customers at an unprecedented and increasing rate. An example of this is Walmart, which is now processing more than 20 million point of sale transactions every day (Alexander, 2008). The customers’ information is stored in the company’s centralised database, but it is useless without data mining software that can extract and analyse the relevant information. Data mining benefits Walmart in various ways, such as determination of sales trends, development of marketing campaigns, and prediction customer loyalty. Data mining processes use information from previous experiences to prove a solution to a present problem or to evaluate a situation (Witten and Frank, 2005). According to Witten and Frank (2005) OLAP (online analytical processing) enables individuals to analyse data from various perspectives by discovering correlations between customers of the business. The main steps of the data mining process are (i) input of data, (ii) pre-processing, (iii) analysis and (iv) validation (Witten and Frank, 2005). Wang (1999) claimed that data mining methods could be applied to the analysis of process systems, and that the results could be used for tasks such as process design, process improvement, process monitoring and operator training.

Piatetsky-Shapiro et al. (1996) stated that applications for data mining techniques have ranged from law enforcement to radio astronomy. Medical applications such as lymphography, breast cancer, primary tumour and thyroid (Quinlan, 1987; Quinlan, 1988; Michalski et al., 1996). Other applications include marketing (Thorpe et al., 1989), finance (Selby and Porter, 1988), banking (Carter and Catlett, 1987), soil classification (Dale et al., 1989), military decision making (Lirov et al., 1989), the space shuttle (Modesitt, 1987) and others. Most applications use data mining for classification and prediction.

2.2 Inductive Learning methods

Inductive learning is a learning method to explore historical observations and extract a limited number of general principles. Rather than simply remembering all experiences, human intelligence uses inductive learning to deal with the rapid increase of information brought about by the information age. That is to say, the experts can predict what will happen in the future and adopt appropriate behaviour on the basis of learned principles.

The main purpose of inductive learning is to execute a process to separate new knowledge. This process is autonomous, of the form given to the information input. The first task in the formation of knowledge through inductive learning is the collection of representative examples of expert decisions.

According to Bigot (2002) inductive learning is the process of learning in which a general rule is induced by the system from a group of observed events. Bigot (2002) further stated that inductive learning involves classification of a particular input and the name of the group in which that input belongs. Classification is of great importance in various problem solving tasks. Blake and Merz (1998) said that an inductive learning system should involve its own group descriptions. Burdick et al. (2001) stated that it is the task to be performed that determines the class definitions. According to Agarwal and Srikant (1994), an inductive learning algorithm has three main components: representation, search, and evaluation. Agarwal et al. (1993) described the representation component as a formal language used to describe concepts, while the results of an inductive learning algorithm are statements presented in this formal language. According to Kung (2014), the search component is the process in which the concept description is searched by the learning algorithm in a space containing the descriptions arising from the representation process. Sajja and Akerkar (2012) stated that the evaluation component is a process in which and individual proceeds with the measurement of quality. The evaluation process guides the search process and helps in determining when to stop searching. Kersting (2006) stated that evaluation procedures are used for two main purposes. According to Bigot (2002) inductive learning has two main categories, known as decision tree induction and rule induction. The single conjunctive rule algorithm is a machine algorithm also known as inductive learning. The main purpose of inductive learning is to produce a group of rules that capture all the knowledge (generalised) within a set of data. In inductive learning, a classifier is based on the use of rules to test instances that are triggered by matching features of the left hand side rule. There are various forms of rules that can be ordered. According to Kersting (2006), the ordered rule conflagrates the classification outcome and stops the progress of the overall classification process. Many researchers have studied inductive learning over the past 10 years (Sajja and Akerkar 2012). According to Bigot (2002), induction is the process that moves the focus from specific to general. It is the process of generalising a description of a procedure obtained from observed data. The main purpose of inductive learning is to execute an amalgamation of new knowledge where this knowledge is different from the input information. Kung (2014) stated that a researcher can form knowledge base for inductive learning by gathering a group of representative examples regarding expert decisions. It was further explained that every example relates to a class and can be described by its number of attributes. An inductive learning process attempts to search for a method that can classify an example. This method can be further expressed as the function of different attributes which explain the training examples. This process can also be used for classification of previous cases.

Inductive learning attempts to find the underlying concepts within a data set. Concepts can be represented in different ways, for instance first-order logic, decision trees and neural network weights. Several learning systems have been developed, including neural networks, decision trees, rule induction programs and Bayesian networks. In this section, decision tree algorithms and rule induction algorithms are reviewed, since they are directly linked to this work.

Inductive learning creates a set of decision rules or a decision tree from training examples with known classification. Holland et al. [1986] stated that knowledge representation could either be rule-based or use decision trees. Based on this assumption, these two types of induction will be discussed in this study.

2.2.1 Decision Tree-based Algorithms.

As discussed above, classification is the most popular and well-known technique for data mining. Its industrial applications include pattern recognition, imaging and fault detection. Braha (2001) stated that almost all current approaches to categorisation use some knowledge of the overall data set. A decision tree is an algorithm consisting of internal nodes, roots which are labelled specific questions, and arches that correspond to possible answers to these questions (Cervone et al., 2001). In decision tree analysis, each tree node corresponds to a prophecy of a solution to a particular problem. Monostori (2002) remarked upon the popularity of the decision tree algorithm, and proposed the use of leaf nodes to indicate the class to which the corresponding tuple belongs. According to Pham et al. (2002) the decision tree algorithm assumes that all the entries are linked with each other. Pham et al. (2002) suggest that the decision tree algorithm does not reflect upon both horizontal and vertical interactions, even though both of these interactions are common in learning processes. Instead, the authors propose IBSEAD to support decision tree algorithms by filling this gap. It was further explained that IBSEAD is comprehensive, providing a clearer and more precise picture than its precursors. IBSEAD is a program that can answer the problems of a question while accumulating awareness and perception in machines. That is something that decision tree algorithms fail to include. IBSEAD is a program that contains entities as individuals with a common goal, rather than representing them as groups. This structure mimics the true observed in almost every dynamic and living environment. One example is the situation where a robot is tasked with boarding a railway train. To do so it must interact with people, either in groups or individually. This situation contains many unknown entities that are not possible for the robot to see, even though their effects can be felt by the robot. Composite situations like this can be handled by IBSEAD, but not by decision tree algorithms, or by any other algorithm. According to Cervone et al. (2001), the IBSEAD program is comprehensive and can be used more deeply and more easily than other hierarchical structures based on decision tree algorithms. It was further explained that the aptitude of algorithms to execute higher levels of human awareness and knowledge are not credible, and that IBSEAD can help in this regard. Cervone et al. (2001) described numerous algorithms that can be employed for constructing decision trees, with the most popular among them being CART and ID3 (Quinlan, 1986). ID3 (Iterative Dichotomiser), C4.5 (Quinlan, 1993) and C5.0 (Rulequest Research, 2001) are all algorithms classified as divide and surmount inductive systems. Cervone et al. (2001) further explained that the knowledge provoked by these algorithms can be used to represent decision trees. Every internal node of a decision tree represents a test of its characteristics, while every branch of the decision tree represents the possible results of the test.

The common mode of action for making decision trees for a training data set *T* is to start from a single root node and operate recursively. This procedure was used in extensive experiments regarding induction, with separate implementations of concept learning systems (Hunt et al., 1966).

The general procedure is as follows:

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

To avoid over-fitting in the presence of noise, the procedure can be terminated earlier by applying pruning techniques. The heuristic measure plays a major role in deciding the quality of the formed decision tree. It also assists in the formation procedure by selecting 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 that it allows easy understanding and explanation by providing visualisation for users.

The divide-and-conquer approach has a number of deficiencies. A similar sub-tree may exist many times. 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.

2.2.1.1 ID3 group (all ID group)

All ID group is a decision tree system that utilises the information gain standard in order to split the nodes of a decision tree. Gained information is then ‘entropy-measured’ in order to distinguish the impurities while collecting examples from the data set. This process is explained in more detail below. For a sample data set, S, the entropy is defined as:

$$(S) \equiv -\sum_{j=1}^{k} p(C_j, S)\log_2 P(C_j, S) \qquad (2.2)$$

Let a test *T* with *B* outcomes separate the data set *S* into further sub-groups defined as *S1, S2, S3, S4* and so on. The total entropy of the separated data set can then be represented by the weighted entropy of the subsets, as given below. The number of instances within *Si* and *S* are [*Si*] and [*S*] respectively. The information that is gained by separation of the data sets of test *T* is given by:

$$Gain(S,T) = EntropyS - Entropy(S and T) \qquad (2.3)$$

In the above equation, gain (*S, T*) represents the expected reduction in the entropy after the separation of the data sets into exclusive sub-sets on the basis of test *T*. Braha (2001) stated that the gain standard selects the most appropriate test to minimise the information. According to Monostori (2002), ID3 selects the nodes for the growth of the decision tree in accordance with the entropies or information contents of the selected data set. The use of the entropies helps in finding solutions to the problem by providing coherent and effectual ways to construct the decision tree.

2.2.1.2 CART

Cervone et al. (2001) described CART, a binary decision tree algorithm that is expansively used. Table 2.1 shows a training data set while Figure 2.3 shows the decision tree developed from the data in the table.

Table 2.1: Training set for the weather problem (Witten and Frank, 2005).

No	*Outlook*	*Temperature*	*Humidity*	*Windy*	*Play*
1	sunny	hot	high	false	no
2	sunny	hot	high	false	no
3	overcast	hot	high	false	yes
4	rainy	mild	high	false	yes
5	rainy	cool	normal	false	yes
6	rainy	cool	normal	true	no
7	overcast	cool	normal	true	yes
8	sunny	mild	high	false	no
9	sunny	cool	normal	false	yes
10	rainy	mild	normal	false	yes
11	sunny	mild	normal	true	yes
12	overcast	mild	high	true	yes
13	overcast	hot	normal	false	yes
14	rainy	mild	high	true	no

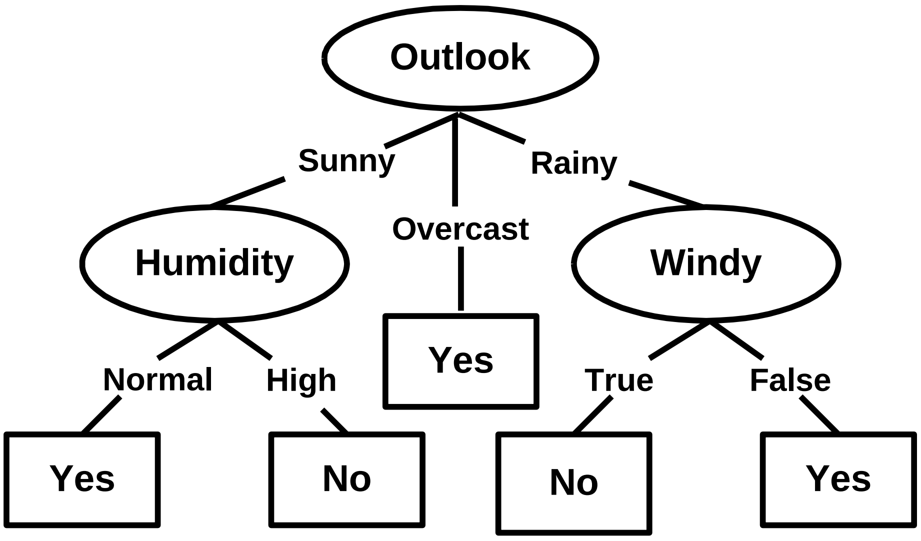


Figure 2.3: The resultant data set in Table 2.1 represented as a decision tree.

Cervone et al. (2001) explained that an insignificant characteristic *Ai* can have all the possible values (A_i, n), i.e. c_i1, c_i2, c_i3, c_i4, etc. Within A_i there are different branches that are invented by internal nodes. To assess the continuous characteristics of A_i, a binary test should be carried out on each branch, A_i and c_i. A second branch corresponding to $A_i > c_i$ is an entrance to the sphere of A_i. Every leaf node represents the classification that should be assigned to an example. Cervone et al. (2001) stated that a separate path from the root of the decision tree to its leaf node is identified on the basis of the characteristics of the example in order to classify that example. The classes of a leaf node represent the predicted class of the example. It was further explained that the evaluation function that is used for partitioning in CART is known as the Gini Index:

$$Gini(S) \equiv 1 - \sum_{j=1}^{k} p(C_j, S)^2 \tag{2.4}$$

Wu et al. (2007) said that decision trees are generated in a hierarchical manner from the training data. The first stage of the decision tree is assigned to the examples of the data set. If all the examples are of the same class then there is no need of partitions, as the solution is complete, but if the examples relate to different groups then there is a need to test those examples in order to split up the decision tree. Cervone et al. (2001) elaborate that CART is a form of classification and regression, also known as HODA (Hierarchical Optimal Discriminant Analysis). CART is a generalisation of optimal discriminant analysis. It is used to identify the statistical mode with the maximum accuracy of predicting the value of a dependent variable in a data set containing both continuous and categorical variables. CART is a learning technique that is non parametric and produces classification or regression of decision trees on the basis of the dependent variable categorically, numerically or respectively (Han, 2001). Decision trees can be formed by collecting rules on the basis of the variables in the sample data set. In CART, rules for the values are selected on the basis of dependent variables so that the best split can be achieved in differentiation of observations.

Han (2001) further explained that after selecting the rule and splitting the nodes of a tree, the same process is applied to the related sub-nodes. It was further explained that the splitting process is completed after the detection of the nodes. Every branch of the decision tree ends at a terminal node and each and every observation falls into a single terminal node.

2.2.1.3 C4.5 and C5.0 (all C groups)

C4.5 is an extension to ID3 and is also an important decision tree algorithm. This algorithm utilises the gain ratio standard as it has a strong bias in favour of the characteristic tested along with various variables. In order to reduce the bias in the gain ratio standard, the split information standard is employed via the following equation:

$$SplitInformation(S,T) \equiv -\sum_{i=1}^{k} \frac{|S_i|}{|S|} . \log_2\left(\frac{|S_i|}{|S|}\right) \tag{2.5}$$

According to Lavrač et al. (2004), a split information standard can be observed by choosing the given characteristic as a test. However, this technique reduces the choice of characteristics with various values. The given ratio then provides:

$$GainRatio(T) = \frac{Gain(S,T)}{SplitInformation(S,T)} \tag{2.6}$$

The gain ratio calculation for a supposed characteristic test is comparatively undemanding. For continuous characteristics, the number of possible values in a subset interlinked with the internal node is determined. After this, all the possible splits on a continuous characteristic are examined effectively. The split that has the tendency to maximise the gain ratio method is then chosen as the entrance. As discussed previously, the decision tree has the potential to provide an over fitted solution as it contains components that are accurate to noise and can easily present the training data. To overcome this situation, the C4.5 algorithm employs the pruning method to simplify the overall tree while eliminating sub-trees that are too specific. The pruning method can be performed by examining sub-trees which can then be replaced with one of its branches, if it does not demean the precision of the sub-tree.

According to Lavrač et al. (2004), the C5.0 algorithm is a decision tree process that is widely used in machine learning. After the development of C4.5, the C5.0 algorithm was developed to respond to noise and to easily locate the missing data in data sets. This algorithm provides boosting (Giudici, 2003). A large decision tree can be difficult to handle or read with other algorithms, but C5.0 can easily handle large data sets. While employing C5.0, an individual can easily understand a large decision tree as C5.0 views large decision trees as a group of rules. C5.0 solves the over fitting problems and reduces errors in pruning techniques. C5.0 can easily predict the characteristics that are relevant to the classification process. This technique of predicting the relevant characteristics in the classification process is known as Winnowing, and it is useful in dealing with high-dimensional data sets. In the C5.0 algorithm the input is the example data, target or attributes, while the output is the decision tree, which is constructed by use of training data sets. The decision tree classifies the data accurately and handles both continuous and discrete characteristics. According to Witten and Frank (2000), C5.0 is more efficient than C4.5. It is more accurate and its rule sets have lower error rates, though both of these methods can have similar accuracy rates.

2.2.2 Rule-based Algorithms

According to Giudici (2003), the rule-based induction method is one of the most important machines learning techniques as it can express the regularities regarding rules that are frequently hidden in the data. It is the most fundamental tool in the data mining process. Generally rules are expressions of the form:

if (characteristic 1 is equal to value 1) *and* (characteristic 2 is equal to value 2) *and* (characteristic n is equal to value n) *then* (decision will be equal to value).

Witten and Frank (2000) stated that some rule induction systems provoke more complex rules, in which the characteristic values are expressed by the contradiction of some other values or by a value of the overall subset of the characteristic domain. It was further explained that the data by which the rules are provoked are generally presented in a form similar to a table that shows different cases (rows) against the variables (characteristics and decisions). Lavrač et al. (2004) said that the rule induction belongs to supervised learning, and all of it cases are pre-classified by experts. In simple words the decision values are assigned by the experts in this process. Lavrač et al. (2004) further elaborated that the characteristics represent independent values, while the decisions represent the dependent variables. The covering method represents classification of knowledge in the form of a set of rules which represent or give a description of each class. There are a number of covering method algorithms that are widely used, such as CN2 (Clark and Niblett, 1989; Clark and Boswell, 1991), RIPPER (Cohen, 1995) or AQ (Michalski, 1969). The most common versions are AQ19 (Michalski and Kaufman, 2001) and Rules Family (Pham et al., 1995). Covering method algorithms deduce rules from a training set of examples, making use of the same general procedure as used for the first time in the AQ algorithm.

This procedure makes use of the following search process to produce the rules for each class in the training set *T*:

While the Stopping Criterion is not satisfied:

- Form a new rule to cover examples belonging to a target class employing the Rule Forming Process;
- Add this rule to the Rule Set;
- Remove all examples from *T* which are covered by this new rule.
- Stop the procedure when there are no more classes to classify.

This process is further explained by applying it to the training set of weather problem in Table 2.1 (Witten and Frank, 2005) to determine the decision of the player to play or not to play golf, based on the weather conditions. Each row of the Table 2.1 represents an instance, and the whole table represents the training data set. Each instance contains the values of different characteristics and it also shows the values of the equivalent classifications.

A possible classification set that can be derived from the Table 2.1 is as follows. The table has four variables, two of which can take three values, while the other two can take two values. As a result, 36 possible combinations (3 x 3 x 2 x 2 = 36) are created. Out of these combinations, 14 are present in the set of input examples. From the information described above, the set of rules shown in Figure 2.4 defined as a decision list.

IF Outlook = sunny AND Humidity = high $\Rightarrow$ *Play = No*

IF Outlook = rainy AND Windy = true $\Rightarrow$ *Play = No*

IF Outlook = overcast $\Rightarrow$ *Play = Yes*

IF Humidity = normal $\Rightarrow$ *Play = Yes*

IF none of the above $\Rightarrow$ *Play = Yes*

Figure 2.4: The resultant data set in Table 2.1 represented as a decision list.

The rules that have been described so far have the ability to make a prediction on how the examples can be classified in terms of whether to play or not to play. These rules are acknowledged as classification rules.

2.2.2.1 AQ

AQ is a rule induction algorithm which was developed by Michalski, et al. in 1970's. Various algorithms based on this have been developed by other people. In AQ, *A* is the set of all the characteristics *A1, A2, A3 ... An*. Wojtusiak et al. (2006) stated that the seed is a member of a concept, which is an positive case. A selector is an expression relating to a variable that is a characteristic or a decision regarding the value of a variable such as a contradiction of values. The main inspiration of the AQ algorithm is the generation of cover for every concept while calculating stars and choosing single complexes for the cover from those stars. According to Cervone et al. (2010) an AQ algorithm requires the calculation of conjuncts of incomplete stars.

In worst cases, the complexity of time in this calculation is *O* (*nm*), where *n* is the number of characteristics that are being used and *m* is the number of cases. The developers of the AQ algorithm suggested the use of the MAXSTAR parameter as a tool for reducing computational complexity. They believed that the set computed by the conjunction of incomplete stars can be reduced in size if it has a greater number of members than MAXSTAR. However, this also reduces the quality of the results of this algorithm.

2.3 Clustering

Data clustering is also known as unsupervised learning, which can be defined as dividing a set of data into various homogeneous clusters. Clustering algorithms help to allocate the large number of data points into a smaller number of groups. The reason for assigning data points into groups is to combine data points which have similar properties, while separating data points that are dissimilar. Clustering forms a part of active research in various fields such as machine learning, statistics, data mining and pattern recognition. Pham and Afify (2007) described clustering as an important technique relating to data exploration. Clustering includes various applications in diverse areas of engineering including manufacturing, system design, engineering design, production planning, quality assurance, modelling, control, and monitoring. Mirkin (2005) stated that engineering analysis, grouping of web pages market segmentation, scientific analysis and information retrieval are all included in the applications of clustering. Clustering can also be very useful in exploratory analysis and data cleaning. According to Wu (2012), in the case of identifying pattern classes, clustering can be used as pre-processing step for supervised classification. Wagstaff et al. (2001) said that representation of data puts clustering in a historical perspective that relates to statistics, numerical analysis and mathematics. Teknomo (2007) said that clusters respond to hidden patterns; the search for clusters is unsupervised learning, and the results represent the concepts of the data.

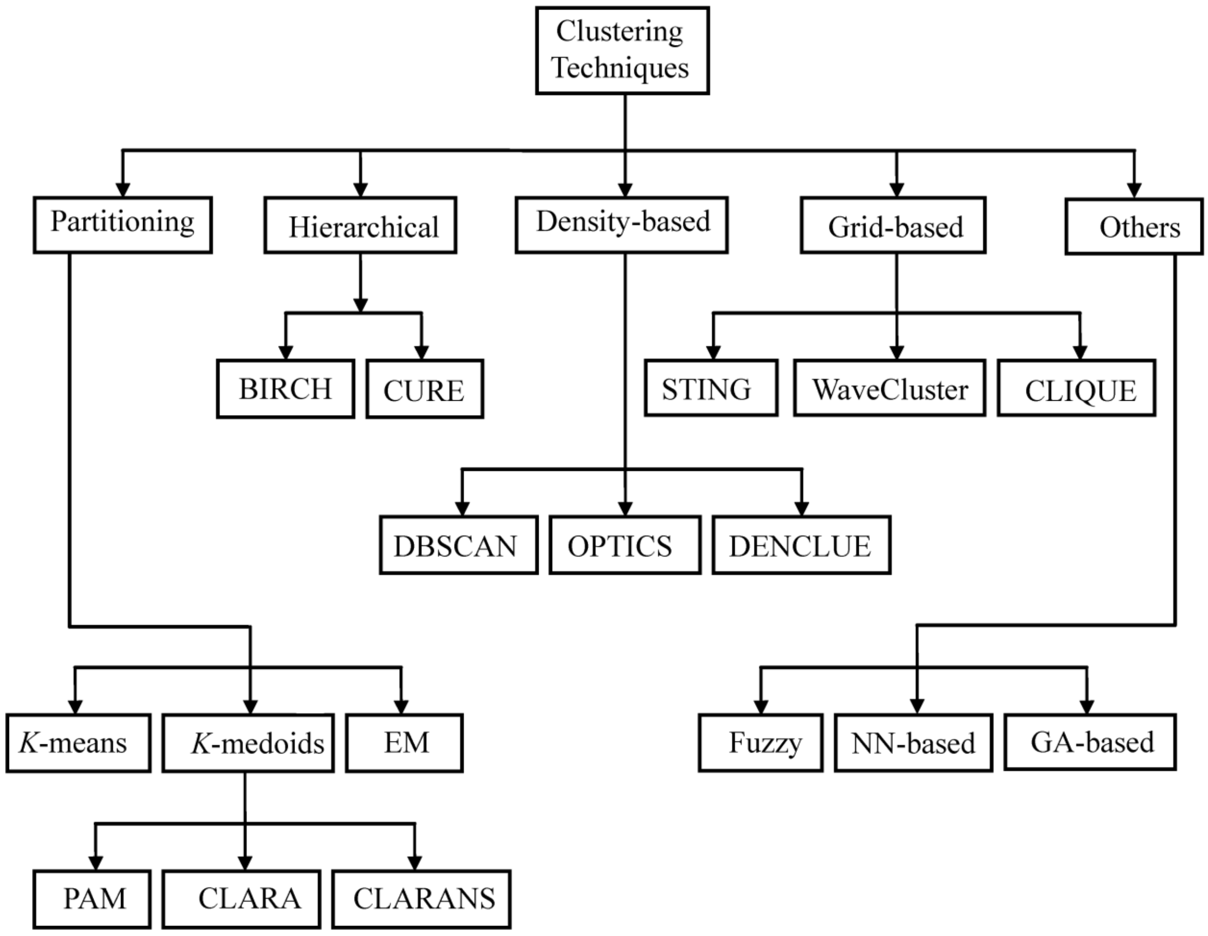


Figure 2.5: A classification of the main clustering techniques. (Pham and Afify, 2007)

2.3.1 Hierarchical methods

According to Rokach and Maimon (2010), hierarchical clustering, also called Hierarchical cluster analysis (HCA), develops a hierarchy of clusters. Ferreira and Hitchcock (2011) said that hierarchical methods generate a nested sequence of 1 to *N* clusters for a data set of size *N*. Rani and Rohil (2013) stated that this method groups points of data into a tree of clusters having one cluster at each root of that tree. Rani and Rohil (2013) further defined this tree of clusters as a dendrogram. Murtagh and Contreras (2011) described a tree of clusters as a tree containing different data points, with one cluster at each of its roots. At the roots of each cluster there are various leaves. There are *N* clusters, and each cluster contains one data point. According to Halkidi et al. (2001), there are two approaches to the hierarchical method: the agglomerative approach and the divisive approach. Rokach and Maimon (2010) explained the agglomerative approach as one that operates in a bottom up manner while performing a series of agglomerations of small clusters where each cluster has a single data point. Rani and Rohil (2013) said that the agglomerative approach combines these small clusters to form different larger clusters. According to Halkidi et al. (2001), there are three methods that can be used to elaborate the distance between two or more clusters. These are: centroid based, single link and group average. According to Madhulatah (2012), the centroid based method calculates the contrast between the center of two clusters. The single link method calculates the shortest distance between two points of data, while the group average method calculates the distance between all pairs of data points in different clusters.

BIRCH

BIRCH is the acronym for Balanced Iterative Reducing and Clustering using Hierarchies. According to Halkidi et al. (2001), BIRCH is a multi-phased clustering algorithm in which large groups of data sets are clustered without being affected by the available memory of the computer. Rani and Rohil (2013) said that BIRCH utilises a data structure known as CF tree (clustering feature tree) which sums all the data points before the application of the hierarchical clustering algorithm. Halkidi et al. (2001) elaborated that clustering can be executed quickly with the help of a CF tree. There are two types of node in a CF tree: internal nodes and leaf nodes. Rokach and Maimon (2010) explained that every leaf node carries almost *A* entries and each entry relates to a cluster which can be represented as a tuple:

$$\left(n_j, \sum_{i=1}^{n_j} x_i^{(j)}, \sum_{i=1}^{n_j} (x_i^{(j)})^2\right) \tag{2.7}$$

where n_j the number of data points in a cluster, *j*, and *i* is the index of the data point within cluster *j*. These tuples can also be regarded as features extracted from different data points in sub-clusters. According to Murtagh and Contreras (2011), BIRCH is a reliable method that can effectively handle noise. On the other hand, CF nodes in a CF tree do not handle a large number of data points because of their size.

CURE

CURE is an acronym for Clustering Using REpresentatives. According to Rani and Rohil (2013), it is a clustering algorithm that merges the hierarchical and sampling clustering approaches through a new distance approach. According to Madhulatah (2012), the CURE method uses the different scattered points of a cluster to represent it and to capture its shape. Halkidi et al. (2001) said that representative points are shrunk towards the centre of their cluster to avoid the noisy points before the computation of two clusters. According to Madhulatah (2012), CURE measures distance on the basis of the two closest points collected from the representatives of two different clusters.

2.3.2 Density-based method

The density based method is a clustering method in which dense regions of data points which are separated by low density regions are clustered (Jiharabadkar and Kulkarni, 2009). Yin et al. (2007) said the density method is less sensitive, but that different clusters of arbitrary shapes can be discovered by adopting this approach. According to Shah et al. (2012) there are three major algorithms which employ the density method: DBSCAN, OPTICA and DENCLUE.

DBSCAN

DBSCAN is the acronym for Density-Based Spatial Clustering of Applications with Noise. DBSCAN is an example of a density based algorithm (Jiang et al., 2010), which includes the two parameters, Eps and MinPts. Eps is a specific radius while MinPts is a certain number of data points. Chen et al. (2013) said that a specific radius must have a certain number of data points. According to Cao et al. (2006), there are many steps of DBSCAN and every step builds up a cluster. Yin et al. (2007) stated that the algorithm in this method does not relate to any other cluster previously discovered. According to Jiharabadkar and kulkarni (2009), DBSCAN handles a large amount of data and its processing order does not affect the shape of the cluster. Cao et al. (2006) elaborated that the main disadvantage of DBSCAN is that it is difficult to establish Eps and MinPts, and the result is very sensitive for these two parameters. Cao et al. (2006) mentioned that a further drawback of DBSCAN is that it cannot handle data that contains clusters with diverse densities.

OPTICS

OPTICS is the acronym for Ordering Points To Identify the Clustering Structure. According to Yin et al. (2007) the reason for proposing the OPTICS method was the sensitivity of DBSCAN. As described previously, DBSCAN is sensitive to the input parameters (Eps and MinPts) so it was proposed to order data points to identify the clustering structure. This would overcome the weakness of DBSCAN. According to Jiharabadkar and kulkarni (2009), the OPTICS method is a clustering method that calculates enlarged cluster ordering for automatic cluster analysis. Jiang et al. (2010) said that OPTICS requires input parameters just like DBSCAN, but that this method produces clustering results for a single pair of parameter values. Shah et al. (2012) said that values of both parameters can be computed and visualised easily and effectively by adopting this method of clustering. However, a limitation of OPTICS is that it cannot handle high-dimensional data (Yin et al., 2007).

DENCLUE

DENCLUE is the acronym for DENsity- based CLUstEring. According to Yin et al. (2007), the DENCLUE method of clustering is used for handling high dimensional data. Jiang et al. (2010) further elaborated that in this method the overall density of a cluster is analysed based on the influence of the data points in the cluster. This method measures the effects of data points on their surroundings. Shah et al. (2012) said that the DENCLUE method is adopted in order to efficiently compute the effects of influence functions. Jiang et al. (2010) said that this method of clustering helps with the mathematical identification of clusters by finding local maxima of the density function. According to Shah et al. (2012) the major advantage of the DENCLUE method is that it is resistant to noise and can handle large groups of clusters. According to Yin et al. (2007), this is an effective method compared to the other density-based algorithms. Nevertheless, for this method the clustering parameters should be carefully selected as they can influence the quality of the results.

2.3.3 Grid-based methods

According to Park and Lee (2004), the grid based method indirectly constructs different summaries of data. This method constructs a summary of an attribute space. According to Rama et al. (2010), each attribute space is divided into predetermined cells, forming a grid structure through which all the clustering functions are performed. Madhulatha (2012) was of the view that grid-based methods handle outliers well. Ansari et al. (2013) stated that this method of clustering is ineffective because of the increased number of dimensions. In this method the number of dimensions is increased because summarised information is used.

STING

STING is the acronym for STatistical INformation Grid. Park and Lee (2004) said that a statistical information grid (STING) is an approach that is used for spatial data mining. Park and Lee (2004) stated that in this approach a spatial area is divided into numerous rectangular cells that form a hierarchical structure. Park and Lee (2004) said that in this method statistical values such as the mean, minimum and maximum are pre-computed and stored in each grid cell. This process uses a top down approach to work through the hierarchical structure. The relevancy of each and every cell is examined by the algorithm (Madhulatha, 2012). According to Ansari et al. (2013), relevancy is the proportion of data points in a cell that satisfy the conditions of the query. The STRING process examines each and every level of hierarchical structure starting from its top. It is a fast and effective approach that allows the user to efficiently retrieve data from different cells.

Wave Cluster

Wave cluster is the acronym for wavelet based clustering. Rama et al. (2010) explained that this approach is a multi-resolution algorithm in which data points are clustered by a wavelet transform method. According to Shehada et al. (2012), the wavelet transform method processes signals and then decomposes them into various frequency sub-bands. Chang et al. (2009) said that the wave cluster method relates to data points of multi-dimensional signals. Park and Lee (2004) said that the signal processing method is adopted to discover the high and low frequency components of multi-dimensional signals that represent a feature space. Rama et al. (2010) said that this method used to detect the clusters within different grids. Ansari et al. (2013) stated that this method evaluates the results in multi-resolution clusters from different scales by applying wavelet transforms multiple times. According to Chang et al. (2009), the wave cluster method is insensitive and efficient when it comes to the processing of data, and it helps to find clusters of arbitrary shape with complex structure. This approach also handles outliers well. Park and Lee (2004) said that this approach cannot be applied to low dimensional data.

CLIQUE

According to Chang et al. (2009), clustering of high dimensional space is an approach that allows user to cluster high dimensional data space. In this approach, sub-spaces of a high-dimensional data space are automatically identified by the algorithm. Ansari et al. (2013) stated that the CLIQUE approach allows a user to efficiently cluster data points from an original space. Shehada et al. (2012) said that this method partitions a data space into rectangular units that do not overlap with each other, and that this method also discovers the number of points in a cell. The dense rectangular units are then further examined by employing a depth algorithm to determine the clusters. Ansari et al. (2013) said that this clustering of high-dimensional space is an approach that is sensitive to the input of data points. This method is a simple, but the accuracy of the results may be degraded by using this approach.

2.3.4 Other approaches of clustering

While some of the most renowned and commonly used methods of clustering have been discussed in the earlier sections, this section will present some alternative approaches, including:

- Fuzzy clustering
- NN (natural network) based clustering
- GAs (genetic algorithms) etc.
- MAXNET clustering

Fuzzy clustering was described at the International Federation of Classification Societies Conference (2004) as setting clustering algorithms while incorporating an element of uncertainty into the process. The reason that fuzzy clustering is gaining the attention of the practitioners is the fact that the traditional clustering algorithms generally end up providing non-overlapping clusters which means that data points clearly belong, or do not belong, to a given cluster. Fuzzy clustering, on the other hand involves algorithms that provide the margin of association or degree of attachment of a data point to a cluster. This means that an uncertainty element is imbedded, such that a data point does not need to be described as either belonging to a cluster or not, and the degree of relation is best suited to resolving the uncertainty aspect. FCM is the most commonly applied algorithm for fuzzy clustering.

The element of learning ability in the clustering techniques is what makes the foundation of natural networks (NNs). Use NNs, in particular artificial NNs, has increased considerably over the last decade, as has its application throughout the engineering disciplines. Prominent examples of NNs are self-organising maps (i.e. SOM) and adaptive resonance theory (ART). NNs like SOMs have two layers referred to as input and output layers, where neurons of the input layer are connected to those of the output. Networks learn to recognise the patterns in the input layer to adjust the output accordingly. ART 1 had been launched for the purpose of clustering binary inputs while ART 2 was further advanced for perpetual inputs. Use of the fuzzy method in ART made it possible to respond to both binary and continuous input values. An advantage of NNs is their increasing use and applicability across the board, but a disadvantage is the need to gather suitable patterns.

Genetic algorithms (Gas) operate on the principle of evolution, and they tend to search for the best available option from the population. Various GAs have been developed for solving clustering problems in industry. A GA system comprises a solution, i.e. typical, valid data bifurcations and the cluster centroids. They are represented by bit strings. Finding the appropriate solution is initiated by populating the solutions, and if the problem is not resolved then new solutions with operations comparative to the existing solutions are developed. If the solution is achieved as desired then it becomes part of the population for the subsequent stages.

MAXNET is a self-organising neural network model that calculates the highest number of *n* given values. Self-organisation implies that despite the cluster having no prior supervisory information, it still learns from the input and to order the neurons correspondingly. It could be ascertained that self-organisation operates on similar principles of sensory paths as those of human visual system. In this case, self-organisation is regarded as a useful ordering of the neurons in the brain. For 'ordering' that does not need the neurons to be physically moved, a commonly known self-organising neural network model is referred as feature maps (Kohonen, 1989).

MAXNET is a recurrent network. It has a single layer of *n* nodes (Figure 2.6) which compete to ascertain the node with the maximum initial value. Each node is adjoined to all other nodes embodying itself. The network executes an iterative process whereby each node gets inhibitory inputs from other nodes through *lateral* (intra-layer) associations. All the nodes update their output in the meantime (in parallel) [Mehrotra et al., 1997; Datta et al., 2000].

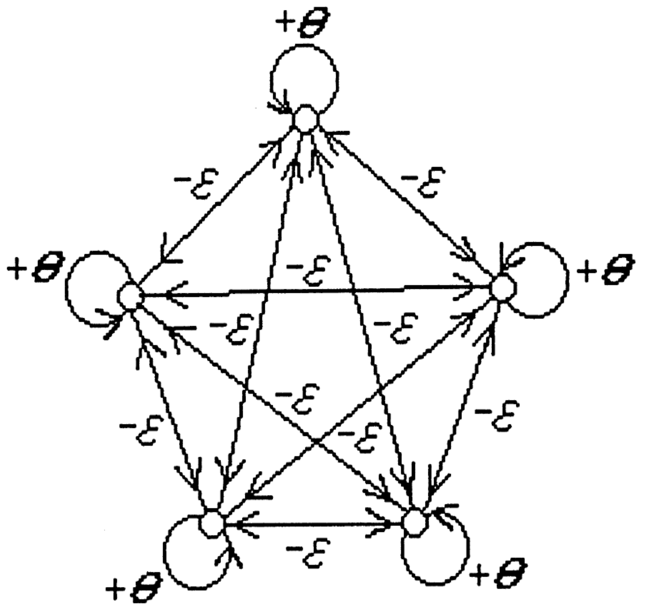


Figure 2.6: MAXNET architecture (Datta et al., 2000).

Thus MAXNET allows the parallel computation of the maximum value from a given set of values, where every computation is local to each node rather than being controlled by a central processor. It can be seen that the number of iterations to select the winning node does not depend on the number of data points, *n*.

2.4 Data engineering

According to Chan et al. (2009), data engineering is the multi-corrective practice that is used in engineering computer software and computing systems to help the user by extracting information via data analysis. Chan et al. (2009) said that data engineering generates data in various ways. Data engineering employs different techniques for analysing data, including machine learning, pattern recognition. It also uses various techniques (optimisation, visualisation, prototyping, knowledge elicitation and different database systems) for efficient extraction of data. The main objective of data engineering is to generate additional data from the available data and in doing so to understand the overall process that is being investigated. The main characteristic of data engineering is the procedure of analysing data and creating new tools for a specific task.

2.4.1 Data Pre-processing

According to Axelson (2012) data pre-processing is an essential step in the data mining process. Han et al. (2011) stated that the phrase “Garbage in and Garbage out” is predominantly applicable to machine learning and data mining processes. He further explained that data collection techniques are mostly controlled by the users which gather incomplete values and unfeasible data combinations. Axelson (2012) stated that if the data is not being carefully analysed then it might corrupt the results. Therefore, the the representation of quality data is the most important step and should be carefully executed before analysing the data for a particular task. Han et al. (2011) stated that if inappropriate and unnecessary data is collected by the user then this will affect the overall process and will make it difficult for the user to discover knowledge about a particular task. Data pre-processing contains transformation, normalisation, data ordering, data selection and data cleaning.

2.4.2 Data Noise

According to Zhou et al. (2007), data noise relates to the meaningless data that has no value to the user and is unnecessary for the user. Noisy data is also known as corrupt data. Zhou et al. (2007) explained missing data as the data that is construed correctly by machines like unstructured data in the form of text. Noisy data can be defined as data that is received, stored and changed, so that it cannot be used by any other the program except the original program that created it. Data noise causes many different problems for users and it unreasonably increases the storage space that is required for the task. It also strongly hinders the results of the data mining process. Zhou et al. (2007) stated that data noise can cause programming errors, hardware failures and many other problems, and makes it difficult for the user to efficiently complete a task.

This section will discuss the problems of data noise might pose for the rules family building procedure that has been described earlier?

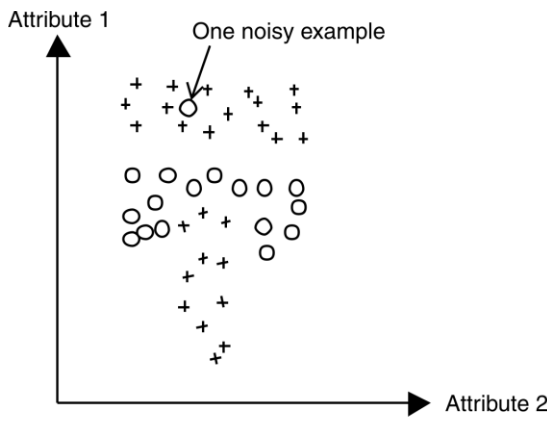


Figure 2.7: Training data containing noise (Bigot, 2002).

Taking into consideration the small training set for the weather problem shown in Table 2.1, the assumption is that the attribute ‘Outlook’ of example 1 is ‘Overcast’ and that this has been recorded incorrectly. Examples 1 and 3 will belong to different classes but then have identical descriptions and as a result the attributes for this training set become inadequate. Furthermore, if the ‘Windy’ attribute of example 4 is corrupted to true then the attributes will become inadequate because the example would then contradict example 14. Finally, the first training set can be considered by the simple rule set of Figure 2.3, comprising of 5 rules, has been produced by both RULES-3 Plus and RULES-5 Plus. Considering that the class of example 3 were corrupted to ‘No’, an appropriate rule for this corrupted training set would now have to give an explanation for the special case of example 3. More rules are produced for both RULES-3 Plus and RULES-5 Plus as they produce 9 rules and 7 rules precisely in the given order.

Figure 2.8 and Figure 2.9 demonstrate results which highlight two problems: errors in the training set (i) may lead to rulesets of spurious complexity, or (ii) may cause the attributes to become inadequate.

IF Outlook = sunny AND Humidity = high ⇒ Play = No

IF Temperature = hot AND Humidity = high ⇒ Play = No

IF Outlook = rainy AND Temperature = mild AND Windy = false ⇒ Play = Yes

IF Humidity = normal AND Windy = false ⇒ Play = Yes

IF Outlook = rainy AND Humidity = normal AND Windy = true ⇒ Play = No

IF Outlook = overcast AND Humidity = normal ⇒ Play = Yes

IF Outlook = sunny AND Humidity = normal ⇒ Play = Yes

IF Outlook = overcast AND Windy = true ⇒ Play = Yes

IF Outlook = rainy AND Temperature = mild AND Windy = true ⇒ Play = No

Figure 2.8: The rule set produced by RULES-3 Plus over a noise class in example 3.

IF Outlook = sunny AND Humidity = high ⇒ Play = No

IF Outlook = overcast AND Temperature = hot AND Humidity = high ⇒ Play = No

IF Outlook = rainy AND Windy = false ⇒ Play = Yes

IF Outlook = rainy AND Windy = true ⇒ Play = No

IF Outlook = overcast AND Humidity = normal ⇒Play = Yes

IF Outlook = sunny AND Humidity = normal ⇒Play = Yes

IF Outlook = overcast AND Temperature = mild ⇒ Play = Yes

Figure 2.9: The ruleset produced by RILES-5 Plus over a noise class in example 3.

2.4.3 Missing data

According to Han et al. (2011), missing data is an empty cell in a table representing a data set.

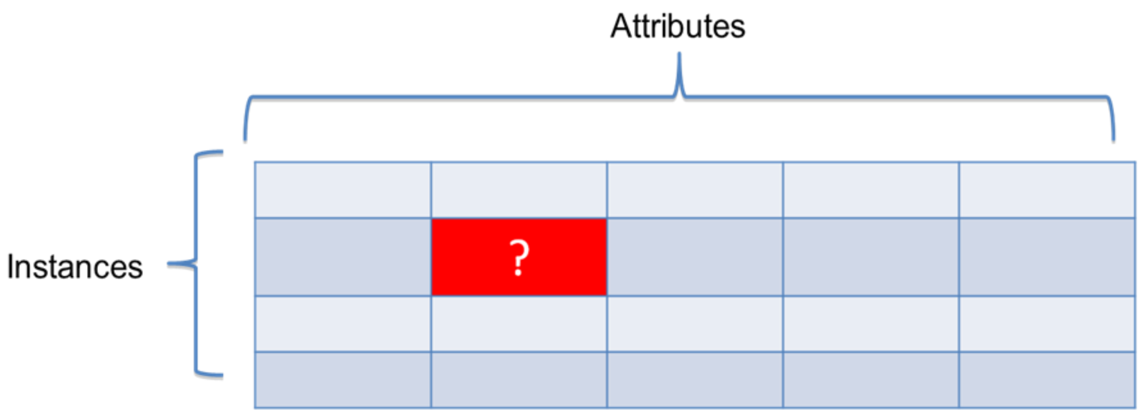


Figure 2.10: Data set including missing data.

Han et al. (2011) further stated that there are numerous ways that concealed values can occur in a data set. The most common prospect is that someone intentionally enters false values into the data set. On the other hand, default values can also become a concealed source of missing data. For example if an online form has the default gender as male and the default country as United Kingdom then if a person filling in the form does not want to share their identity or personal information it may lead to the missing values being disguised as default values. Klösgen et al. (2002) stated that the most vital source of missing data is the lack of standard missing data representation. It was further explained that even a single file might have multiple codes representing the same missing data. Every organisation has its own approach to the representation of the data that leads to disguised missing data, and organisations can reduce the fake entries in data sets by developing a standard approach for the representation of such data.

2.5 Present Progress in Machine Learning Research

Bell (2014) states that the research upon machine learning is making considerable progress in many directions. This section of the thesis will evaluate two core directions and also current problems will be discussed in this section. The two directions that will be discussed in this section are selling up of the machine scale and learning multiple methods.

2.5.1 Scaling up Machine Learning Algorithms

The core research area concerns to the techniques for scaling up of the machine learning algorithms so that they can process the large amount of data sets efficiently, while developing best possible models from them. Brownlee (2013) stated that the recent emergence in the data mining process as the major application for the machine learning has significantly increased the overall need of the algorithms to handle complex and large amount of data sets that are currently beyond their scope. Brownlee (2013) further stated that the data mining process includes the data sets having millions of training examples and also hundreds and thousands of attributes and classes. Barber (2012) stated that currently these processes represent the databases containing data in gigabytes or even terabytes. So the development of effective and suitable applications for handling such data has been increased among researchers.

Busse (2005) stated that there are numerous approaches that are presented or implemented by the experts for scaling up of the machine learning algorithms and one of the most effective and straight forward application is to produce more efficient algorithms so that the overall efficiency of the existing algorithms can be enhanced. In regards to this approach Flach (2012) stated that it includes a wide range of algorithm design techniques so that the overall representation and search can be optimised and also to find the approximate rather than exact solutions. According to Koyu & Deshpande (2014) another common approach for scaling up of the machine language is to partition the data while avoiding the need to run the algorithms in very large scale data sets. In this regard Barber (2012) and Bell (2014) stated that this approach has proved to be advantageous for scaling up of the machine learning as it includes the process of breaking the data sets in to further subsets, while considering one or more subsets and also by combining the results. According to Freitas (2002) the partitioning of the data helps in avoiding or reducing the memory management problems that occurs when an algorithm attempts to process huge data sets from the main memory. In this regard Murphy (2012) further stated that an approach orthogonal to the selection of the examples from the subsets helps with the selection of the relevant features upon which there is a need to focus. Mohri et al (2012) states that for focusing on the specific details the application of the inductive learning techniques for handling of the large data sets has now been reviewed by most of the experts and the issues and techniques are being discussed generalised to other machine learning techniques.

Maimon and Rokach (2006) stated that the decision tree algorithm has been improved for handling of the large data sets efficiently and also allot of new algorithms have been proposed by the experts during last few years. In 1991 Catlett has proposed two methods in order to improve the time utilised while developing the classifier. The first method that was proposed by Catlett uses data sampling at each node of the decision tree and the second method that was proposed focuses on discretised continuous attributes. In this regard Sra et al (2012) stated that these two methods have significantly reduced the overall time for the development of the classifier but on the other hand these methods have reduced the overall accuracy of the classification. Furthermore Catlett has only considered the data sets that may fit in the main computer memory. While on the other hand, Maimon and Rokach (2007) stated that these methods have significantly helped in enabling overall classification of the large data sets. In regards to the incremental learning methods Mohri et al (2012) stated that incremental learning methods are the methods in which the data is classified into batches and the cumulative cost of classification of the data may sometimes exceed the overall cost of classifying the overall training sets at once. Perner (2009) stated that the decision tree classifier is known as SLIQ that uses the novel techniques for the pre sorting purposes, MDL based pruning and breadth first growth in order to improve the overall learning time of the classifier without compromising with its accuracy. As this classifier uses the memory resident data structure that scales with the size of the training set and SLIQ has the upper limit on various examples that it can process. Maimon and Rokach (2006) stated that in 1996 Shafer et al has presented an classification algorithm that is known as SPRINT and in an efficient algorithm that helps with the removal of the memory restrictions that further restricts the existing decision tree algorithms while exhibiting the same excellent behaviour as SLIQ. Maimon and Rokach (2006) further stated that the SPRINT algorithm significantly helps with the classification of the large data sets and it also can be easy and efficiently parallelised. However there are some disadvantages of SPRINT algorithm as well such as it uses the data structures known as attribute lists which is an costly method and includes the potential tripling of the overall size of data set. Maimon and Rokach (2006) further stated that just like C4.5 both SPRINT and SLIQ are the two staged algorithms that include the development and the pruning phases. He further stated that the development of the decision tree in two distinct phases can be tricky and complex which can result into the substantial amount of waste effort as the first phase requires the development of an entire sub tree which is then pruned in the next phase.

According to Perner (2009) PUBLIC is also a decision tree classifier that significantly integrates the overall pruning process into the development phase rather than performing them one after another. Bell (2014) stated that the integrated approach for the PUBLIC algorithm can significantly enhance the overall performance in comparison to the traditional classifiers like SPRINT. According to Maimon and Rokach (2006) in 1998 Gehrke has proposed and Rainforest which is an framework for the development of a fast and effective algorithms for developing the decision trees that can efficiently be adjusted to the amount of the available memory. Maimon and Rokach (2006) further stated that after Gehrke in 2998 Morimoto has also developed algorithms for the development of the decision tress for categorising the attributes with large domains. The overall aim behind development of this algorithm was to enhance the overall quality of the resulting tree. There are various other rule induction algorithms associated with the decision tree algorithm that can significantly help with the scaling of the large data sets. Maimon and Rokach (2006) further stated that IREP was the rule that was developed in 1994 which was a rule learning algorithm that has the potential to handle the complex and noisy data efficiently. Maimon and Rokach (2006) further stated that the core reason behind the efficiency of this rules was that it utilises the technique that is known as incremental reduced error pruning. In this regard Perner (2009) stated that this technique helps in pruning each and every rule immediately after its introduction rather than pruning after generating all of the rules. This process further increases the overall speed of the induction process as the pruned rules helps with the removal of the larger subsets. However, the overall accuracy of the class depiction in this algorithm is lower in comparison to that of C4.5.

According to Maimon and Rokach (2007) in 1995 Cohen has detailed some modifications in IREP in order to improve the overall accuracy of its results and this process included the rule evaluation criterion, posting processing optimisation operation, stopping criterion and production of an algorithm named RIPPER. Maimon and Rokach (2007) further states that RIPPER algorithm is compatible with the rues of C4.5 in regards to the error rates and it also helps in maintaining the overall efficiency of the IREP. Barber (2012) also stated that RIPPER algorithm supports the missing characteristics, multiple classes and Continuous variables which further make it applicable to a wide range of benchmark problems.

2.5.2 Learning Multiple Models

The second active area of concern for this research is the particular method for improvement of the overall accuracy of the results in the supervised learning. Freitas (2002) states that the term multiple methods is also known as ensemble of classifiers and it is generally used for identification of the sets of classifiers and their individual decisions are combined with the classification of the new examples. Freitas (2002) further stated that multiple methods are more accurate in comparison to the individual classifiers and it also has substantial theoretical foundation. In this regard Murphy (2012) stated that an ensemble is more accurate in comparison to its component classifiers only of the individual classifier is diverse and accurate. While on the other hand Sra et al (2012) explained accurate classifier as the classifier that performs better than a random guessing. Sra et al (2012) further stated that two classifiers can be regarded as the diverse classifiers only of they make different data points or different errors.

Bell (2014) stated that there are numerous methods that have been proposed by the experts for obtaining the multiple classifiers while using the same learning algorithm. He further stated that most of the methods were manipulating the training sets so that the multiple hypotheses can be generated. Sra et al (2012) stated that in these methods the learning algorithms runs multiple times and each time it utilised diverse distribution of the training instances. Sra et al (2012) further stated that this technique performs better with the unstable learning algorithms. He further explained unstable algorithms as the algorithms in which the output classifier endure some major changes in response to the small changes in the training data. Maimon & Rokach (2007) stated that there is another classifier that is constructed from the learned classifiers by a weighted voting scheme through which each and every component of the classifier contributes to the final classification having a different strength on the basis of its accuracy on the training instances that was trained with boosting and bagging which is further dependent upon the instability of the boosted learning system. Maimon and Rokach (2006) stated that another technique for carrying out the ensemble of the classifiers which is to manipulate the set of classes in the training sets that are given in the learning algorithms. Maimon and Rokach (2006) further stated that in 1995 Bakiri and Dietterich has introduced an technique named error correcting output coding which is an efficient technique that helps in handling multi class problems while solving the multiple two class problems. Maimon and Rokach (2006) further states that ECOC technique represents the classes having the set of output bits and in this technique each and every bit encodes an binary classification function that further corresponds with an unique partition of the classes. In this regard Flach (2012) states that the algorithm that utilises ECOC technique can learn the function in correspondence to each and every bit and all of the functions are then combined in order to generate the class predictions.

Koyu and Deshpande (2014) stated that ECOC technique helps in generating the ensembles while using the single learning algorithm and there are also some other techniques that can help with the production of the ensemble while combining the classifier developed through different learning algorithm. He further stated that the diversity is implied when different learning algorithms are combined. So they just need to be checked for their accuracy while employing some of the weighted combination and this technique for development of the ensembles has been proved to be advantageous in most of the applications.

2.6 Summary

This chapter reviewed the history of machine learning and data mining. It has covered the evolution of learning, with an in-depth look at the developments in the processes of machine learning and data mining.

The chapter initially inspected the fundamental ideas behind inductive learning algorithms recounted the two main types of algorithm that are readily obtainable. Furthermore several examples of each type of algorithm have been outlined, and an overview of clustering algorithms from a data mining perspective has been presented.

Finally the chapter discussed the issue of data quality during data pre-processing. Problems with data quality, such as erroneous data, missing values and mislabelled instances were outlined with the examples of these problems being reviewed and also in this chapter the recent directions of the machine learning research were presented.

Chapter 3: An experimental evaluation of using seed examples in rule learning

3.1 Preliminaries

It is essential to gain adequate knowledge of a domain to enable the construction of a useful knowledge base for an expert system. Owing to the sudden increase in the amount of data that can now be contained in data storage devices, this operation has shown to be a bottleneck. This issue is tackled by deploying the automatic knowledge acquisition techniques (Pham, 1999).

Automatic techniques for knowledge acquisition include inductive learning, decision trees and the extraction of knowledge in the form of IF–THEN rules. An inductive learning program normally needs an input as a set of examples. Each example is distinguished by the values of several attributes, and the class to which it belongs. One approach to inductive learning is through an operation known as 'in 'divide-and-conquer', attributes are chosen according to some strategy (e.g., to increase to the greatest possible amount, or the degree of information gain) to separate the primary example set into subsets. The inductive learning program then generates a decision tree that accurately classifies the given example set. The tree represents the knowledge extracted from the distinct examples in the data set. This can be successively deployed to deal with situations not covered by the example set (Pham and Pham, 1999).

ID3 (Quinlan, 1986) and C4.5 (Quinlan, 1993) are commonly used algorithms based on divide-and-conquer. As a means to produce decision trees that infer well the information content of the example set, these algorithms initially compute the entropy of each attribute and the entropy of the entire data set. Knowledge gain is calculated with reference to each attribute. The attribute with the minimum gain is chosen as the base of the tree. Every branch emanating from the root describes a subset of examples that deviate from the parent subset. A similar procedure is iterated for all the subsets up to the point when all subsets have zero entropy. For C4.5 and ID3, it is a tedious task to construct a new decision tree at each phase and divide the examples into the various subsets connected to the branches of the newly produced tree. The 'covering approach' (Pham and Pham, 1999) uses the inductive learning program to seek groups of attributes uniquely shared by examples in given classes, and forms rules with the IF part as conjunctions of those attributes and the THEN part as the classes. The program removes the accurately classified examples from those being examined, and halts when rules have been generated to classify all the examples in the given set. This 'Separate-and-Conquer' learning rule can be separated into two main stages: in the initial stage, a single rule is learned from the data; in the second stage all the (positive) examples covered by the learned rule are taken away from the training set. The next single rule is learned from the remaining examples. The two stages are iterated as long as (positive) examples are left uncovered in the training set. This ascertains that each positive example is covered by a minimum of one rule (completeness) and every negative example is excluded (consistency). This approach is employed by the AQ algorithm and the RULES family algorithm (Aksoy, 2008), which will be described in Section 3.2.

3.2 RULES family

The RULES family of extraction systems is a series of versions of algorithms for automatic knowledge acquisition from examples. The RULES family of inductive learning algorithms follows the rule-based approach to inductive learning. The RULES family that was developed by previous researchers in the author's group is specifically described in this section.

3.2.1 RULES-1

RULES-1 is an algorithm which was developed by Pham and Aksoy, and is an abbreviation of (RULE Extraction System-1) Pham and Aksoy (1995a). This algorithm is used for extraction of classification rules for the collection of objects that belongs to the given set of classes. The algorithm uses a rule searching technique and a simple metric function that assesses the generality and accuracy of the rule. According to Aksoy (2008) an object is a fixed set of characteristics having its own possible values. For example, weather and temperature are attributes which have values such as sunny, rainy, and snowy, and high, average, and low, respectively. *Na* represents the number of characteristics then a rule may encompass. A rule therefore has between 1 and *Na* conditions which must have different values. Freitas (2002) explained that if the rule contains more than one condition then the combination of values is permitted so that all the characteristics can be different. The major drawback of RULES-1 is that it requires a long training time, especially when dealing with the problems that have a large number of characteristics and values. This is because RULES-1 employs an array of values for all the unclassified examples for a given iteration in order to extract the rules. In the worst cases the rule extraction time is over. Another drawback of RULES-1 is the selection of a large number of rules; this is because it has less control over the training data. Freitas (2002) stated that RULES-1 is incapable of handling incomplete examples and numerical values.

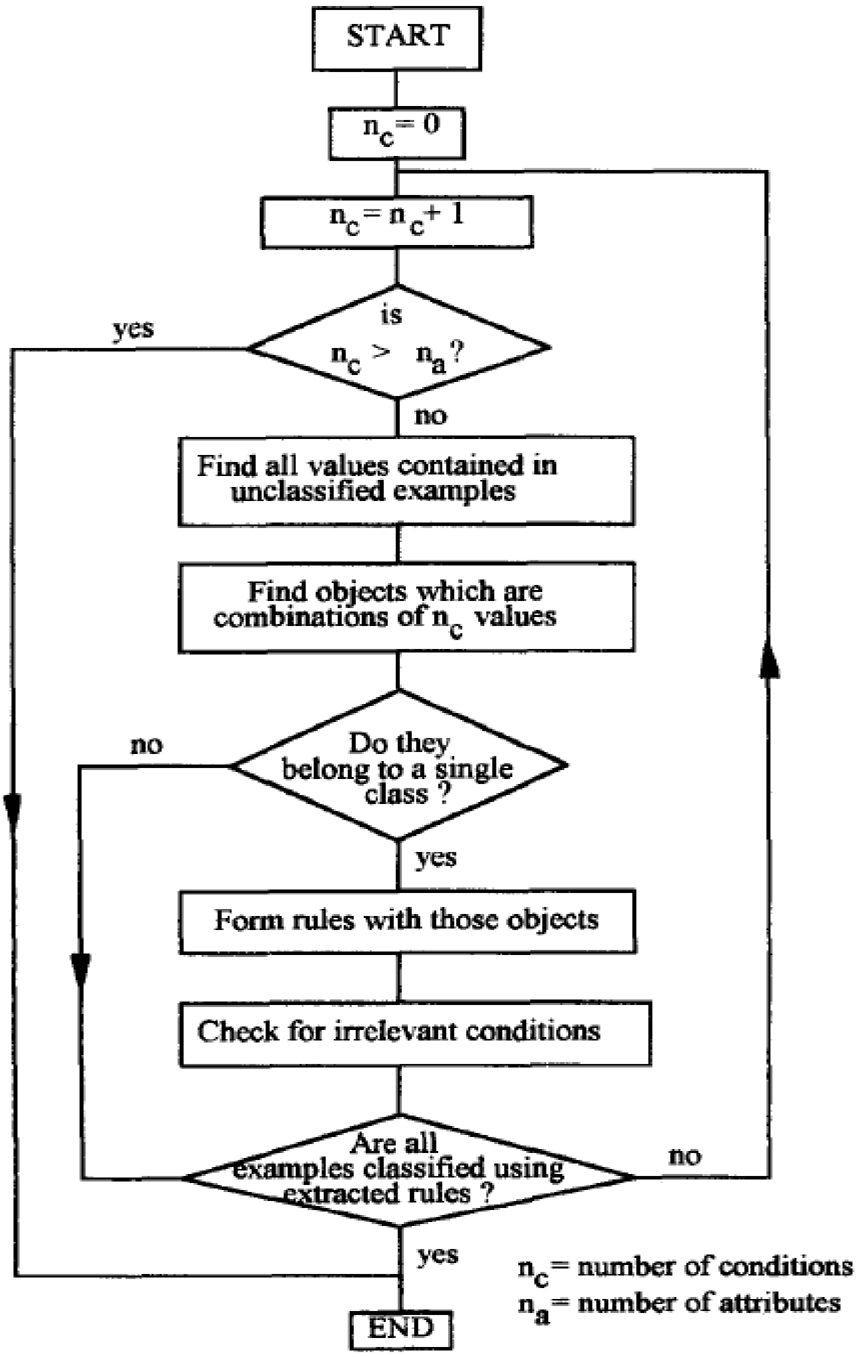

START
$n_c = 0$
$n_c = n_c + 1$
is $n_c > n_a$?
yes
no
Find all values contained in unclassified examples
Find objects which are combinations of n_c values
Do they belong to a single class ?
no
yes
Form rules with those objects
Check for irrelevant conditions
Are all examples classified using extracted rules ?
no
yes
END
n_c = number of conditions
n_a = number of attributes

Figure 3.1: The flow chart for RULES-1 Pham and Aksoy (1995a).

3.2.2 RULES-2

To improve upon RULES-1, Pham and Aksoy (1993) developed RULES-2. The structuring procedure of RULES-2 is same as in RULES-1. The major difference between these two rules is that RULE 2 uses values from unclassified examples for the development of rules and for classification of the examples, instead of using unclassified examples from each iteration. Aksoy (2008) stated that RULES-2 is more efficient than RULES-1, and that its algorithm requires a smaller number of rule selection operations in the induction process. This algorithm controls the upper limit of the rules that are required for the extraction process, as it deals with only one unclassified example in each iteration. Zaki and Meira (2014) stated that RULES-2 allows a minimum limit to be set for the extraction of the rule which is 1.

RULES-2 automatically shuns irrelevant conditions and no additional step is required by this algorithm during the induction process. RULES-2 easily handles incomplete data, which further differentiates it from RULES-1. During the induction process, incomplete values are ignored by this algorithm without any interference in the operation of the algorithm. Aksoy (2008) said that RULES-2 can easily handle characteristics with numerical values by quantising them. This is the main reason for the importance of RULES-2 in engineering applications.

3.2.3 RULES-3

According to Mathkour (2010), RULES-3 is the third version of the RULES family of automatic extraction systems, and it retains all the advantageous features of its predecessors. Aksoy (2008) stated that RULES-3 has two main features that differentiate it from the previous RULES algorithms. The first is that it provides the user with the option to adjust the accuracy of the extracted rules, and the second is that it generates condensed sets with more general rules. Mathkour (2010) stated that RULES-3 specifies the minimum number of conditions required for the rule. By doing so the user can develop a more accurate set of rules and can also reduce the number of searching operations that are required to discover the rule set for data with various characteristics. OUTPUT: Long term memory, Short term memory, and updated frequency distribution of training examples and updated range of values required for numerical characteristics. Figure 3.2 shows the flow chart for RULES-3.

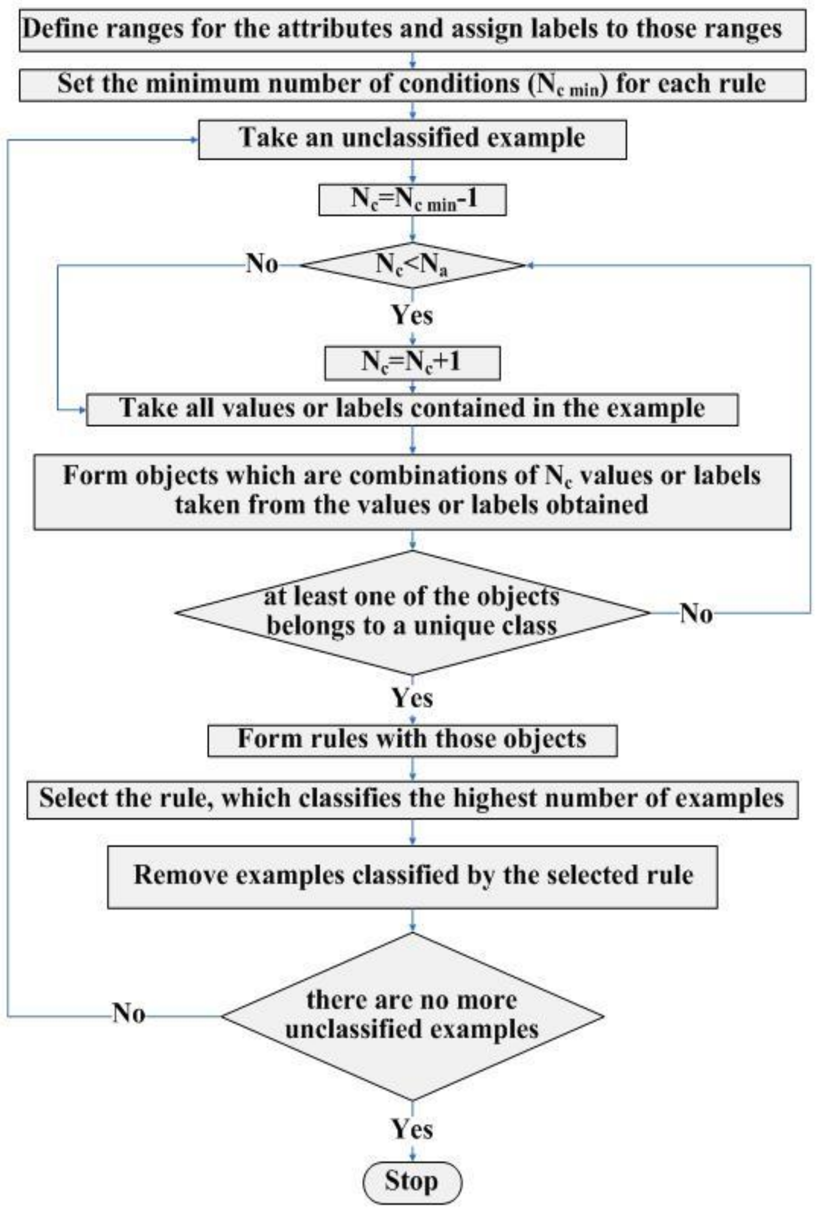

Define ranges for the attributes and assign labels to those ranges
Set the minimum number of conditions ($N_{c\ min}$) for each rule
Take an unclassified example
$N_c=N_{c\ min}-1$
No
$N_c<N_a$
Yes
$N_c=N_c+1$
Take all values or labels contained in the example
Form objects which are combinations of N_c values or labels taken from the values or labels obtained
at least one of the objects belongs to a unique class
No
Yes
Form rules with those objects
Select the rule, which classifies the highest number of examples
Remove examples classified by the selected rule
No
there are no more unclassified examples
Yes
Stop

Figure 3.2: The flow chart of RULES-3 Pham and Aksoy (1995b).

3.2.4 RULES-3 Plus

Pham and Dimov (1997) further improved RULES 3 to create RULES-3 Plus, which includes two more important features. Aksoy (2008) said that RULES-3 Plus employs a more efficient rule searching technique and uses a simple metric for categorisation and selection of candidate rules based on their accuracy.

Rules are formed by selecting those with the maximum "*H measure*", defined in Equation 3.1, which evaluates the information content of the expressions during the rule forming process (Clark and Niblett, 1989).

$$H = \sqrt{(\frac{E^C}{E})}\left[2 - 2\sqrt{(\frac{E_i^C E_i}{E^C E})} - 2\sqrt{(1 - \frac{E_i^C}{E^C})(1 - \frac{E_i}{E})}\right] \quad (3.1)$$

where E^c is the number of examples covered by the rule (the total number of examples classified, correctly or incorrectly), E is the total number of examples, E_i^c is the number of examples covered by the expression and belonging to target class i (the number of examples correctly classified) and E_i is the sum of examples in the training set which is a part of target class i.

The computation of the *H measure* is dependent on the following three parameters:

- The number of examples in the training set;
- The number of examples classified, either correctly or incorrectly, by the rule;
- The number of correctly classified examples covered by the rule.

Pham and Dimov (1997) further explained RULES-3 Plus as the algorithm that includes a set of attributes and values that are formed by using attribute-value pairs used in the example that is under consideration.

3.2.4.1 Rule forming Procedure

To form a rule, RULES-3 Plus performs a general-to-specific beam search for the most general and consistent rule. It starts with the most general rule and gradually specialises it, considering only conditions extractable from the selected seed example. The aim of specialisation is to construct a rule that covers the seed example and as many positive examples as possible while excluding all negative examples. The result is a rule that is as consistent and as general as possible.

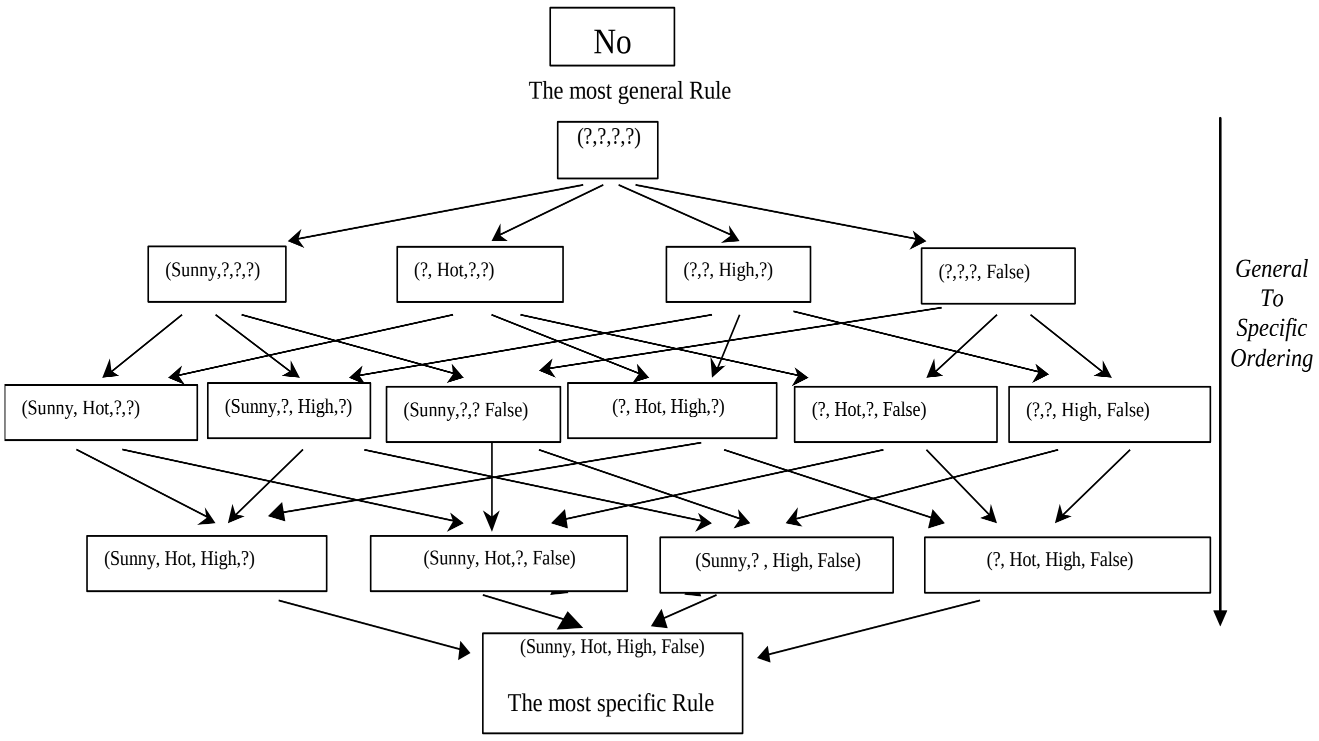


Figure 3.3: Illustration of Beam Search for the first example in Table 2.1

Figure 3.4 presents a summary of the rule forming procedure for RULES-3 Plus. Despite its simplicity, RULES-3 Plus achieves a considerable high level of performance in several engineering applications Aksoy (2005). These include the recognition of design form features in CAD models for computer aided process planning (Pham and Dimov, 1998), the mapping of manufacturing information to design features (Pham and Dimov, 1998) and the classification of defects in automated visual inspection (Jennings, 1996).

3.2.4.2 The specialisation Procedure

The specialisation process in RULES-3 Plus can result in the following three conclusions:

- There are no rules that remain unchanged. All rules in PRSET are specialised further by rerunning the same process.
- Only one rule remains unchanged. The rule is added to the rule set and the search stops.
- More than one rule remains unchanged. The rule with the maximum value for the H measure is added to the rule set and the search stops.

```
Step-1.   Quantise attributes that have numerical values
Step-2.   Select an unclassified example and form array SETAV
Step-3.   Initialise arrays PRSET and T_PRSET (PRSET and T_PRSET will consist of
          $m_{PRSET}$ expressions with null conditions and zero H measures) and
          set $n_{CO} = 0$.
Step-4.   IF $n_{CO} \langle n_a$
          THEN $n_{CO} = n_{co} + 1$ and set $m = 0$;
          ELSE the example itself is taken as a rule and goes to Step 7.
Step-5.   DO
          $m = m + 1$;
          Form an array of expressions (T_EXP). The elements of this array are combinations of
          expression m in PRSET with conditions from SETAV that differ from the conditions
          already included in the expression m (the number of elements in T_ EXP
          is: $n_a - n_{CO}$. Set k = 1;
            DO
                $k = k + 1$;
                Compute the H measure of expression k in T_ EXP;
                IF its H measure is higher than the H measure of any
                expression in T_PRSET
                THEN replace the expression having the lowest H measure with expression k;
          WHILE $k \langle n_a - n_{CO}$;
          Discard the array T_EXP;
          WHILE $m \langle m_{PRSET}$
Step-6.   IF there are consistent expressions in T_PRSET
          THEN choose as a rule the expression that has the highest H measure and discard the
          others; mark the examples covered by this rule as classified;
          go to Step 7;
          ELSE copy T_PRSET into PRSET;
          initialise T_PRSET and go to Step 4
Step-7.   IF there are no more unclassified examples
          THEN STOP;
          ELSE go to Step 2.
```

Figure 3.4: A pseudo-code description of RULES-3 Plus (Pham and Dimov, 1997b).

3.2.5 RULES-4

According to Zaki and Meira (2014) RULES-4 is the first incremental learning algorithm of the family that has the capability to refine and update its knowledge according to new examples. RULE 4 uses the same procedure for forming rules as in RULES-3 Plus. Pham and Dimov (1997) explained the incremental procedure of RULES-4 as:

- INPUT: Long term memory, short term memory, frequency distribution of training examples, and the range of values required for numerical characterisations.
- OUTPUT: Long term memory, short term memory, an updated frequency distribution of training examples, and an updated range of values required for numerical characteristics.

In step one of the process RULES-4 updates the frequency distribution of the training examples. In step two it tests the range of numerical characteristics, and if it is not in range then it updates the range of the characteristics. In step three it tests the classification of rules in the long term memory. In step four it prunes the long term memory by removing the rules, and in the last step it tests the limit of pre-specified examples and replaces the examples accordingly. The pre-specified level discussed in step 4 is known as the noise threshold, and can be defined by the following equation:

$$T = 2 - 2\sqrt{(1-NL)\tfrac{E_i}{E}} - 2\sqrt{NL(1-\frac{E_i}{E})} \tag{3.2}$$

where *NL* represents the pre-specified level of noise. If the accuracy of the measure, *A*, for the rule is less than the threshold, *T*, then it will be automatically removed from the long term memory.

Figure A1 in Appendix A shows the main steps in RULES-4. The extraction of rules for a process planning expert system has been used to illustrate the operation of RULES-4 by Pham and Dimov (1997a).

3.2.6 RULES-5

According to Aksoy (2008), RULES-5 was developed to overcome the deficits of RULES-3 Plus, as this algorithm uses different methods for handling continuous characteristics. This algorithm does not require quantisation for processing the numerical values of the characteristics. Pham and Bigot (2003) explained RULES-5 as the process of selecting and handling continuous characteristics, which is the main procedure that is employed. Pham and Bigot (2003) said that RULES-5 measures the distance between the examples. Supposing E_1 and E_2 are the examples, then it can be elaborated by the following equation:

$$Distance\_E_1 - E_2 = \sqrt{\sum_C \left(\frac{V_{E1}^i - V_{E2}^i}{V_{\max}^i - V_{\min}^i}\right)^2 + \sum_d d\_distance}\,, \qquad (3.3)$$

where $\sum_C$ represents the number of continuous attributes while $\sum_d$ represents the number of discrete characteristics. V_{E1}^i represents the *ith* attributes in example E_1, while V_{E2}^i represents the *ith* attributes of example E_2. $V_{\max}^i$ represents the maximum value of the *ith* continuous attribute, $V_{\min}^{i}$ represents the minimum value of the *ith* continuous attribute and *d_distance* is defined for every discrete attribute by applying the following rule:

If $V_{E1}^{i} = V_{E2}^{i}$ then *d_distance* will be equal to zero, otherwise it will be equal to one.

3.2.6.1 Rule forming Procedure

The initial step in this process is that of selecting of a seed example (*SE*), which is the example that is used to generate a new rule. In RULES-5, the first example in the list of training examples that is not covered by the previously created rules is normally the seed example. The second step adopts a specific search process that creates a consistent and general rule that covers *SE*. The most significant characteristic of this search is that the numerical ranges of the attributes are automatically created during the process of forming rules. That is to say, they are not pre-discretised. The result is a rule, *R*, where all numerical conditions take the form $V_{\min R}^{\ i} \le A^{i} \le V_{\max R}^{\ i}$ (excluding the edges). These conditions can cover large areas in the example space. Therefore, as the third and final step, the algorithm deploys a post-processing method that reduces the coverage of some of the numerical attribute conditions to cover the examples in *T* only. All numerical conditions will take the form $V_{\min R}^{\ i} \le A^{i} \le V_{\max R}^{\ i}$. This prevents the coverage of 'unknown' areas and lowers the possibility of having overlapping rules.

The complete rule forming process of RULES-5 can be summarised as follows:

```
WHILE there is an example in T not covered by any rule in the rule set formed so far, DO
    • Select one of these uncovered examples as a seed example (SE)
    •   Form the all-inclusive rule covering SE: (IF no condition THEN class is class of SE)
        WHILE this rule covers example not belonging to class of SE, DO
    •   Append a condition to it that excludes the closest example to SE that does not belong to class of
        SE.
        END
        • Add this rule to the rule set;
END
```

For more details on RULES-5 Plus, see Figure A2: RULES-5 Plus forming procedure (Bigot 2002).

3.2.6.2 The Specialisation Procedure

As already seen, RULES-5 creates a set of rules whose coverage is limited to the training examples only. Hence, when a set of rules is used as a classification model, there are three possible outcomes:

- Only one rule covers the example. The example belongs to the class of the covering rule.
- More than one rule covers the example. The rule with the highest *H* measure is used to classify the example.
- No rules cover the example. The rule 'closest' to the example in the attribute space is employed to classify it. To find the 'closest' rule, the distance between a rule *R* and an example *E* is defined as follows:

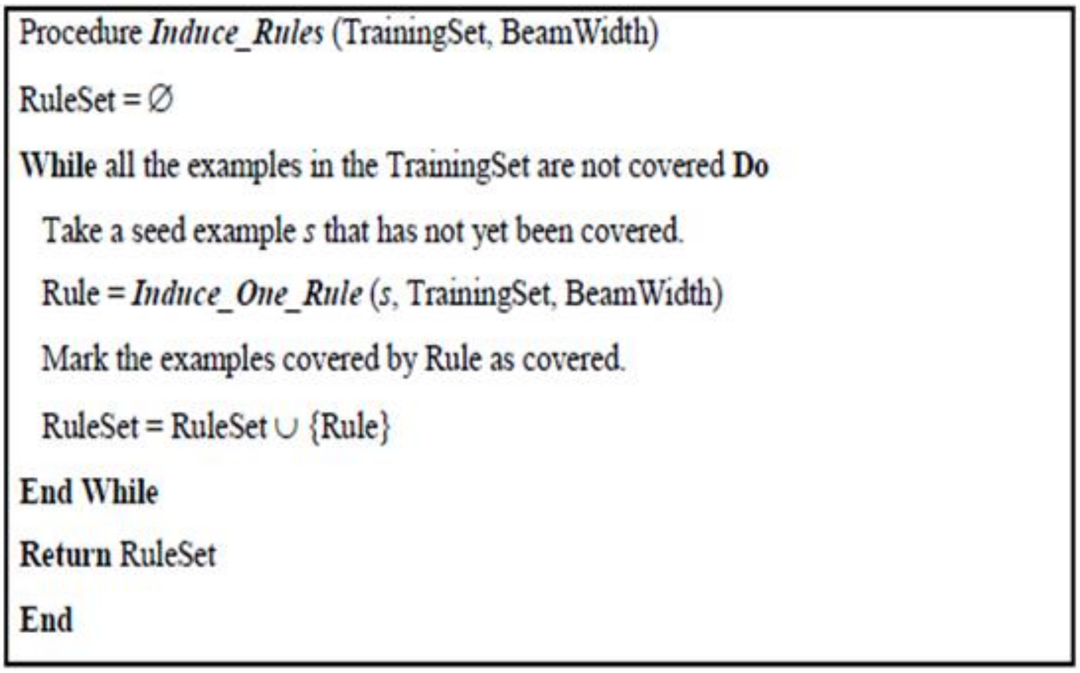
$$\text{Distance}_{R/E} = \sqrt{\sum_c c\_distance + \sum_d d\_distance} \qquad (3.4)$$

An example to illustrate the operation of the RULES-5 algorithm can be found in Pham et al. (2003).

3.2.7 RULES-6

RULES-6 (Pham and Afify, 2005) is an improved version of the RULES-3 Plus algorithm. The innovation in RULES-6 is that it has the ability to handle noise in the data, which is achieved by employing a search method that tolerates inconsistency in the rule specialisation process. This makes the rule sets extracted by RULES-6 both more accurate and substantially simpler than those produced using RULES-3 Plus (Afify, 2004). RULES-6 also employs appropriate search-space pruning rules to avoid useless specialisations and to terminate the search during rule construction. This substantially increases the efficiency of the learning process. Secondly, RULES-6 adopts a very simple criterion for evaluating the quality of rules, and a robust method for handling attributes with continuous values, further improving the performance of the algorithm. Moreover like its predecessors in the RULES family, RULES-6 extracts rules by processing one example at a time (Afify, 2004). The algorithm first selects a seed example; the first example in the training set not covered by previously created rules, and then calls the Induce-One-Rule procedure to extract a rule that covers that example. Following this, all covered examples are marked, the learned rule is added to the rule set, and the process is repeated until all examples in the training set have been covered. The Induce-One-Rule procedure searches for rules by carrying out a pruned general-to-specific search. The search aims to generate rules which cover as many examples as possible from the target class and as few examples as possible from the other classes, while ensuring that the seed example remains covered. As a consequence, simpler rules that are not consistent, but are more accurate for unseen data, can be learned. This contrasts with the rule forming procedure of RULES-3 Plus, which restricts its search to only those rules that are completely consistent with the training data, leading to overfitting if the data is noisy.

A beam search is employed to find the best rule. This is done by using two rule lists named PartialRules and NewPartialRules. PartialRules, which is the same size as the beam width W, stores the W best partial rules during the specialisation process (Afify, 2004). Only the rules in this list are considered.



```
Procedure Induce_Rules (TrainingSet, BeamWidth)
RuleSet = ∅
While all the examples in the TrainingSet are not covered Do
  Take a seed example s that has not yet been covered.
  Rule = Induce_One_Rule (s, TrainingSet, BeamWidth)
  Mark the examples covered by Rule as covered.
  RuleSet = RuleSet ∪ {Rule}
End While
Return RuleSet
End
```

Figure 3.5: A pseudo-code description of RULES-6 (Afify, 2004).

3.2.8 Conclusions about the RULES family

RULES-3 Plus, RULES-5 and RULES-6 are members of the RULES family of simple inductive learning algorithms. They have been used successfully in many engineering applications. RULES-5 employs a more efficient rule forming technique and a more advanced method of dealing with continuous attributes. RULES-3 Plus requires modification in order to be a practical tool for problems involving large data sets. RULES-6 employs a fast and noise-tolerant search method for extracting IF-THEN rules from examples.

An inductive learning algorithm usually demands a set of examples as an input. These examples result in the formulation of a specific rule. Distinct sequences of seed examples that can result in separate and distinct rule sets. Generally, the RULES family algorithms do not follow any guidelines for any of the instances of selecting seed examples. During the systematic series of continuous actions used to select objects in the training set, the algorithm relies entirely on a random method. In theory, a compromise between the accuracy and generality is required, and attained, throughout the duration of the evaluation procedure. This results in a rule that extends over many positive examples while at the same time covering a limited number of negative examples (as few as possible). Generally, a trade-off between the conditions of accuracy and generality is achieved by deploying evaluation methods. Simplicity is a third condition that can be taken into account as it aims to obtain rule sets that are small and easy to comprehend.

Nevertheless, there are a number of research topics in inductive learning that require further investigation. One such topic includes the input training data set, which has the capacity to greatly affect the learning process. This chapter, and the following two, will focus on discovering new ways to pre-process training data prior to inputting it into inductive algorithms.

This study is focused particularly on the RULES family of algorithms, owing to the fact that the research was primarily built and conducted based on this family. Efforts have previously been made to improve RULES-3 Plus, as shown earlier by the RULES-5 and RULES-6 algorithms. The ideas presented in these two algorithms can also be combined to form a significant improvement on the RULES-3 Plus algorithm. RULES-3 Plus does not always select the best rule during the specialisation process, because it relies more on the consistency than the heuristic measure. RULE-5 and RULES-6 deal with this case by steering the search space and in some cases the consistency measure is not taken into account. The strained compromise between the consistency and heuristic measure leads to the formulated rules being either more accurate or more general.

Therefore, development of an improved technique for ranking the data is required to overcome this before they are introduced to RULES family inductive learning algorithm. A suitable buffer strategy could also be used to seed candidate rules and hence improve the quality of the formed rule sets.

3.3 Preprocessing of data

Although numerous methods of data pre-processing have been developed, it remains an active area of research due to the huge amount of inconsistent or dirty data and the complexity of the problem.

There are a number of existing data pre-processing techniques. Before using an inductive learning algorithm, pre-processing of data is often required, for instance to select relevant attributes, correct errors or deal with missing information. Here, a new pre-processing technique is introduced in an attempt to create a sequence of seed examples that leads to the best achievable rule set (Pham and Dimov, 1997).

3.3.1 Definition of the algorithm

The RULES family of algorithms deploys a search strategy based on the selection of seed examples. Each time a seed example is chosen it can result in the formation of a specific series of seed examples, which can produce different rule sets. In the current RULES family versions, there are no guidelines on the selection of seed examples. The first example to be selected is one that remains uncovered and is located in the training set In this manner, the rule set produced becomes dependent on the storage of the examples. In the suggested algorithm, the selection of seed examples is carried by employing two parameters known as *knowledge measurement* and *ranking*.

Knowledge measurement is inversely proportional to probability. For example, if the probability for output x is $p(x)$, then the knowledge measurement, $I(x)$, for the same output is denoted with Equation 3.5 (Pham, Bigot and Dimov , 2006).

$$I(x) = f\left(\frac{1}{p(x)}\right) \tag{3.5}$$

Hence, the entropy of any event x with number of variables, *n*, can be calculated by Equations 3.6 to 3.8.

$$E(x) \equiv \sum_{i=1}^{n} p_i \log\left(\frac{1}{P_i}\right) \tag{3.6}$$

$$E(x) \equiv \sum_{i=1}^{n} p_i I_i \tag{3.7}$$

$$E(x) \equiv \sum_{i=1}^{n} p_i \log(p_i) \qquad (3.8)$$

In the above equations, any base can be used for the logarithmic functions. However, base 2 has been commonly used in the literature. In this case, bits (binary digits) may be used as the unit of knowledge measurement and entropy (Blake and Merz, 1998).

The multiplication of the entropy of an attribute with the number of items the attribute gives the reformulated entropy, as shown in Equation 3.9:

$$REntropy(A_i) \equiv E(A_i)xn_i \qquad (3.9)$$

Here A_i denotes the *ith* attribute, and n_i denotes the number of items in the *ith* attribute.

Entropy is computed as follows

$$Entropy(S) \equiv \sum_{i=1}^{c} - p_i \log_2 p_i \qquad (3.10)$$

where $Entropy(S)$ is the entropy of S relative to c class classification, S is a set of objects containing representatives of more than one class, p_i is the proportion of S belonging to class i , and Log_2 is the base 2 logarithm. A base 2 logarithm has been used in this equation because entropy is a measure of the expected encoding length measured in bits. The target attribute can take on *c* possible values so the entropy can be as large as Log_2 c.

3.3.2 The algorithm

The proposed method aims at assessing the levels of precedence for each example in order to obtain ranks. It relies on the calculation of the entropy (Equation 3.10) of each attribute-value pair, which can then be used to assess the sum of the entropies $I_{SE,i}$ of each attribute-value pair in each example by using Equation 3.11:

$$I_{SE,i} = \sum_{j=1}^{m} (E_j) \qquad (3.11)$$

where *m* is the number of attributes in the *ith* example and (E_j) is the entropy for the *jth* attribute-value pair in the *ith* example. Thus, each example is given a level of precedence. The most significant seed examples (lowest $I_{SE,i}$ measure) are considered at the beginning of the rule forming process. The less significant seed examples from the example set are either considered at the final stage of the rule forming process or neglected completely. In this way, it is expected that more general rules can be generated. In a broad sense entropy is defined as the disorder of a given system.

The data ordering method proposed in this work is shown in Figure 3.6. During the pre-processing stage the entropy is first calculated (Equation 3.10). The output is the entropy value for all attributes. Secondly, the sum of entropies for each example (I_{SE}) is calculated using Equation 3.11. Thirdly a ranking process orders the data by arranging the sequence of the seed example to select the most representative example. At the processing stage, an ordered data set is obtained and used for the training process in one of the RULES family of algorithms. A Rule Set is then created.

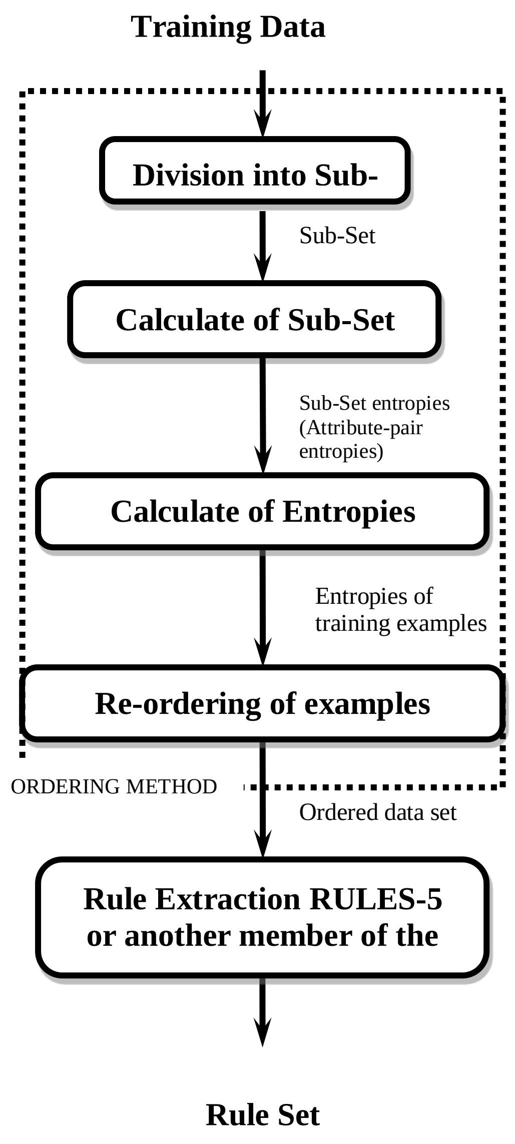


Figure 3.6: Data Ordering Method (DOM).

3.4. Ranking.

Using the Weather Problem in Table 2.1, the attribute-value pair is calculated for each example.

For the first example:-

For (*sunny*), $E_{Outlook,Sunny}(2{,}3) = 0.972$ bits

For (*overcast*), $E_{Outlook,Overcast}(4{,}0) = 0$ bit

For (*rain*), $E_{Outlook,Rainy}(3{,}2) = 0.972$ bit

The other examples are shown in the Table 2.1

Table 3.1: Entropies of attributes values.

Attribute	Value	Entropy (bit)
Outlook	Sunny	0.972
	Rainy	0.972
	Overcast	0.00
Humidity	High	0.986
	Normal	0.811
Windy	False	0.919
	True	0.972
Temp	Hot	1.00
	Cool	0.920
	Mild	0.811

At this stage examples are ranked and the new data in Table 2.1 are used for each example to calculate I_{SE} by using Equation (3.11).

For instance:

$$I_{se,1} = E_{Outlook,Sunny} + E_{Humidity,High} + E_{Temperature,Hot} + E_{Wind,False} = 3.877$$

And similarly:

$$I_{se,2} = 3.877$$

The value of I_{SE} is calculated in a similar manner for all examples and sorted from lower to higher.

The rules obtained by the RULES-5 Plus algorithm with data ordering method were produced using entropy values as shown in Figure 3.7.

IF Outlook = sunny AND Humidity = high $\Rightarrow$ *Play = No*

IF Outlook = overcast $\Rightarrow$ *Play = Yes*

IF Outlook = rainy AND Windy = true $\Rightarrow$ *Play = No*

IF Outlook = rainy AND Windy = false $\Rightarrow$ *Play = Yes*

IF Outlook = sunny AND Humidity = normal $\Rightarrow$ *Play = Yes*

Figure 3.7: Generated rules for the set of examples given in Table 3.2

In particular, the examples with lower $I_{SE,i}$ were of higher precedence and were presented at the beginning of the inductive learning process. The second rule obtained, as shown in Figure 3.7, covers more examples than the other rules (4 examples out of 14 with entropy value 0.00 bit). However, the examples with high sum of entropies were of lower precedence, and as a consequence were presented at the end of the learning process. None of the rules include the attribute Temperature with value Hot, which is associated with the highest entropy, of value 1 bit.

3.4.1 Illustrative example of DOM

Figure 3.7 (a) shows 2D data belonging to 7 separable classes. In the training stage of an inductive learning algorithm it is desirable to generate classification rules that are as accurate and compact as possible. For this example the latest version of the algorithms belonging to the RULES family was used. The latest version is RULES-5 Plus, which presents several improvements over the older algorithms in the RULES family.

When the data was presented in a random order to the RULES-5 Plus algorithm, 10 rules were created as shown in Figure 3.7 (b). The RULES family algorithms use the seed example *(SE)* method, where conditions are formed based only on the attribute-value pairs of the selected seed example (data). It can be seen that even in this simple example the random order has affected the RULES-5 Plus algorithm (RULES-3 Plus for this same example generated 17 rules) due to the *SE* method. Three more rules were created, R2, R5, and R8, which are irrelevant or redundant in this case since these rules are completely contained by rules R7, R10, and R9 respectively.

Using the DOM as its initial stage, RULES-5 Plus generated 7 rules as shown in Figure 3.7 (c). The rules obtained in this case are what one would expect in a 7-class problem where each class is separable from the others. Another point of note is the order in which the rules where formed. In Figure 3.7 (c) it is clear that the more separate and simple rules are formed first, that is R1, R2, and R3. On the other hand, these rules in Figure 3.7 (b) correspond to R3, R4, and R9 respectively.

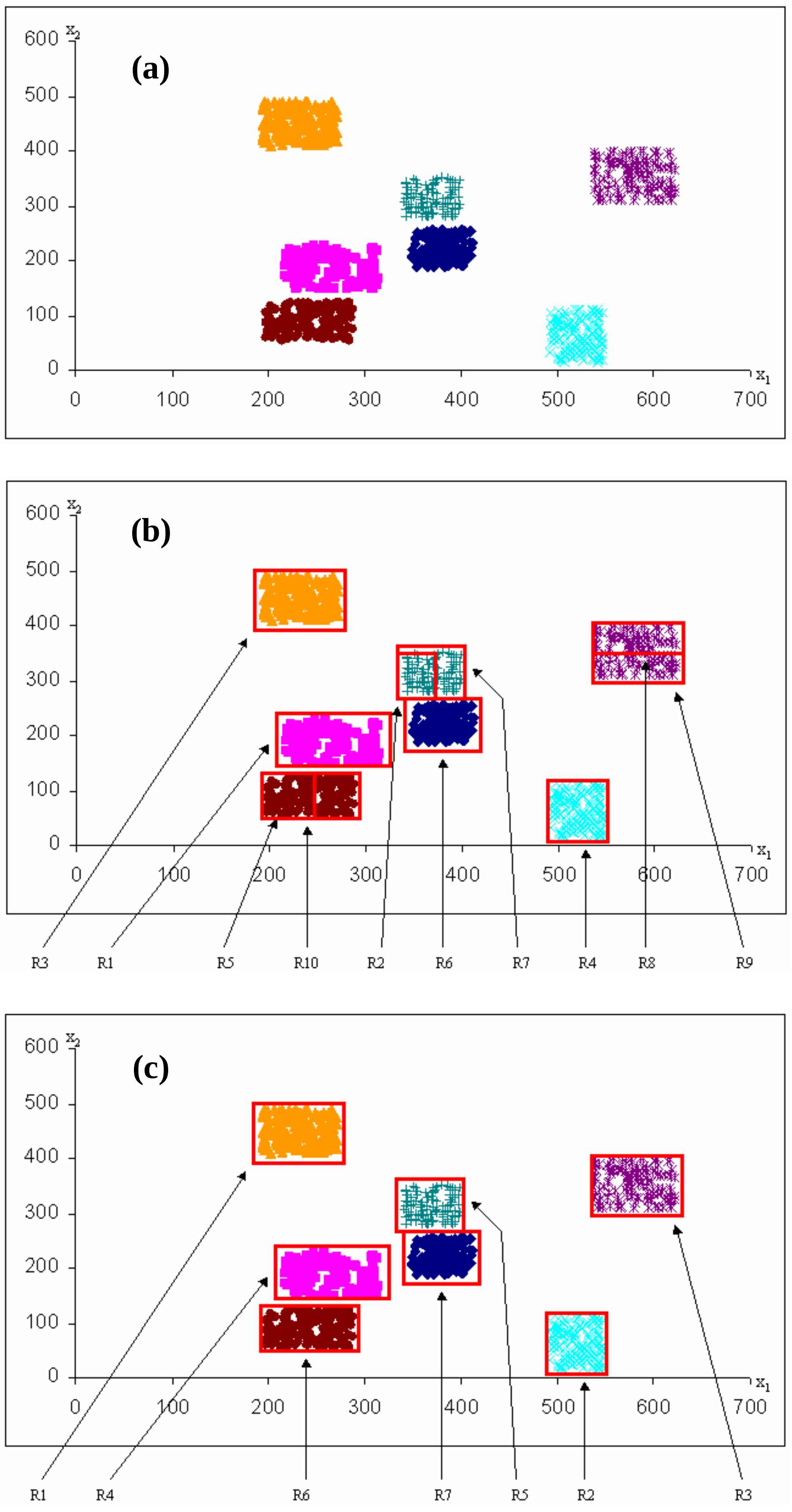


Figure 3.7: Example of random presentation of the data in the training process of the RULES-5 Plus algorithm and ordered presentation of the data using the DOM. (a) data belonging to 7 separate classes; (b) rules formed in the training process with random presentation of the data; (c) rules formed in the training process with ordered presentation of the data.

3.5 Experiments

The Method proposed in this work was tested using 21 benchmark data sets (alarm problem, car, chess, haberman, hayes, glass, heart disease, heartr, hepatitis, mass, monk1, monk2, monk3, mushroom, pima, post-operative, shuttle, statlog (A.C.A.), statlog (H), tic-tac-toe, voting, weather) from the UCI Machine Learning Repository (Blake and Merz 1998). A list with this data set describing the number of attributes, classes are shown in table 3.2.

To measure how sensible the rule generation process (training stage) is to different orders of pattern presentations, 3 random ordered presentations of the training data sets were used in the Rules family algorithms and compared to the performance obtained by the method. The Rules family algorithms that were used in this experiment were RULES-3 Plus, RULES-5 and RULES-5 Plus (without pruning).
All tests were conducted on an Intel Pentium 2.0 GHz Dual-Core computer with 2 GB of RAM and Windows XP operating system.

Table 3.2: Description of the benchmark data sets.

no	data set	attributes	classes	#examples
1	AlarmProblem	03	02	8
2	Car	06	02	1728
3	Chess	06	02	28056
4	Haberman	03	02	306
5	Hayes	05	02	160
6	Heart Disease	75	02	303
7	Heart	13	02	270
8	Hepatitis	19	02	155
9	Mass	06	02	961
10	Monk1	07	02	432
11	Monk2	07	02	432
12	Monk3	07	02	432
13	Mushroom	22	02	8124
14	pima	08	02	768
15	post-operative	08	02	90
16	Shuttle	06	02	15
17	Statlog(A.C.A.)	14	02	690
18	Statlog (H.)	13	02	270
19	Tic-Tac-Toe	09	02	958
20	Voting	16	02	435
21	Weather	04	02	14

3.6 Results and Discussions

To assess the effect of different seed example orders on the created rule sets, five different orderings were tested. These were: (i) RULES-3 Plus (Pham and Dimov, 1997), (ii) RULES-5 Plus (Pham, Bigot and Dimov, 2004) and (iii) RULES-5 (Pham, Bigot and Dimov, 2003). The first ordering under test was the original default one given in the data set repository. The second ordering tested was the one proposed in the previous section. Three more random orders were created to appropriately evaluate the quality of this new ordering method. The size of the models has been used as the only performance indicator in this study. Future studies will include a validation of the generated models using a distinct testing set.

Table 3.3 displays the highest number of deviations in the number of rules generated by each algorithm when the 5 different ordering systems are applied to each data set.

The results show that the ordering of seed examples has a measurable effect on the number of rules created by the three algorithms, with a variation in the number of rules of up to 26.32% (13 rules), 39.92% (24 rules) and 57.25% (75 rules) for RULES-5 Plus, RULES-5 and RULES-3 Plus respectively.

Table 3.3: Deviations in number of rules created.

	Maximum difference in number of rules created			**Maximum deviation (%) in number of rules created**		
Data Set	**RULES5Plus**	**RULES5**	**RULES3Plus**	**RULES5Plus**	**RULES5**	**RULES3Plus**
AlarmProblem	0	0	0	0.00	0.00	0.00
car	0	0	4	0.00	0.00	0.02
chess	6	24	75	14.29	36.92	57.25
haberman	6	9	0	7.89	10.34	0.00
Hayes	4	5	2	13.79	14.29	5.26
Heart Disease	5	15	13	9.09	16.30	9.42
Heart	4	6	4	21.05	27.27	17.39
Hepatitis	2	7	2	12.50	21.88	5.88
Mass	13	13	3	8.07	6.91	2.38
Monk1	0	0	3	0.00	0.00	7.14
Monk2	0	0	0	0.00	0.00	0.00
Monk3	0	0	0	0.00	0.00	0.00
Mushroom	5	4	4	26.32	18.18	18.18
pima	10	22	10	11.24	13.58	3.40
post-operative	6	9	2	16.67	22.50	5.13
Shuttle	0	0	0	0.00	0.00	0.00
Statlog (A.C.A.)	6	12	17	9.84	12.24	9.19
Statlog (H.)	2	7	9	7.41	15.22	12.33
Tic-Tac-Toe	2	4	11	8.33	13.79	29.73
Voting	2	4	6	6.67	11.11	16.22
weather	1	1	1	16.67	16.67	16.67
Maximum	**13.00**	**24.00**	**38.00**	**26.32**	**36.92**	**57.25**
Average	**2.31**	**4.44**	**5.19**	**5.93**	**8.04**	**6.74**

Table 3.4: The results obtained using the 3 algorithms.

		Average deviations(%) from average number of rules, obtained with the 3 algorithms				
Data Set	**Average No Rules**	**Original Order**	**DOM**	**Random Order 1**	**Random Order 2**	**Random Order 3**
Alarm Problem	4.00	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
car	185.53	0.25	**-0.47**	*0.25*	0.25	-0.29
chess	56.87	*23.68*	8.44	**-14.42**	-13.25	-4.45
haberman	65.53	0.20	*2.75*	0.71	0.20	**-3.87**
Hayes	31.80	**-3.56**	*3.77*	-1.47	-1.47	2.73
Heart Disease	89.27	0.07	-1.42	-1.42	**-1.79**	*4.56*
Heart	19.47	*6.16*	**-14.38**	2.74	2.74	2.74
Hepatitis	25.13	*7.43*	**-1.86**	**-1.86**	**-1.86**	**-1.86**
Mass	153.73	*2.34*	1.69	-1.56	-0.69	**-1.78**
Monk1	22.87	0.58	**-2.33**	*2.04*	*2.04*	**-2.33**
Monk2	302.67	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
Monk3	218.33	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
Mushroom	19.47	-2.40	**-14.38**	2.74	6.16	*7.88*
pima	173.87	*3.91*	-1.07	-1.27	**-2.99**	1.42
post-operative	34.87	*8.03*	**-3.44**	-1.53	-1.53	-1.53
Shuttle	10.00	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
Statlog (A.C.A.)	108.00	*1.54*	**-1.23**	-0.93	-0.31	0.93
Statlog (H.)	45.07	**-3.85**	0.59	-0.15	*6.51*	-3.11
Tic-Tac-Toe	27.47	*3.16*	*3.16*	**-10.19**	0.73	*3.16*
Voting	32.53	-0.61	*4.51*	*4.51*	**-4.71**	-3.69
weather	5.60	**-10.71**	**-10.71**	*7.14*	*7.14*	*7.14*
Average Deviation		**1.13%**	**-0.82%**	**-0.46%**	**-0.09%**	**0.24%**
% of Best Result		**21.88%**	**34.38%**	**21.88%**	**25.00%**	**25.00%**
% of Worst Result		**25.00%**	**12.50%**	**12.50%**	**9.38%**	**12.50%**

Table 3.4 shows the average deviation in the number of rules created from the total average obtained when applying the various algorithms.

The results obtained show that on average, there is no significant difference between the three random orders. The original order appears to give slightly worse results (an average increase of 1.13 % in the average number of rules). The average deviations among the data sets reveal that the new ordering method (DOM) gives the best results (lowest number of rules) for 34.38 % of the 21 data sets. This can be compared with 21.88 % and 23.61 % for the original order and the 3 random orders respectively). This is highlighted in bold in Table 3.4.

It should be noted that a large part of the processing time used for the proposed data ordering method is spent on assessing entropy values for each attribute-value pair used by Equation 3.10 in each seed example. This information could be passed on to the covering algorithms, as they use this information in the rule forming process to evaluate each created rule. Thus the computational effort used by this ordering method more efficient. As mentioned previously, the aim of rule pruning is to reduce the number of generated rules without reducing the test set accuracy. The results given in the previous section show that addressing the algorithm and the proposed data ordering method produces a much smaller set of rules. Moreover, the processing time for the majority of data sets could be improved by combining the two algorithms.

3.7 Summary

In this chapter, the RULES family was fully reviewed and a new data ordering method was proposed (DOM) for the inductive learning algorithm of the RULES family. The purpose of this new technique is to reduce the number of rules generated by random and inefficient presentation order of the training patterns. The method consists of 3 main steps: firstly, the entropy is calculated for each of the attribute-value pairs; secondly, the seed example entropy values are calculated by adding all the entropy values to each example; and finally, examples are reordered according to their entropy values.

The training is carried out on 21 data sets and the results show that the performance of the new method, in most cases, was better than random presentation order and the original algorithm.

Chapter 4: Improved data ordering method with new management strategy for rule induction systems

4.1 Preliminaries

Automated data mining or learning disclosure systems can be used to extract principles from databases. Recently the use of data mining has gained significant interest. Scientists have created and connected various machine learning strategies to extract information from large databases and to learn rules for master frameworks. The areas of data mining and machine learning converge, as they both have a similar initial functionality of first removing obscure information from databases.

Sorting is one of the most important problems in computer science. It is a well-studied area and there are many good algorithms that can be used for sorting. Most sorting calculations work by comparing the information that is to be sorted. Large pieces of information or large amounts of data are sorted by taking into account just a sample of the information. The sample of information used to focus the sorting request is known as the key. Sorting calculations are normally judged by their effectiveness. In sorting applications, effectiveness refers to the algorithmic productivity. The extent of the information developed is vast, and for the most part reflects the quantity of components to sort. The majority of the calculations being used have an algorithmic effectiveness of either $O(n^2)$ or $O(n \times log(n))$. While several types of calculation can have the same productivity, they don't necessarily perform at the same speed on the same amount of data. To begin with, calculations must be judged in light of their productivity for a normal case, the best case, and the most pessimistic scenario. Some calculations (for example 'quick sort') perform particularly well for some inputs, yet provide bad results for others. These algorithms offer efficiency and simplicity while bearing in mind minimisation of memory use and other factors. There has been a growing interest of late in improving sorting algorithms to improve performance by enhancing data locality (LaMarca and Ladner, 1997; Jim'enez-Gonz'alez, et al., 2003; Wang and Mendel, 1991).

As noted in chapter 3, feature selection refers to the selection of a useful feature subset for a learning task by reordering a data set. Many data mining algorithms, such as inductive learning, clustering, and association rule discovery, can benefit from feature selection techniques. However, this chapter focuses on selecting features for learning classifiers.

This chapter introduces a new data sorting method, the method developed for inductive learning (DSM). DSM orders the data set by using the entropy value, which is not used in the RULES family algorithms, to give a top priority to the Examples and Attributes with higher importance. This results in generate more compact and more accurate rule sets.

4.2 Performance Improving Techniques

Performance improving techniques help to pinpoint execution issues in a framework. This includes recognising data bottlenecks and resolving them. It is prescribed that progressions should be made to a framework when it has been confirmed that there is a bottleneck. Execution change, by its definition, is iterative. Thus, evacuating the first bottleneck may not quickly introduce execution change, on the grounds that another bottleneck may be uncovered. Likewise, at times, if serialisation leads to a more inefficient system, then performance could decline. With the use of improvement techniques and optimisation methods, applications can be repaired and made adaptable. Execution issues by and large result from either an absence of throughput, inadmissible client/system reaction time, or both.

The issue may be restricted between application modules, or it may be for the whole framework.

These basic achievement elements are better characterised by observing genuine business objectives rather than framework measurements. The main aim is to dispense with any bottlenecks that corrupt execution. These bottlenecks could be brought about by wasteful utilisation of resources. Since every single imparted asset is constrained, the objective of improvement methodology is to augment the business operations with proficient utilisation of resources. This methodology is iterative and it is unavoidable that a few examinations will be conducted that have little effect on the execution of the framework. It requires analysis and experience to build up the essential abilities to precisely pinpoint discriminating constraints and bottlenecks.

In order to enhance the performance techniques of the algorithm or application it is important to gain a full understanding of working framework, database, and application measurements when the execution is both good and bad. This system recognises the greatest bottleneck and uses a target method to deal with execution change. The emphasis is on making huge execution enhancements by expanding application effectiveness and wiping out asset deficiencies and bottlenecks. In this procedure, it is foreseen that negligible execution additions are produced using resource tuning, and extensive increases are produced by secluding application inefficiencies.

The steps associated with this describe how an execution designer may search for system constraints without utilising programmed analytic highlights. These steps are proposed as a rule for the manual methodology. This examination accepts measurements for both the working framework and the database that has been assembled.

If the system efficiency is not achieved despite improving and introducing new rules then the application is most likely not coded or outlined ideally, and it will not be satisfactory in certain circumstances.

In the scenario when the CPU usage is more than 40 %, then an in depth analysis is conducted on the working framework for system exchanges and the paging, swapping, or methodology is adapted.

The data sorting methodology was evaluated and assessed based upon:

- Performance efficiency
- Implementation methods
- Complexity of the code and application

As discussed in previous chapters, it was suggested that the learning mechanism was proposed through techniques such as decision trees and production rules. Several conditions play an important role in any learning algorithm:

- It must be defined with respect to the same set of relevant attributes.
- It must be disjoint in example space.
- It must have uniform distributions (Kibler and Aha, 1987).

Generally, the availability of good sources of samples has a significant value in accomplishing an execution that satisfies all the requests and prerequisites of a machine learning system.

Amsterdam (1988) showed that reordering the training samples provides a better efficiency than being fed with an abundance of examples from a teacher.

Certain calculations, for example AQ and the RULES family, have all the samples promptly available from the earliest starting point of adapting. It is highly valuable to force a satisfactory request on the preparation cases. This is considerably more noteworthy when an incremental learning calculation is used.

It is essential to obtain adequate space information in order to have the capacity to construct a helpful learning base for an expert system. Due to an increase in the volume of information, this methodology has resulted in a bottleneck. Programmed information securing procedures have been created to handle this issue. Inductive learning, or the extraction of information in the form of IF-THEN statements or decision trees, is a programmed procedure for secure learning. An inductive learning program typically requires information to be an arrangement of illustrations. Every illustration is described by the estimations of various sets of ordered data, and the class to which it belongs. One way to deal with inductive learning is through a procedure whereby data orders are chosen according to a method which separates the first case set into subsets. The inductive learning system then manufactures a decision tree that effectively groups the given illustration set. The tree accumulates the learning from the particular samples in the set. This can be used to handle circumstances that are not secured by the sample set. Additional information is established from the base tress. Each branch originating from a root is used to characterise a subset of cases that are incoherent from other subsets. The same method is applied to all the subsets until they all have zero entropy. In another methodology, the inductive learning system determines gatherings of traits imparted by illustrations in given classes. It structures rules, with the IF part as conjunctions of those traits.

In summary, example illustration based on entropy weights is significant for yielding a satisfactory performance. The next section will briefly describe the effect of reordering the examples after each rule is generated.

4.3 The entropy of attributes

A quantitative measure of information or data based upon the probability can be described as entropy. To mathematically define entropy, one must assume that the system state *S* has a probability distribution *P(S)* which determines whether the system is in state *S*. Then the variety which is assumed to be *V* can be used to define entropy *E,* as shown in Equation 4.1:

$$E(P) \equiv -\sum_{s \in S} P(s).\log\ P(s) \tag{4.1}$$

If the probability of a certain state occurring is more than that presumed for the other states, then the entropy *E* can be said to reach its maximum value. *E* communicates the instability of the framework's state. *E* can be equivalent to 0 if the likelihood of a certain state is 1 and the likelihood of every other state is 0. Once all things are considered, there is maximal assurance (complete data) regarding the state that the framework is in. Limitation is characterised as that which diminishes vulnerability, that is, the distinction of a position in between maximal and genuine instability. This distinction can likewise be deciphered in an alternate manner, as data, and verifiably *E* was presented by Shannon as a measure of the limit for data transmission along a correspondence channel. In reality, in the event that some data is collected about the condition of the framework, then this will decrease the vulnerability of the framework's state, by diminishing the likelihood of various states. The data collected from a perception is equivalent to the extent to which instability is decreased:

$$Information = E(before) - E(after) \tag{4.2}$$

Information theory introduces the concept of entropy. Information is directly proportional to entropy, meaning that if there is more information content then entropy will increase. Entropy is used to rank attributes in terms of the reduction in the uncertainty of the class label. The class entropy is a weighted average over all the possibilities. The entropy measures and calculates the information gain, reflecting the quality of an attribute as the branching attribute. The knowledge representation system can be defined as:

$$J = (U, C \cup D). \tag{4.3}$$

This is expressed using a data set with a discrete-valued condition and discrete-valued decision attributes, where:

Set of data samples $U = (u_1, u_2, \dots \dots, u_s)$,

Set of condition attributes $C = (c_1, c_2, \dots \dots, c_n)$, and

Single element set $D = \{d\}$.

D can take on m distinct values, that is $d_{i(\text{for } i=1, \dots, m)}$. If s_i is the sample numbers of U in class d_i, then the expected entropy (I) that can classify a given sample is given by:

$$I(_{S_1,\dots,S_m}) = -\sum_{i=1}^{m} p_i \log_2 p_i \,, \tag{4.4}$$

where p_i is the probability that an arbitrary sample belongs to class s_i, and m is the sample number. The entropy of attribute A, takes values of $\{a_1, a_2, \dots, a_v\}$, with v distinct values. Attribute A can be used to partition U into v subsets from the class s as $\{S_1, S_2, \dots, S_v\}$ for $S_{i(j=1,\dots,v)}$. These subsets contain the samples in U that have value a_j of A. S_{ij} is the number of samples in class d_i. The S_j subset will yield the entropy of attribute A as:

$$E(A) = \sum_{j=1}^{v} \frac{S_{1j} + \dots + S_{mj}}{S} I(_{S_{1j},\dots,S_{mj}}) \tag{4.5}$$

The term $\frac{S_{1j}+\cdots+S_{mj}}{S}$, which acts as the weight of the jth subset, is the number of samples in the subset divided by the total number of samples. When the entropy value is small, the purity value of the subset partitions is greater. The attribute with the largest entropy is normally selected as the attribute that branches off.

4.4 RULES-F, a fuzzy inductive learning algorithm

This section presents a technique that allows a covering algorithm to generate fuzzy rules. This technique is especially adapted to algorithms based on the structure of RULES-5. RULES-5 iteratively applies a specific rule-forming process in order to create a complete rule set covering all examples. Three particular steps of this process are of interest to the development of RULES-F (Pham et al., 2006).

Generation of fuzzy rules is based upon representing a decision tree in form of multiple dimensions. Every cell in the network is known as a fuzzy subspace and represents a rule when connected with a specific result. Various studies have been performed to improve this methodology. Fuzzy logic models have been heavily deployed in automatic knowledge acquisition to build expert systems. Fuzzy logic models can handle vagueness and uncertainty and they possess similarities to some aspects of human reasoning, by dealing with numerical outputs without the need for sophisticated mathematics (Pham, et al., 2006). Using fuzzy logic, one can generate fuzzy rules based on a decision table which takes the form of a multidimensional matrix. This matrix has one dimension per attribute with each cell representing a fuzzy subspace. When the total number of attributes increases, this method of generating fuzzy rules becomes impractical. The process can be automated by using a list of training examples, T in place of the expert knowledge. To do this, a well-known method by

Wang and Mendel (1991) is employed to predefine fuzzy membership functions and to split the increased attribute space into fuzzy regions A fuzzy rule is thereby generated for each example, and the rule is stored in a decision table. As many more rules are able to be generated and stored due to growing memory, it becomes difficult to select the best rule. Frequencies in each fuzzy domain are therefore defined to identify the true rule (Pham, et al., 2006). Many more methods have been proposed to handle the generation of fuzzy rules, but as Hong and Lee (1996) noted, the equal and fixed partitions of the fuzzy membership function that is created makes it difficult to automate fuzzy rule generation. Adapting inductive learning techniques for numerical attribute discretisation have been proposed by several writers (Pham et al., 2006) to provide a solution to the problem of automating membership function design.

From inductive learning techniques emerges the field of fuzzy rule induction, where each fuzzy rule is created from a set of training examples, *T*. In the training set of examples, each example, *E*, is described by the output value, V^{out} of *E*, and a vector of *m* attributes, *A*.

Thus,

$$E = (A^1 = V_E^1, \dots, A^i = V_E^i, \dots, A^m = V_E^m, Output = V_E^{out}), \qquad (4.6)$$

If *R* is a fuzzy rule, then it has a number of fuzzy conditions (*Cond* of *R*) on each of the m input attributes, and a fuzzy output set (F^{out} of *R*). This can be expressed as: $Cond_R^1 \wedge \dots \wedge Cond_R^i \wedge \dots \wedge Cond_R^m \Longrightarrow F_R^{out}$. For RULES-F, the triangular form is used to facilitate computation instead of using different shapes used for the fuzzy sets F_R^i of the *i*th attribute. This fuzzy set for numerical values can be defined as *Tr(a,b,c)*, where *a* and *c* are the base corners of the triangle as shown in Figure 4.1, and b is the apex of the same triangle. Once the output can be predicated using a fuzzy rule, we can directly compare this method to inductive learning. For every fuzzy rule, R, an example, E, will have a particular 'degree of match', $(\mu\_rule_R(E))$, so an evaluation can be conducted to find out how likely a rule is to predict the output value of an example.

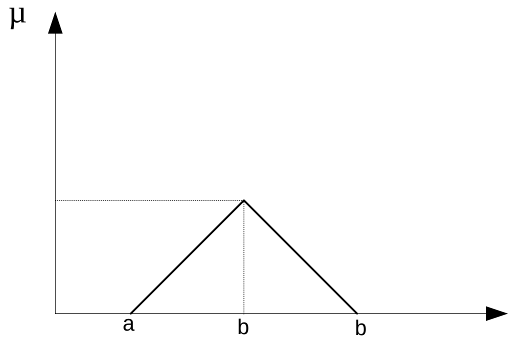


Figure 4.4: Triangular fuzzy set (Pham, et al., 2006).

In RULES-F, the method adopted to evaluate the 'degree of match' of an example to each rule is to use all membership degrees. Therefore the 'degree of match' can be calculated as:

$$\left(\mu\_rule_R(E)\right) = \prod_{i=1}^{m} \left(\mu_{F_R^i}\left(V_E^i\right)\right), \qquad (4.7)$$

where $\mu_{F_R^i}\left(V_E^i\right)$ is the membership degree of each attribute, V_E^i. This can be used to determine the class of a new example after a complete model (a set of rules) has been created from the set of training examples (T). Concentrating on the numerical output, most applications require defuzzification which is a process of converting to a single numerical value. For RULES-F a weighted-average method is adopted to achieve a defuzzified output:

$$output = \frac{\sum_{R=1}^{r}[\mu\_rule_R(E).C_R^{out}]}{\sum_{R=1}^{r}\mu\_rule_R(E)}, \tag{4.8}$$

where r is the total number of rules generated, E is the new example, and C_R^{out} is the centre of the output fuzzy rule set R.

In covering algorithms, a seed example (SE) is selected to help in the generation of a new rule. This SE, for instance in RULES-5, is the first in the list of training examples that has not been covered by other previously created rules. This allows the creation of a consistent and general rule covering SE by employing a specific search process.

The output, C_R^{out}, is numerical and that every output value has to be fuzzified to determine the output fuzzy set before the rules are created. Therefore, the number is split into a fixed number of membership functions (Nf) that the user determines. The higher the Nf, the more accurate the created rule set.

4.5 The Data Sorting Method (DSM) Algorithm

There exist various data sorting algorithms, such as: Selection Sort, Insertion Sort, Bubble Sort, Quick Sort, Merge Sort, and Shell Sort, which are all commonly used for data sorting.

Given here the complete rule forming process of DSM that can be summarised as follows, where *T* is the set of training examples:

```
While there is an example in T not covered by any rule, DO
    • Compute  the Entropy of Attributes
    • Sort them from low to high
    • While the Attributes sub group exist
            • Compute the Entropy of each sub group
            • Compute examples total Entropy value (aggregated Entropy of each row)
    • END
    • Sort all rows (examples) from low to high based on aggregated value
    • Generate a rule using the RULES F algorithm
    • Delete the row and smaller examples covered by this rule
     END
```

Figure 4.5: Data sorting method algorithm(DSM).

4.6 Illustrative Problem

When using entropy values, rules are induced according to the values with higher knowledge gain by calculating the attributes and entropies of the values. The highly disordered and less important attributes for the example set are either taken into account slightly or ignored completely. Attributes are included in the rules in accordance with their level of importance. General rules can therefore be created by following this method. Entropy is broadly defined as the disorder of a given system. It is an important physical concept for which many disciplines have developed distinct entropy functions. Examples include thermodynamic entropy and topological entropy. As the disorder of a given system becomes greater, any increasing function can be regarded as the entropy function.

To illustrate the operation of the DSM method that was selected, the algorithm was applied to the weather problem in Table 2.1. A step by step execution of DSM using this data set is provided in Figure 4.3. For this simple data set, 5 epochs are required to complete the START PHASE and only one division of a rule is carried out. The resultant rule set after this phase consists of 5 rules present in their conditional part. The resultant rule set is the most general rule set for this data.

Start Phase: Initialise the rule set, RS.

```
Processing of data set:
        Compute and sort the entropy of Attributes
Processing of data set:
        Compute and sort the entropy of Examples
```
Iteration 1:
```
Processing of Example 1:
    RS = {}
        There is no rule yet to classify Example 1.
Processing of Example 1:
    RS = {}
        No rule can classify Example 1.
Processing of Example 1:
    RS = {Rule1}
        Grate Rule1: Outlook = overcast $\Rightarrow$ Play = Yes
            Rule1 classified examples 1, 2, 3, 4
                Delete examples covered by such rule and go to Phase 1
```
Iteration 2:
```
Processing of Example1:
    RS = {Rule1}
        There is no rule yet to classify Example 1.
Processing of Example 1:
    RS = {Rule1, Rule2}
        Grate Rule2: IF Humidity = high AND Outlook = sunny $\Rightarrow$ Play=No
            Rule2 classified examples 1, 2, 9
                Delete examples covered by such rule and go to Phase 1
```
Iteration 3:
```
Processing of Example 1:
    RS = {Rule1, Rule2}
        There is no rule yet to classify Example 1.
Processing of Example 1:
    RS = {Rule1, Rule2, Rule3}
        Grate Rule3 IF Outlook=rainy AND Windy=false $\Rightarrow$ Play= Yes
            Rule3 classified examples 3, 4, 6
                Delete examples covered by such rule and go to Phase 1
```
Iteration 4:
```
 Processing of Example 1:
    RS = {Rule1, Rule2, Rule3}
        No rule can classify Example 1.
Processing of Example 1:
    RS = {Rule1, Rule2, Rule3, Rule4}
        Grate Rule4: IF Outlook=sunny AND Humidity=normal $\Rightarrow$ Play=Yes
            Rule4 classified examples 1, 2
                Delete examples covered by such rule and go to Phase 1
```
Iteration 5:
```
Processing of Example 1:
    RS = {Rule1, Rule2, Rule3, Rule4}
        There is no rule yet to classify Example 1.
Processing of Example 1:
        RS = {Rule1, Rule2, Rule3, Rule4, Rule5}
            Grate Rule5: IF Outlook=rainy AND Windy=true $\Rightarrow$ Play=No
                Rule5 classified examples 1, 2
                    Delete examples covered by such rule and go to Phase 1
```

Figure 4.6: A step by step execution of DSM for the training set in Table 2.1.

4.6.1 Changing the order of data training set, first iteration

To illustrate the idea, of the set of 14 objects in the weather problem (Table 2.1), 9 are of class Yes and 5 are of class No.

The attributes and their potential values are given below:

Attributes	*Values*
A_1 = (Outlook)	Sunny, Overcast, Rainy
A_2 = (Temperature)	Hot, Mild, Cool
A_3 = (Humidity)	High, Normal
A_4 = (Windy)	False, True

The entropy is calculated for each attribute and value by using Equation 4.1:

• The first attribute, *outlook*, can take the values {*sunny, overcast, rain*}. The entropy for each value of the attribute will be calculated.

Five of the 14 objects have the first value (*sunny*). Of these, two are from class Yes, and three are from class No.

For (*sunny*), $E_{Outlook,Sunny}(2{,}3) = 0.972$ bits

And similarly

For (*overcast*), $E_{Outlook,Overcast}(4{,}0) = 0$ bit

For (*rain*), $E_{Outlook,Rainy}(3{,}2) = 0.972$ bit

From the above calculations, the entropy for attribute, *Outlook*, is calculated as:

$$E_{Outlook} = \frac{5}{14} \times E_{Outlook,Sunny} + \frac{4}{14} \times E_{Outlook,Overcast} + \frac{5}{14} \times E_{Outlook,Rainy}$$

$E_{Outlook} = 0.694$ bits

• The second attribute, *Temperature*, takes the values {*hot, mild and cool*}, and the entropy for each value of the attribute will be calculated below.

Four of the 14 objects have the first value (*hot*), two of them from class Yes and two from class No.

For (*hot*), $E_{Temperature,Hot}(2{,}2) = 1\text{ bit}$

And similarly

For (*mild*), $E_{Temperature,Mild}(4{,}2) = 0.920\text{ bit}$

For (*cool*), $E_{Temperature,Cool}(3{,}1) = 0.811\text{ bit}$

From the above calculations, the entropy for the attribute, *Temperature,* is found as:

$$E_{Temperature} = \frac{4}{14} \times E_{Temperature,Hot} + \frac{6}{14} \times E_{Temperature,Mild} + \frac{4}{14} \times E_{Temperature,Cool}$$

$$E_{Temperature} = 0.913\text{ bits}$$

• The third attribute, *Humidity*, can take the values {*high, normal*}, and the entropy for each value of the attribute will be calculated below.

Five of the 14 objects have the first value (*high*), three of them from class Yes and four from class No.

For (*high*), $E_{Humidity,High}(3{,}4) = 0.986\text{ bit}$

And similarly

For (*normal*), $E_{Humidity,Normal}(6{,}1) = 0.593\text{ bit}$

From the above calculations, it follows that the entropy for the *Humidity* attribute is given by:

$$E_{Humidity} = \frac{7}{14} \times E_{Humidity,High} + \frac{7}{14} \times E_{Humidity,Normal}$$

$$E_{Humidity} = 0.790\ bits$$

• The fourth attribute, *Windy,* takes values {*false, true*}, and the entropy for each value of the attribute will be calculated below.

Five of the 14 objects have the first value (*false*), six of them from class Yes and three from class No.

For (*false*), $E_{Windy,False}(6{,}3) = 0.919\text{ bit}$

And similarly

For (*true*), $E_{Windy,True}(3{,}2) = 0.972$ bit

From the above calculations, the entropy for attribute, Windy, is calculated as:

$$E_{Windy} = \frac{9}{14} \times E_{Windy,False} + \frac{5}{14} \times E_{Windy,True}$$

$E_{Windy} = 0.938$ bits

The following is an Example of the data sorting mothed. For the data set given in Table 2.1, there are four attributes: *Outlook, Temperature, Humidity*, and *Windy*.

DSM hence carries out three (the number of attributes minus 1) partitioning loops on the data set.

At Level = 0, the rule matrix (RM) contains all the 14 examples in Table 2.1.

Using these 14 examples, the Level 1 entropy for each attribute is given below using the formulas in equation (2.9).

$E_{Outlook} = 0.694$ bits

$E_{Temperature} = 0.913$ bits

$E_{Humidity} = 0.790\ bits$

$E_{Windy} = 0.938$ bits

From these entropy values, *Outlook* appears to have the lowest entropy, and therefore is the highest relevant attribute.

Using these 4 Attribute entropy values, the Level 2 attributes are sorted by their value from smallest to largest.

$E_{Outlook} = 0.694$ bits

$E_{Humidity} = 0.790\ bits$

$E_{Windy} = 0.938$ bits

$E_{Temperature} = 0.913$ bits

Table 4.1: Entropies of attributes and values in first iteration.

Attribute	Entropy (bit)	Value	Entropy (bit)	Gain Ratio
Outlook	0.694	Sunny	0.972	0.156
		Rainy	0.972	
		Overcast	0.00	
Temp	0.913	Hot	1.00	0.550
		Cool	0.920	
		Mild	0.811	
Humidity	0.790	High	0.986	0.151
		Normal	0.811	
Windy	0.938	False	0.919	0.051
		True	0.972	

Using these 4 Attribute entropy values and 14 attributes sub group value, the training data set is reordered as shown in Table 4.2.

Table 4.2: Reordered data set in first iteration.

No	*Outlook*	*Humidity*	*Temperature*	*Windy*	*Play*
1	overcast	normal	cool	true	yes
2	overcast	normal	hot	false	yes
3	overcast	high	mild	true	yes
4	overcast	high	hot	false	yes
5	rainy	normal	cool	false	yes
6	sunny	normal	cool	false	yes
7	rainy	normal	cool	true	no
8	rainy	normal	mild	false	yes
9	sunny	normal	mild	true	yes
10	rainy	high	mild	false	yes
11	sunny	high	mild	false	no
12	rainy	high	mild	true	no
13	sunny	high	hot	false	no
14	sunny	high	hot	false	no

4.6.2 Changing the order of data training set after the rule produced

The rule obtained by RULES-F with the data sorting method was produced using the reordered set of training data.

The first iteration ends here and the rule is obtained as:

IF Outlook = overcast $\Rightarrow$ *Play = Yes*

Examples 1, 2, 3, 4 covered by this rule are then deleted, and the new data set is reordered both in terms of the attributes and the examples for the second iteration.

Table 4.3: Training data set after first iteration.

No	*Outlook*	*Temperature*	*Humidity*	*Windy*	*Play*
1	sunny	hot	high	false	no
2	sunny	hot	high	false	no
3	rainy	mild	high	false	yes
4	rainy	cool	normal	false	yes
5	rainy	cool	normal	true	no
6	sunny	mild	high	false	no
7	sunny	cool	normal	false	yes
8	rainy	mild	normal	false	yes
9	sunny	mild	normal	true	yes
10	rainy	mild	high	true	no

4.6.3 Changing the order of data training set, second iteration

To illustrate the process, a set of objects in a new weather problem (Table 4.3) is used. Of the 10 objects, 5 are of class Yes and 5 are of class No.

The attributes and their values are given below:-

Attributes	*Values*
A_1 = (Outlook)	Sunny, Rainy
A_2 = (Temperature)	Hot, Mild, Cool
A_3 = (Humidity)	High, Normal
A_4 = (Windy)	False, True

The first attribute, *outlook* takes values of {*sunny, rain*}, the entropy for each value of the attribute will be calculated below.

Five of the 10 objects have the first value (*sunny*), two of them from class Yes and three from class No.

For (*sunny*), $E_{Outlook,Sunny}(2,3) = 0.972 \text{ bits}$

And similarly

For (*rain*), $E_{Outlook,Rainy}(3,2) = 0.972 \text{ bit}$

From the above calculations, the entropy for attribute, *Outlook*, is calculated as:

$$E_{Outlook} = \frac{5}{10} \times E_{Outlook,Sunny} + \frac{5}{10} \times E_{Outlook,Rainy}$$

$$E_{Outlook} = 0.972 \text{ bits}$$

• The second attribute, *Temperature,* has values {*hot, mild and cool*}. The entropy for each value of each attribute will be calculated here.

Two of the 10 objects have the first value (*hot*), none of them from class Yes and two from class No.

For (*hot*), $E_{Temperature,Hot}(0,2) = 0 \text{ bits}$

And similarly

For (*mild*), $E_{Temperature,Mild}(3,2) = 0.972 \text{ bits}$

For (*cool*), $E_{Temperature,Cool}(2,1) = 0.916 \text{ bit}$

From the above calculations, the entropy for attribute, *Temperature,* is found as:

$$E_{Temperature} = \frac{2}{10} \times E_{Temperature,Hot} + \frac{5}{10} \times E_{Temperature,Mild} + \frac{3}{10} \times E_{Temperature,Cool}$$

$$E_{Temperature} = 0.760 \text{ bits}$$

• The third attribute, *Humidity,* has values {*high, normal*}, and the entropy for each value of the attribute will be calculated here.

Five of the 10 objects have the first value (*high*), one of them from class Yes and four from class No.

For (*high*), $E_{Humidity,High}(1,4) = 0.721 \text{ bit}$

And similarly

For (*normal*), $E_{Humidity,Normal}(4,1) = 0.721 \text{ bit}$

From the above calculations, it follows that the entropy for the attribute, *Humidity*, turns out to be:

$$E_{Humidity} = \frac{5}{10} \times E_{Humidity,High} + \frac{5}{10} \times E_{Humidity,Normal}$$

$E_{Humidity} = 0.721$ bits

• The fourth attribute, *Windy*, has values of {*false*, *true*}, and the entropy for each value of the attribute will be calculated below.

Seven of the 10 objects have the first value (*false*), four of them from class Yes and three from class No.

For (*false*), $E_{Windy,False}(3,4) = 0.985$ bit

And similarly

For (*true*), $E_{Windy,True}(1,2) = 0.994$ bit

From the above calculations, the entropy for attribute, *Windy*, is calculated as:

$$E_{Windy} = \frac{7}{10} \times E_{Windy,False} + \frac{3}{10} \times E_{Windy,True}$$

$E_{Windy} = 0.994$ bits

Table 4.4: Entropies of attributes and values in second iteration.

Attribute	Entropy (bit)	Values	Entropy (bit)
Outlook	0.971	Sunny	0.972
		Rainy	0.972
Temperature	0.760	Hot	0.00
		Cool	0.916
		Mild	0.971
Humidity	0.721	High	0.721
		Normal	0.721
Windy	0.994	False	0.985
		True	0.916

Using these 4 Attribute entropy values and 10 attributes sub group value, the training data set is reordered as shown in Table 4.5.

Table 4.5: Reordered data set in second iteration.

No	*Humidity*	*Temperature*	*Outlook*	*Windy*	*Play*
1	high	hot	sunny	false	no
2	high	hot	sunny	false	no
3	normal	cool	rainy	true	no
4	normal	mild	sunny	true	yes
5	high	mild	rainy	true	no
6	normal	cool	rainy	false	yes
7	normal	cool	sunny	false	yes
8	high	mild	rainy	false	yes
9	high	mild	sunny	false	no
10	normal	mild	rainy	false	yes

4.6.4 Changing the order of training data set after the rule is produced

The rule obtained by RULES-F with the data sorting method was produced using a reordered set of training data:

IF Outlook = overcast $\Rightarrow$ *Play = Yes*

IF Humidity = high AND Outlook = sunny $\Rightarrow$ *Play = No*

Delete examples 1, 2, 9 covered by this rule.

The rule is obtained, and the data has to be reordered in terms of both the attributes and the examples.

Table 4.6: Training data set after second iteration.

No	*Humidity*	*Temperature*	*Outlook*	*Windy*	*Play*
1	normal	cool	rainy	true	no
2	normal	mild	sunny	true	yes
3	high	mild	rainy	true	no
4	normal	cool	rainy	false	yes
5	normal	cool	sunny	false	yes
6	high	mild	rainy	false	yes
7	normal	mild	rainy	false	yes

4.6.5 Changing the order of training data set on the third iteration

To illustrate the process, the set of objects in the new weather problem in (Table 4.6) are considered. Of the 7 objects, 5 are of class Yes and 2 are of class No.

The attributes and their values are given below:-

Attributes	*Values*
A_1= (Humidity)	High, Normal
A_2 = (Temperature)	Mild, Cool
A_3= (Outlook)	Sunny, Rainy
A_4 = (Windy)	False, True

• First the *Humidity* attribute, which can take the values {*high, normal*}, the entropy for each value of the attribute can be calculated.

Two of the 7 objects have the first value (*high*), one of them from class Yes and one from class No.

For (*high*), $E_{Humidity,High}(1,1) = 1\text{ bit}$

And similarly

For (*normal*), $E_{Humidity,Normal}(4,1) = 0.721\text{ bit}$

From the above calculations, it follows that the entropy for attribute, *Humidity*, turns out to be:

$$E_{Humidity} = \frac{2}{7} \times E_{Humidity,High} + \frac{5}{7} \times E_{Humidity,Normal}$$

$$E_{Humidity} = 0.920\text{ bits}$$

• The second attribute, *Temperature*, has values {*cool, mild}*, and the entropy for each value of the attribute will hence be calculated here.

Three of the 7 objects have the first value (*cool*), two of them from class Yes and one from class No.

For (*cool*), $E_{Temperature,Cool}(2{,}1) = 0.967$ bit

And similarly

For (*mild*), $E_{Temperature,Mild}(3{,}1) = 0.527$ bits

From the above calculations, the entropy for attribute, *Temperature*, is found as:

$$E_{Temperature} = \frac{3}{7} \times E_{Temperature,Cool} + \frac{4}{7} \times E_{Temperature,Mild}$$

$E_{Temperature} = 0.715$ bits

• The third attribute, *outlook*, can have values {*sunny, rain*}. The entropy for each value of the attribute will be calculated here.

Five of the 7 objects have the first value (*Rainy*), three of them from class Yes and two from class No.

For (*rain*), $E_{Outlook,Rainy}(3{,}2) = 0.971$ bit

And similarly

For (*sunny*), $E_{Outlook,Sunny}(2{,}0) = 0$ bits

From the above calculations, the entropy for attribute, Outlook, is calculated as:

$$E_{Outlook} = \frac{5}{7} \times E_{Outlook,Rainy} + \frac{2}{7} \times E_{Outlook,Sunny}$$

$E_{Outlook} = 0.693$ bits

• The fourth attribute, *Windy*, takes values {*false, true*}, and the entropy for each value of each attribute will be calculated below.

Seven of the 7 objects have the first value (*false*), four of them from class Yes and three from class No.

For (*true*), $E_{Windy,True}(1{,}2) = 0.967$ bit

For (*false*), $E_{Windy,False}(4{,}0) = 0.860$ bit

And similarly

From the above calculations, the entropy for attribute, *Windy*, is calculated as:

$$E_{Windy} = \frac{4}{7} \times \ E_{Windy,False} + \frac{3}{7} \times E_{Windy,True}$$

$$E_{Windy} = 0.905 \text{ bits}$$

Using these 4 Attribute entropy values, the attributes are sorted by their value as shown in Table 4.8.

Table 4.7: Entropies of attributes and values in third iteration.

Attribute	Entropy (bit)	Values	Entropy (bit)
Outlook	0.693	Sunny	0.000
		Rainy	0.971
Temperature	0.715	Hot	0.00
		Cool	0.967
		Mild	0.527
Humidity	0.920	High	1.000
		Normal	0.721
Windy	0.905	False	0.860
		True	0.967

Using these 4 Attribute entropy values and 7 attributes sub group value, the training data set is reordered as shown in Table 4.8.

Table 4.8: Reordered data set in third iteration.

No	*Outlook*	*Temperature*	*Windy*	*Humidity*	*Play*
1	sunny	mild	true	normal	yes
2	sunny	cool	false	normal	yes
3	rainy	mild	false	normal	yes
4	rainy	mild	false	high	yes
5	rainy	mild	true	high	no
6	rainy	cool	false	normal	yes
7	rainy	cool	true	normal	no

4.6.6 Changing the order of data training set after the rule produced

The rule obtained by RULES-F with the data sorting method was produced using a reordered set of training data:

IF Outlook = overcast $\Rightarrow$ *Play = Yes*

IF Humidity = high AND Outlook = sunny $\Rightarrow$ *Play = No*

IF Outlook = rainy AND Windy = false $\Rightarrow$ *Play = Yes*

Delete examples 3, 4, and 6 covered by this rule.

Table 4.9: Training data set after third iteration.

No	*Outlook*	*Temperature*	*Windy*	*Humidity*	*Play*
1	sunny	mild	true	normal	yes
2	sunny	cool	false	normal	yes
3	rainy	mild	true	high	no
4	rainy	cool	true	normal	no

And similarly:

IF Outlook = overcast $\Rightarrow$ *Play = Yes*

IF Humidity = high AND Outlook = sunny $\Rightarrow$ *Play = No*

IF Outlook = rainy AND Windy = false $\Rightarrow$ *Play = Yes*

IF Outlook = sunny AND Humidity = normal $\Rightarrow$ *Play = Yes*

IF Outlook = rainy AND Windy = true $\Rightarrow$ *Play = No*

The rules are obtained. There are NO remaining unclassified examples in the example set and the procedure terminates at this point.

4.7 The Data Sorting Method (DSM)

Chapter 3 introduced a data ordering method, a technique that orders the training data. In this chapter, improvements to the Data Ordering Mothed (in Chapter 3) are discussed and their implementation in a new version of the DSM, is described. This DSM can reorder attributes and give priority to the one with higher knowledge values. To facilitate the processing of attributes by DSM, modifications are introduced to the rule representation scheme. A new procedure is proposed for selecting an attribute to give more general and compact rules. The entropy for each attribute is calculated, and they are ranked from small to large or from left to right.

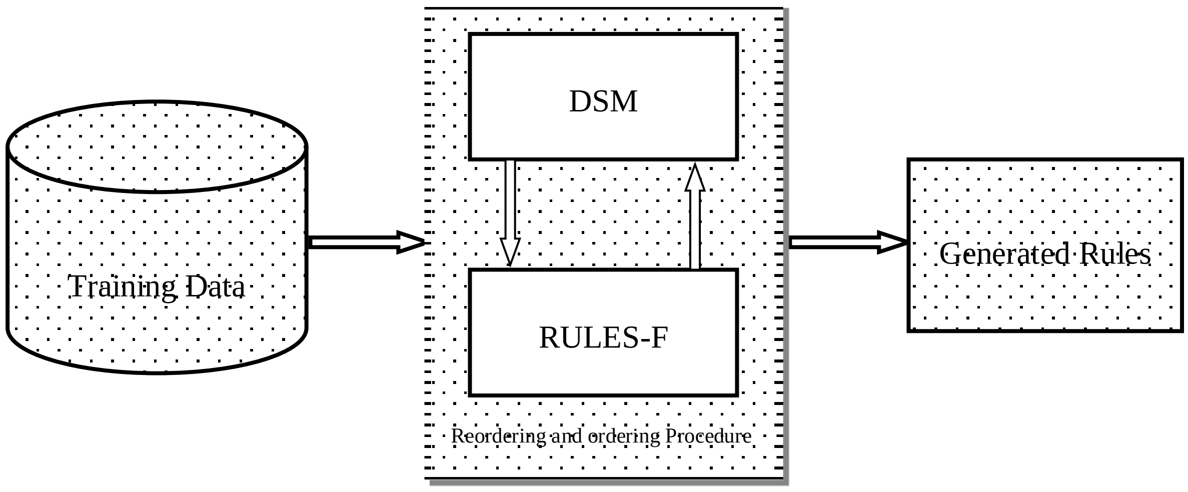


Figure 4.4: RULES-F algorithm with proposed Data Sorting Method.

4.8 Experiments

The ordering (DSM) techniques suggested in this chapter have been tested thoroughly on a population of 14 datasets, all of which have been downloaded from the repository of machine learning databases held at the University of California at Irvine (UCI) (Blake and Merz, 1998). These datasets come from a variety of domains and have been summarised in Table 4.10. The only sampled dataset used in this study is the Weather problem, with 14 instances.

Table 4.10: Summary of data sets used in experiments.

No	Data Set	Attributes	Classes	#Examples
1	Tic-Tac-Toe	09	02	958
2	Weather	04	02	14
3	Car	06	02	1728
4	Haberman	03	02	306
5	Chess	06	02	28056
6	Zoo	17	07	101
7	Post-operative	08	02	90
8	Dermatology	33	06	366
9	Heart	13	02	270
10	Monk1	07	02	432
11	Monk2	07	02	432
12	Balance-scale	04	03	625
13	Contracpetive	09	03	1473
14	king-rook-vs-king	06	18	28056

4.9 Results and Discussions

Six different orderings were used (Original, Data Ordering Method (DOM), Data Sorting Method (DSM), Random1, Random2 and Random3) to measure the effect of the order of SE on the rule sets created using RULES-3 Plus and RULES-F.

The results from different pruning scales show that the choice of ordering method has a measurable effect on the number of rule sets created by the two algorithms. For example, with pruning at 70 %, the percentage difference in number of rules is up to 19.30 % and 13.80 % for RULES-F and RULES-3 Plus respectively.

Table 4.12 displays the highest number of deviations in the number of rules generated by each algorithm when the six different ordering systems are applied to each data set.

Table 4.11: Average number of rules created using six orderings on two algorithms (with pruning 90 %).

		Average deviation in number of rules created	
No	Data set	RULES-F	RULES-3 Plus
1	Tic-Tac-Toe	2	9
2	Weather Problem	1	1
3	Car	1	4
4	Haberman	13	7
5	Chess	14	73
6	Zoo	2	0
7	Post-Operative	5	5
8	Dermatology	7	5
9	Heart	2	4
10	Monk1	0	7
11	Monk2	13	50
12	Balance-scale	11	0
13	Contraceptive	125	0
14	King-rook-vs-king	844	1617
Maximum		844	1617
Average		74.214	127.285

Table 4.12: Average number of rules created using six orderings.

		Average Number of Rules					
	Data set	Original	DOM	DSM	Random1	Random2	Random3
1	Tic-Tac-Toe	24.5	21.5	20	19	20	23.5
2	Weather Problem	5	5.5	6	5.5	5.5	5.5
3	Car	248	246.5	247	246.5	247	246
4	Haberman	154	148.5	150.5	150.5	150.5	145.5
5	Chess	71.5	42.5	39.5	46.5	38.5	38
6	Zoo	25	25	25	25	25	25.5
7	Post-Operative	36.5	36.5	37.5	34.5	34.5	34.5
8	Dermatology	39	39	38.5	42	36.5	39
9	Heart	12	13.5	13.5	11.5	11.5	11.5
10	Monk1	25.5	22	25	27	27	27.5
11	Monk2	154.5	157.5	152	145.5	152	155.5
12	Balance-scale	175.5	179.5	176.5	171.5	172	172.5
13	Contraceptive	735	706.5	720.5	720.5	735	714
14	King-rook-vs-king	14662	14151.5	14122	14666	14695	14715

From Table 4.12 above, six ordering methods were on the data from two algorithms, RULES-F and RULES-3 Plus. Using the original ordering implies that there is no order applied to the data. Meanwhile, DOM is a data ordering method created as shown in Chapter 3, and DSM is a data sorting method created in Section 4.7. Random 1, random 2 and random 3 all are applied to the data randomly using the two algorithms.

When six varieties of varying ordering are applied, different numbers of rules are created by the algorithms. For the example of tic-tac-toe, the rules generated are stated above.

Rules are generated with high confidence, and each rule generated is defined as a binary partition of the item set. As per the above results table, the rules generated from a frequent item set were low, hence decreasing the computational cost. It can be observed that the set of rules generated from the same set of items has more support than a set of rules generated from different item sets.

It can be safely concluded that the example used to the ordering, in this case tic-tac-toe, has close to no effect on the number of rules generated by the algorithms. Overall there is very little distinction between the results acquired. It can also be concluded that the order of the originally produced results that the worst order to use. However, when looking in more detail at the normal deviations among the information sets, the new requesting technique appears to be exceptionally encouraging as it gives the best results. It ought to be specified that a large portion of the handling time used for the proposed information requesting strategy is spent on surveying entropy values. This is done for each characteristic worth pair that is utilised by the mathematical statement 1, as a part of each case. The data gathered from assessing and evaluating the generated rules can be further used in covering algorithms. Therefore the cost incurred in computing the rules is not expensive and the sampling data can be pre-processed.

Table 4.13: The results obtained using 2 algorithms with pruning required 70 %.

			RULES F		RULES3Plus	
No	Data set	Ordering	No of Rules	Classified	No of Rules	Classified
1	Tic-Tac-Toe	Original	12	100	37	100
		DOM	10	100	33	100
		DSM	12	100	28	100
		Random1	12	100	26	100
		Random2	13	100	27	100
		Random3	12	100	35	100
2	Weather Problem	Original	5	100	5	100
		DOM	5	100	6	100
		DSM	6	100	6	100
		Random1	6	100	5	100
		Random2	6	100	5	100
		Random3	6	100	5	100
3	Car	Original	246	100	250	100
		DOM	246	100	247	100
		DSM	246	100	248	100
		Random1	246	100	247	100
		Random2	246	100	248	100
		Random3	246	100	246	100
4	Haberman	Original	128	93.791	180	96.732
		DOM	116	92.157	181	96.732
		DSM	120	92.810	181	97.059
		Random1	120	92.810	181	96.732
		Random2	120	92.810	181	96.732
		Random3	109	91.830	182	97.831
5	Chess	Original	12	93.586	131	100
		DOM	23	96.996	62	100
		DSM	21	98.310	58	100
		Random1	29	98.404	64	100
		Random2	18	97.121	59	100
		Random3	13	97.278	63	100
6	Zoo	Original	10	100	40	100
		DOM	10	100	40	100
		DSM	10	100	40	100
		Random1	10	100	40	100
		Random2	10	100	40	100
		Random3	11	100	40.	100
7	Post-Operative	Original	35	93.333	38	93.333
		DOM	30	92.222	43	92.222
		DSM	34	93.211	41	93.211
		Random1	29	92.222	40	92.222

		Random2	29	92.222	40	92.222
		Random3	29	92.222	40	92.222
8	Dermatology	Original	26	98.907	52	100
		DOM	25	99.453	53	100
		DSM	19	95.902	58	100
		Random1	31	99.727	53	100
		Random2	24	98.901	49	100
		Random3	27	100	51	100
9	Heart	Original	1	91.979	23	95.722
		DOM	1	91.979	26	95.722
		DSM	1	91.979	26	95.722
		Random1	1	91.979	22	95.722
		Random2	1	91.979	22	95.722
		Random3	1	91.979	22	95.722
10	Monk1	Original	10	99.769	41	100
		DOM	10	99.769	34	100
		DSM	10	99.769	40	100
		Random1	10	99.769	44	100
		Random2	10	99.769	44	100
		Random3	10	99.769	45	100
11	Monk2	Original	36	76.157	273	100
		DOM	29	74.306	286	100
		DSM	44	70.602	260	100
		Random1	26	70.602	265	100
		Random2	38	74.537	266	100
		Random3	43	76.852	268	100
12	Balanc-scale	Original	48	84.642	303	100
		DOM	56	86.561	303	100
		DSM	50	85.446	303	100
		Random1	40	82.880	303	100
		Random2	41	83.044	303	100
		Random3	42	84.325	303	100
13	Contracpetive	Original	588	91.174	882	95.180
		DOM	569	90.495	844	95.316
		DSM	590	91.242	851	95.441
		Random1	565	90.970	876	95.384
		Random2	575	91.581	895	95.311
		Random3	549	90292	879	95384
14	king-rook-vs-king	Original	8543	89.85328	20781	100
		DOM	7148	87.872	21155	100
		DSM	7160	89.675	21084	100
		Random1	7188	88.975	22144	100
		Random2	7189	88.869	22201	100
		Random3	7163	88.868	22267	100
Percentage difference%			**19.35074**		**13.80411**	

Table 4.14: The results obtained using 2 algorithms with pruning required 75 %.

			RULES F		RULES3Plus	
No	Data set	Ordering	No of Rules	Classified	No of Rules	Classified
1	Tic-Tac-Toe	Original	17	99.26931	37	100
		DOM	17	99.164925	33	100
		DSM	16	99.164925	28	100
		Random1	17	99.164925	30	100
		Random2	16	99.164925	30	100
		Random3	16	99.164925	29	100
2	Weather Problem	Original	5	100	5	100
		DOM	5	100	6	100
		DSM	6	100	6	100
		Random1	6	100	6	100
		Random2	6	100	6	100
		Random3	6	100	6	100
3	Car	Original	246	100	250	100
		DOM	246	100	247	100
		DSM	246	100	248	100
		Random1	246	100	247	100
		Random2	246	100	248	100
		Random3	246	100	246	100
4	Haberman	Original	128	93.790848	180	96.732025
		DOM	116	92.15686	181	96.732025
		DSM	120	92.810455	181	97.058823
		Random1	120	92.810455	181	96.732025
		Random2	120	92.810455	181	96.732025
		Random3	109	91.830063	182	97.058823
5	Chess	Original	12	93.585732	131	100
		DOM	23	96.996246	62	100
		DSM	21	98.310387	58	100
		Random1	29	98.404259	64	100
		Random2	18	97.121399	59	100
		Random3			63	100
6	Zoo	Original	10	100	40	100

		DOM	10	100	40	100
		DSM	10	100	40	100
		Random1	11	100	40	100
		Random2	11	100	40	100
		Random3	11	100	40	100
7	Post-Operative	Original	35	93.333336	38	93.333336
		DOM	32	92.222221	43	92.222221
		DSM	34	93.333336	43	93.333336
		Random1	29	92.222221	40	92.222221
		Random2	29	92.222221	40	92.222221
		Random3	29	92.222221	40	92.222221
8	Dermatology	Original	28	99.180328	52	100
		DOM	27	99.726776	53	100
		DSM	26	99.453552	58	100
		Random1	30	100	50	100
		Random2	28	98.907104	44	100
		Random3	32	99.726776	48	100
9	Heart	Original	1	91.978607	23	95.721924
		DOM	1	91.978607	26	95.721924
		DSM	1	91.978607	26	95.721924
		Random1	1	91.978607	23	95.721924
		Random2	1	91.978607	22	97.326202
		Random3	1	91.978607	22	97.326202
10	Monk1	Original	10	99.768517	41	100
		DOM	10	99.768517	34	100
		DSM	10	99.768517	40	100
		Random1	10	99.768517	44	100
		Random2	10	99.768517	44	100
		Random3	10	99.768517	41	100
11	Monk2	Original	36	76.15741	273	100
		DOM	29	74.305557	286	100
		DSM	44	77.314812	260	100
		Random1	26	70.601852	265	100
		Random2	38	74.537041	266	100
		Random3	43	76.851852	268	100
12	Balanc-scale	Original	62	88.639999	303	100

		DOM	81	90.879997	303	100
		DSM	72	89.919998	303	100
		Random1	58	88.480003	303	100
		Random2	64	88.959999	303	100
		Random3	63	88.800003	303	100
13	Contracpetive	Original	630	93.143242	882	95.179909
		DOM	640	93.822128	844	95.315681
		DSM	636	93.346909	851	95.044128
		Random1	618	93.279022	879	95.179909
		Random2	612	92.735916	878	95.247795
		Random3	626	93.686356	892	95.383568
14	king-rook-vs-king	Original	11446	93.580696	20781	100
		DOM	10690	91.705872	21155	100
		DSM	10796	92.026665	21084	100
		Random1	10712	92.308243	22197	100
		Random2	10763	92.322495	22209	100
		Random3	10726	92.372398	22322	100
Percentage difference%			**15.805808**		**13.904885**	

Table 4.15: The results obtained using 2 algorithms with pruning required 80 %.

			RULES F		RULES3Plus	
No	Data set	Ordering	No of Rules	Classified	No of Rules	Classified
1	Weather Problem	Original	5	100	5	100
		DOM	5	100	6	100
		DSM	6	100	6	100
		Random1	5	100	6	100
		Random2	5	100	5	100
		Random3	5	100	5	100
2	Tic-Tac-Toe	Original	22	100	37	100
		DOM	22	100	33	100
		DSM	22	100	28	100
		Random1	23	100	35	100
		Random2	22	100	27	100
		Random3	22	100	28	100
3	Car	Original	246	100	250	100
		DOM	246	100	247	100
		DSM	246	100	248	100
		Random1	246	100	247	100
		Random2	246	100	248	100
		Random3	246	100	246	100
4	Haberman	Original	128	93.790848	180	96.732025
		DOM	116	92.15686	181	96.732025
		DSM	120	92.810455	181	97.058823
		Random1	120	92.810455	181	96.732025
		Random2	120	92.810455	181	96.732025
		Random3	109	91.830063	182	97.058823
5	Chess	Original	12	93.585732	131	100
		DOM	23	96.996246	62	100
		DSM	21	98.310387	58	100
		Random1	29	98.404259	64	100
		Random2	18	97.121399	59	100
		Random3	13	97.277847	63	100
6	Zoo	Original	10	100	40	100

		DOM	10	100	40	100
		DSM	10	100	40	100
		Random1	10	100	40	100
		Random2	10	100	40	100
		Random3	11	100	40	100
7	Post-Operative	Original	35	93.333336	38	93.333336
		DOM	32	92.222221	43	92.222221
		DSM	34	93.333336	43	93.333336
		Random1	29	92.222221	40	92.222221
		Random2	29	92.222221	40	92.222221
		Random3	29	92.222221	40	92.222221
8	Dermatology	Original	29	100	52	100
		DOM	28	100	53	100
		DSM	30	100	58	100
		Random1	33	100	48	100
		Random2	29	99.726776	48	100
		Random3	29	100	52	100
9	Heart	Original	1	91.978607	23	95.721924
		DOM	1	91.978607	26	95.721924
		DSM	1	91.978607	26	95.721924
		Random1	2	92.513367	24	95.721924
		Random2	1	91.978607	21	97.326202
		Random3	1	91.978607	21	97.326202
10	Monk1	Original	10	99.768517	41	100
		DOM	10	99.768517	34	100
		DSM	10	99.768517	40	100
		Random1	10	99.768517	42	100
		Random2	10	99.768517	42	100
		Random3	10	99.768517	42	100
11	Monk2	Original	112	87.037041	273	100
		DOM	120	88.425926	286	100
		DSM	133	91.203705	260	100
		Random1	107	83.796295	262	100
		Random2	108	84.722221	264	100
		Random3	110	85.416664	263	100
12	Balanc-scale	Original	118	94.239998	303	100

		DOM	131	97.120003	303	100
		DSM	134	96.959999	303	100
		Random1	117	95.199997	303	100
		Random2	119	95.199997	303	100
		Random3	122	96.160004	303	100
13	Contracpetive	Original	630	93.143242	882	95.179909
		DOM	640	93.822128	844	95.315681
		DSM	636	93.346909	851	95.044128
		Random1	618	93.279022	879	95.179909
		Random2	612	92.735916	878	95.247795
		Random3	626	93.686356	892	95.383568
14	king-rook-vs-king	Original	14846	95.865439	20781	100
		DOM	12301	92.964256	21155	100
		DSM	12399	94.136891	21084	100
		Random1	12412	94.314699	22233	100
		Random2	12765	94.437829	22245	100
		Random3	12755	94.296438	22265	100
Percentage difference%			**17.11552**		**13.60019**	

Table 4.16: The results obtained using 2 algorithms with pruning required 85 %.

			RULES F		RULES3Plus	
No	Data set	Ordering	No of Rules	Classified	No of Rules	Classified
1	Weather Problem	Original	5	100	5	100
		DOM	5	100	6	100
		DSM	6	100	6	100
		Random1	6	100	6	100
		Random2	6	100	6	100
		Random3	6	100	6	100
2	Tic-Tac-Toe	Original	22	100	37	100
		DOM	22	100	33	100
		DSM	22	100	28	100
		Random1	24	100	34	100
		Random2	22	100	29	100
		Random3	23	100	29	100
3	Car	Original	246	100	250	100
		DOM	246	100	247	100
		DSM	246	100	248	100
		Random1	246	100	248	100
		Random2	246	100	249	100
		Random3	246	100	248	100
4	Haberman	Original	179	96.732025	180	96.732025
		DOM	179	96.732025	181	96.732025
		DSM	180	97.058823	181	97.058823
		Random1	178	97.058823	181	97.058823
		Random2	171	97.058823	181	97.058823
		Random3	176	97.058823	182	97.058823
5	Chess	Original	12	93.585732	131	100
		DOM	23	96.996246	62	100
		DSM	21	98.310387	58	100
		Random1	29	98.404259	64	100
		Random2	18	97.121399	59	100
		Random3	13	97.277847	63	100
6	Zoo	Original	10	100	40	100

		DOM	10	100	40	100
		DSM	10	100	40	100
		Random1	10	100	40	100
		Random2	10	100	40	100
		Random3	11	100	40	100
7	Post-Operative	Original	35	93.333336	38	93.333336
		DOM	32	92.222221	43	92.222221
		DSM	34	93.333336	43	93.333336
		Random1	32	93.333336	42	93.333336
		Random2	32	93.333336	42	93.333336
		Random3	32	93.333336	42	93.333336
8	Dermatology	Original	28	100	52	100
		DOM	27	99.726776	53	100
		DSM	28	100	58	100
		Random1	27	100	52	100
		Random2	27	98.907104	54	100
		Random3	29	99.726776	49	100
9	Heart	Original	1	91.978607	23	95.721924
		DOM	1	91.978607	26	95.721924
		DSM	1	91.978607	26	95.721924
		Random1	1	91.978607	21	95.721924
		Random2	1	91.978607	21	95.721924
		Random3	1	91.978607	23	95.721924
10	Monk1	Original	10	99.768517	41	100
		DOM	10	99.768517	34	100
		DSM	10	99.768517	40	100
		Random1	10	99.768517	42	100
		Random2	10	99.768517	42	100
		Random3	10	99.768517	46	100
11	Monk2	Original	139	91.203705	273	100
		DOM	147	93.055557	286	100
		DSM	166	96.759262	260	100
		Random1	148	93.055557	262	100
		Random2	143	91.898148	263	100
		Random3	142	91.435188	263	100
12	Balanc-scale	Original	167	97.440002	303	100

		DOM	175	98.400002	303	100
		DSM	170	97.760002	303	100
		Random1	162	96.480003	303	100
		Random2	162	96.959999	303	100
		Random3	160	96	303	100
13	Contracpetive	Original	670	94.704681	882	95.179909
		DOM	679	94.976242	844	95.315681
		DSM	685	95.112015	851	95.044128
		Random1	664	95.044128	903	95.247795
		Random2	674	94.908348	877	95.247795
		Random3	670	95.044128	890	95.179909
14	king-rook-vs-king	Original	18520	97.786936	20781	100
		DOM	18482	97.786936	21155	100
		DSM	18475	97.786936	21084	100
		Random1	18530	97.786936	22279	100
		Random2	18528	97.786936	22240	100
		Random3	18536	97.786936	22262	100
Percentage difference%			**10.10078**		**13.84475**	

Table 4.17: The results obtained using 2 algorithms with Pruning required 90 %.

			RULES F		RULES3Plus	
No	Data set	Ordering	No of Rules	Classified	No of Rules	Classified
1	Weather Problem	Original	5	100	5	100
		DOM	5	100	6	100
		DSM	5	100	6	100
		Random1	5	100	5	100
		Random2	5	100	5	100
		Random3	6	100	5	100
2	Tic-Tac-Toe	Original	22	100	37	100
		DOM	22	100	33	100
		DSM	22	100	28	100
		Random1	24	100	34	100
		Random2	22	100	29	100
		Random3	23	100	29	100
3	Car	Original	246	100	250	100
		DOM	242	100	247	100
		DSM	246	100	248	100
		Random1	246	100	250	100
		Random2	248	100	247	100
		Random3	248	100	250	100
4	Haberman	Original	179	96	180	96
		DOM	179	97	181	96
		DSM	170	97	181	97
		Random1	178	97	181	97
		Random2	179	97	181	97
		Random3	182	97	182	97
5	Chess	Original	22	93.585	131	100
		DOM	23	96.996	62	100
		DSM	20	98.310	58	100
		Random1	29	98.404	64	100
		Random2	18	97.112	59	100
		Random3	23	97.277	63	100
6	Zoo	Original	10	100	40	100
		DOM	10	100	40	100
		DSM	10	100	40	100
		Random1	11	100	40	100
		Random2	11	100	40	100
		Random3	11	100	40	100
7	Post-Operative	Original	35	92.333	38	93.333
		DOM	32	93.345	43	92.221

		DSM	30	93.333	43	93.33
		Random1	32	93.333	42	93.333
		Random2	32	93.333	42	93.333
		Random3	32	93.333	42	93.333
8	Dermatology	Original	29	100	52	100
		DOM	28	100	51	100
		DSM	28	100	50	100
		Random1	33	100	54	100
		Random2	29	99.726	54	100
		Random3	30	100	52	100
9	Heart	Original	1	91.978	23	95.721
		DOM	1	91.978	26	95.721
		DSM	1	91.978	26	95.721
		Random1	2	94.652	23	97.326
		Random2	2	94.652	23	97.326
		Random3	2	94.652	23	97.326
10	Monk1	Original	10	99.768517	41	100
		DOM	10	99.768517	34	100
		DSM	10	99.768517	40	100
		Random1	10	99.768517	44	100
		Random2	10	99.768517	44	100
		Random3	10	99.768517	41	100
11	Monk2	Original	36	76.15741	273	100
		DOM	29	74.305557	286	100
		DSM	44	77.314812	260	100
		Random1	26	70.601852	265	100
		Random2	38	74.537041	266	100
		Random3	43	76.851852	268	100
12	Balanc-scale	Original	198	99.040	303	100
		DOM	197	99.040	303	100
		DSM	195	99.040	303	100
		Random1	198	99.040	303	100
		Random2	198	99.040	303	100
		Random3	197	99.040	303	100
13	Contracpetive	Original	687	95.180	882	95.180
		DOM	688	95.441	844	95.31
		DSM	679	95.441	851	95.044
		Random1	673	95.180	888	95.180
		Random2	687	95.441	874	95.316
		Random3	688	95.180	870	95.180
14	king-rook-vs-king	Original	18308	99.889	20781	100
		DOM	19028	99.889	21155	100
		DSM	18002	99.889	20084	100
		Random1	19107	99.889	22279	100

		Random2	19109	99.889	22240	100
		Random3	19002	99.889	22262	100
Percentage difference%			**12.14003**		**10.31401**	

Table 4.18: The results obtained using 2 algorithms with pruning required 95 %.

			RULES F		RULES3Plus	
No	Data set	Ordering	No of Rules	Classified	No of Rules	Classified
1	Weather Problem	Original	5	100	5	100
		DOM	5	100	6	100
		DSM	6	100	6	100
		Random1	5	100	5	100
		Random2	5	100	5	100
		Random3	5	100	5	100
2	Tic-Tac-Toe	Original	22	100	37	100
		DOM	22	100	33	100
		DSM	22	100	28	100
		Random1	24	100	34	100
		Random2	22	100	29	100
		Random3	23	100	29	100
3	Car	Original	246	100	250	100
		DOM	246	100	247	100
		DSM	246	100	248	100
		Random1	246	100	251	100
		Random2	246	100	250	100
		Random3	246	100	246	100
4	Haberman	Original	179	96.73203	180	96.73203
		DOM	179	96.73203	181	96.73203
		DSM	180	97.05882	181	97.05882
		Random1	178	97.05882	181	97.05882
		Random2	171	97.05882	181	97.05882
		Random3	176	97.05882	182	97.05882
5	Chess	Original	12	93.58573	131	100
		DOM	23	96.99625	62	100
		DSM	21	98.31039	58	100

		Random1	29	98.40426	64	100
		Random2	18	97.1214	59	100
		Random3	13	97.27785	63	100
6	Zoo	Original	10	100	40	100
		DOM	10	100	40	100
		DSM	10	100	40	100
		Random1	10	100	40	100
		Random2	10	100	40	100
		Random3	10	100	40	100
7	Post-Operative	Original	35	93.33334	38	93.33334
		DOM	32	92.22222	43	92.22222
		DSM	34	93.33334	43	93.33334
		Random1	32	93.33334	40	93.33334
		Random2	34	92.22222	39	92.22222
		Random3	34	92.22222	39	92.22222
8	Dermatology	Original	29	100	52	100
		DOM	28	100	53	100
		DSM	30	100	58	100
		Random1	33	100	50	100
		Random2	28	100	52	100
		Random3	32	100	50	100
9	Heart	Original	13	94.11765	23	95.72192
		DOM	14	94.6524	26	95.72192
		DSM	12	94.6524	26	95.72192
		Random1	15	97.3262	21	97.3262
		Random2	15	97.3262	21	97.3262
		Random3	3	91.97861	23	95.72192
10	Monk1	Original	10	99.76852	41	100
		DOM	10	99.76852	34	100
		DSM	10	99.76852	40	100
		Random1	10	99.76852	39	100
		Random2	10	99.76852	39	100

		Random3	10	99.76852	39	100
11	Monk2	Original	254	100	273	100
		DOM	254	100	286	100
		DSM	254	100	260	100
		Random1	254	100	265	100
		Random2	254	100	270	100
		Random3	254	100	262	100
12	Balanc-scale	Original	230	99.04	303	100
		DOM	235	99.04	303	100
		DSM	234	99.04	303	100
		Random1	231	99.04	303	100
		Random2	229	99.04	303	100
		Random3	230	99.04	303	100
13	Contracpetive	Original	677	95.04413	882	95.17991
		DOM	686	95.17991	844	95.31568
		DSM	688	95.04413	851	95.04413
		Random1	679	95.04413	888	95.17991
		Random2	673	95.17991	874	95.31568
		Random3	687	95.04413	870	95.17991
14	king-rook-vs-king	Original	19230	99.9893	20781	100
		DOM	19228	99.9893	21155	100
		DSM	19226	99.9893	21084	100
		Random1	19238	99.9893	22374	100
		Random2	19237	99.9893	22279	100
		Random3	19246	99.9893	22290	100
Percentage difference%			**14.08436**		**13.04984**	

Table 4.19: The results obtained using 2 algorithms with pruning required 100 %.

			RULES F		RULES3Plus	
No	Data set	Ordering	No of Rules	Classified	No of Rules	Classified
1	Weather Problem	Original	5	100	5	100
		DOM	5	100	6	100
		DSM	6	100	6	100
		Random1	5	100	5	100
		Random2	5	100	5	100
		Random3	5	100	5	100
2	Tic-Tac-Toe	Original	22	100	37	100
		DOM	22	100	33	100
		DSM	22	100	28	100
		Random1	24	100	34	100
		Random2	22	100	29	100
		Random3	23	100	29	100
3	Car	Original	246	100	250	100
		DOM	246	100	247	100
		DSM	246	100	245	100
		Random1	246	100	246	100
		Random2	246	100	247	100
		Random3	246	100	249	100
4	Haberman	Original	179	96.732025	180	96.732025
		DOM	173	97.058823	180	97.058823
		DSM	179	97.058823	177	96.732025
		Random1	178	97.058823	181	97.058823
		Random2	175	97.058823	182	97.058823
		Random3	175	97.058823	182	97.058823
5	Chess	Original	39	100	131	100
		DOM	40	100	95	100
		DSM	37	100	90	100
		Random1	38	100	58	100
		Random2	37	100	58	100
		Random3	40	100	63	100
6	Zoo	Original	37	100	58	100

		DOM	40	100	63	100
		DSM	37	100	58	100
		Random1	40	100	63	100
		Random2	37	100	58	100
		Random3	40	100	63	100
7	Post-Operative	Original	36	93.333336	38	93.333336
		DOM	34	92.222221	43	92.222221
		DSM	32	93.333336	40	93.333336
		Random1	32	92.222221	39	92.222221
		Random2	32	92.222221	39	92.222221
		Random3	32	92.222221	39	92.222221
8	Dermatology	Original	29	100	52	100
		DOM	28	100	53	100
		DSM	28	100	52	100
		Random1	30	100	49	100
		Random2	31	100	52	100
		Random3	31	100	46	100
9	Heart	Original	19	95.721924	23	95.721924
		DOM	18	95.721924	26	95.721924
		DSM	18	95.721924	24	95.721924
		Random1	18	97.326202	25	97.326202
		Random2	18	97.326202	25	97.326202
		Random3	17	97.326202	21	97.326202
10	Monk1	Original	29	100	41	100
		DOM	29	100	34	100
		DSM	29	100	40	100
		Random1	29	100	41	100
		Random2	29	100	41	100
		Random3	29	100	41	100
11	Monk2	Original	254	100	273	100
		DOM	254	100	254	100
		DSM	254	100	250	100
		Random1	254	100	267	100
		Random2	254	100	265	100
		Random3	254	100	260	100
12	Balanc-scale	Original	303	100	303	100

		DOM	303	100	303	100
		DSM	303	100	303	100
		Random1	303	100	303	100
		Random2	303	100	303	100
		Random3	303	100	303	100
13	Contracpetive	Original	686	95.179909	882	95.179909
		DOM	695	95.315681	844	95.315681
		DSM	684	95.315681	841	95.044128
		Random1	679	94.908348	883	94.908348
		Random2	686	95.315681	873	95.315681
		Random3	685	95.247795	894	95.247795
14	king-rook-vs-king	Original	19241	100	20781	100
		DOM	19243	100	21155	100
		DSM	19238	100	21064	100
		Random1	19247	100	22310	100
		Random2	19260	100	22306	100
		Random3	19252	100	22278	100
Percentage difference%			**6.160704**		**13.30783**	

4.10 Summary

The order of input training examples in an inductive learning algorithm can have a significant impact on the overall performance such as the number of rules, compactness of rules, accuracy and time consumption. For these reasons, a method of ordering the training data set during the rule learning process was developed.

The new method overcomes the problem of generating compact rules when training the data set rather than generating a lower number of rules from the initial example set. This compact rule must always be at the beginning of the training procedure. This data pre-processing DSM comprises three tasks: (i) changing the order of training objects, (ii) changing the order of training attributes, and (iii) changing the order of the training data set after each rule is produced.

The result obtained after applying the new DSM method was superior to that produced by the original inductive learning algorithm of the RULES family.

Chapter 5: Clustering Techniques for Input Data Selection.

5.1 Preliminary

An automatic technique for knowledge acquisition is known as inductive learning. It involves the extraction of knowledge in the form of IF–THEN rules (or an equivalent decision tree). A set of examples is normally required as an input by an inductive learning program. Every example is defined by the values of a number of attributes and the class to which it belongs. A number of inductive learning programs have been developed. Some of the more well-known programs include: ID3 (Quinlan, 1983), a divide-and-conquer program; and AQ (Michalski, 1990), a program based on the simple inductive learning algorithms belonging to the RULES (RULE Extraction System) family developed by Pham and Aksoy (Pham and Aksoy, 1993; Pham and Aksoy, 1995).

A disadvantage of the RULES Family of algorithms is that they generalise the power of the rules formulated during the training process, which then affects the presentation order of the patterns utilised in the process. Random presentation orders of the training data can give forth different outcomes, which may be inaccurate or inefficient.

In other classification methods, such as Simpson's Fuzzy Min-Max Neural Networks (Simpson, 1992), the presentation order of the training patterns has also proved to be a deficiency. To rectify this issue two new training algorithms have been suggested for this type of neural network by Rizzi et al. (2002). Generally, a number of ART-type architectures have been subject to this deficiency (Carpenter et al., 1992).

A data grouping method (DGM) is proposed in this chapter as a means to minimise the variability of the generalisation of the RULES family algorithms owing to different presentation orders of the data in the training process. The DGM is based on a primary clustering stage followed by the ranking of each data point in each cluster. This is achieved by a particular density measure that takes into account the distribution of the patterns in the pattern space, in the same manner as the rule sets created by the RULES family algorithms.

This chapter has been structures as follows:

Section 5.2: The clustering techniques used by the DGM are specified.

Section 5.3: The density measure used to rank the data is presented.

Section 5.4: The Data Grouping Method (DGM) is introduced.

Section 5.5: Experiments are applied.

Section 5.6: Results and discussions of the DGM are presented.

Section 5.7: A summary is given.

5.2 Clustering Techniques

According to Teknomo (2007), clustering is the division of data into groups. or clusters, of similar objects. It was further said that if the data is represented by a small number of clusters then fine detail may be lost but simplification can be attained. Kanugo et al. (2002) said that different clustering techniques can be used to group the data and represent it by its clusters. These techniques have been discussed in detail in Chapter 2 of this thesis. Figure 5.1 details the clustering technique that has been used in this work.

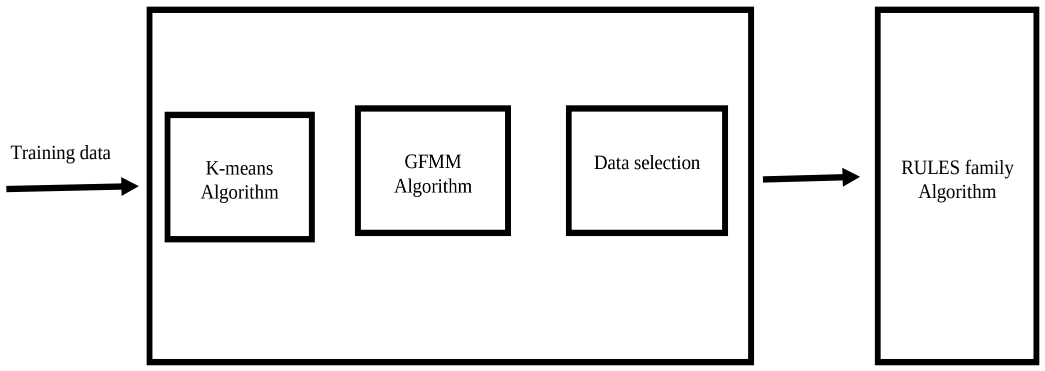


Figure 5.1: Data clustering as a pre-processing method.

2.2.1 K-means clustering Algorithm

The K-means clustering algorithm searches for a way to divide a set of data into sub-sets so that points within a given subset bear a degree of similarity to one another whilst being markedly different from members of other subsets (Teknomo, 2007; Jang, 1997). Such subsets are generally known as clusters.

Kanugo et al. (2002) said that the K-means algorithm is the most common and effective tool for clustering that is used in various scientific applications. According to Wagstaff et al. (2001), the ‘k’ in the k-means algorithm represents the number of clusters in a data set and ‘means’ represents the average location of all the elements of that specific cluster. This algorithm is employed to find patterns similar to a certain cluster, and is regarded as a quick and effective method. Teknomo (2007) said that the square error standard is used by the k-means algorithm for reassessment of a sample from one cluster to another cluster, which further helps in decreasing the total square errors by the users. Kanugo et al. (2002) stated that this algorithm is simple and easy to execute and its results are effective compared to other methods. According to Teknomo (2007) there are several different methods that can be used for calculation of initial points of data. These include: Corner, Bins, Centroid, Spread and PCA. explained the corner method as one in which all the values of a data set are represented by -1 or 1. Teknomo (2007) further said that this method is not effective, as the initial points of data lies on the boundaries of data. According to Teknomo (2007), the bins method covers entire data sets and picks out random points of data from a bit after dividing the data space into bins. Teknomo (2007) explained the centroid method as a method that selects all the starting clusters that are close to the centroid of the data sets. Clusters are calculated by the accumulation of random perturbation into the centroid of the data set that is being clustered. In the spread method the centres of clusters are randomly distributed in order to cover the intact space. According to Wagstaff et al. (2001), PCA is a method that projects data points into the space of principle components. In this method the centre of a cluster is calculated by determining the clusters obtained through one dimensional space.

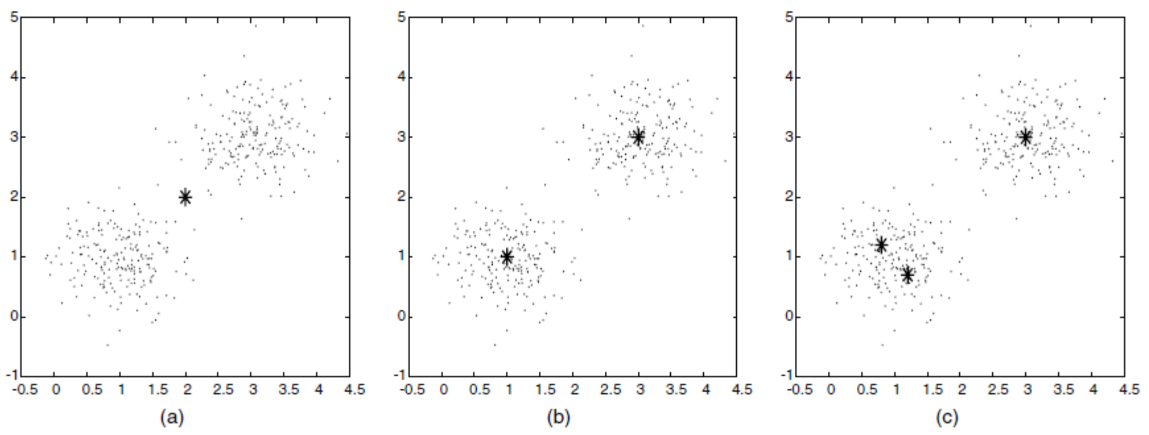


Figure 5.2: the results of the k-means algorithm for a two-cluster data set with (a) k = 1, (b) k = 2, (c) k = 3, where (*) denotes the locations of the converged the seed example (SE) or the cluster centres.

Kanugo et al. (2002) explained that the name of this algorithm represents every single *k* cluster *C* in terms of a mean value, for example a weighted average of its data points, also known as the centroid. Kanugo et al. (2002) further explained that this algorithm has a good sense of numerical attributes both statistically and geometrically. The number of discrepancies between a cluster's centroid and its points can be expressed by the objective function (appropriate distance). The objective function may be based on the L^2 norm. The number of squared errors between centroids and their points is equal to the total variance in intra clusters.

According to Teknomo (2007), the sum of squared errors in a mixture of models can be rationalised by a negative of log likelihood, a technique widely used in statistics. Hence the k-means algorithm can be derived by employing a probabilistic approach. Teknomo (2007) said that by using standard cluster deviation the individual errors can be normalised by simple modification in the clusters. Kanugo et al. (2002) stated that there are many algebraic properties of an L^2 norm which is based on the objective function, as it corresponds to pairwise errors. Kanugo et al. (2002) further explained that the L^2 norm corresponds to a pairwise error with a difference between the inter cluster and the total data variance. Hence the cluster can be separated by the tightness of cluster. The k-mean algorithm can be summarised as shown below:

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

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

Step 3: If the cluster set does not move the algorithm stops. Else go back to step 2.

Figure 5.3: K-means clustering algorithm.

By this process, the K-means algorithm improves its results by optimising the centres of the clusters. The K-means clustering algorithm used in this chapter (Tou and Gonzalez, 1974) is a method for finding K vectors $\mu_j (j = 1,2,\dots\dots,K)$ that represent an entire data set. The data is considered to be partitioned into K clusters, with each cluster represented by its mean vector and each data instance assigned to the cluster with the closest vector.

The K-means algorithm works iteratively. At each stage, the N data examples $A = \{x_1,.,x_N\}$ are partitioned into K disjoint clusters A_j, each containing N_j instances. A cost function (or an objective function) of dissimilarity (or distance) is defined as

$$J = \sum_{j=1}^{K} J_j = \sum_{j=1}^{K} \left(\sum_{x_i \in A_j} \| \; x_i - \mu_j \; \|^2 \right), \qquad (5.1)$$

where μ_j is the centre of the *jth* cluster, given by the mean of the data instances belonging to the cluster:

$$\mu_j = \frac{1}{N_j}\left(\sum_{x_i \in A_j} x_i\right). \tag{5.2}$$

The initial partition of the data is random. The following two steps are then iterated until there is no further change to the cost function, *J*.

1. The mean vector μ_j for each cluster is calculated using equation (5.2).

2. Rearrange the clusters: each data instance x_i is assigned to the *jth* new cluster if x_i is closer to μ_j than to other mean vectors.

5.3 Density Measure

Once the input data is clustered, it is necessary to rank the clusters from the most dense to the least dense. In this manner the RULES algorithm yields more accurate and compact rule sets as it utilises more representative data in its training process. A density measure is required to rank the data with the following expression:

$$D_j = \sum_{i=1}^{M} d_j(x_i), \tag{5.3}$$

Where D_j is the *jth* pattern x_j density value, $x_i\ (i = 1, \dots., M)$ is the *ith* pattern belonging to the same cluster as x_j , and $d_j(\ x_i\ )$ is a distance function (dissimilarity) between x_j and x_i .

When the Euclidean distance is chosen as the dissimilarity measure, the densest pattern for a given cluster will be the pattern that obtains the lowest value for *D*.

Due to the fact that the RULES family algorithms generate IF-THEN rules to classify data, the partitioning of the input data space is achieved by hyperplanes which are parallel to the main reference system. In many cases these hyperplanes intersect to form compact hyperrectangles rule sets. Given that the rules cover the patterns with hyperrectangles it is desirable to measure a pattern's density level amongst the other patterns in the same cluster by also following a rectangular distribution. The Euclidean distance is not used as the dissimilarity function in this case, since the Euclidean distance describes a circular boundary, which is not how the patterns are classified by the RULES family of algorithms.

To be able to measure the similarity between patterns, a fuzzy membership function is proposed. This membership function is a special case of the Fuzzy Hyperbox membership function used in the General Fuzzy Min-Max (GFMM) Neural Network for clustering and classification tasks presented by Gabrys and Bargiela (2000). Following Simpson (1993), let the jth hyperbox fuzzy set, $\mathrm{B_j}$, be defined by the ordered set:

$$\mathrm{B_j} = \{\mathrm{A_h}, \mathrm{V_j}, \mathrm{W_j}, \mathrm{b_j}(\mathrm{A_h}, \mathrm{V_j}, \mathrm{W_j})\}. \qquad (5.4)$$

For all $h = (1, \dots, m)$, where $A_h = (a_{h1}, a_{h2}, \dots, a_{hn}) \in I^n$ (n -dimensional unit cube) is the h^{th} pattern in the data set, $\mathrm{V_j} = \{\mathrm{V_{j1}}, \mathrm{V_{j2}}, \dots, \mathrm{V_{jn}})$ is the minimum point of the jth hyperbox point, $\mathrm{W_j} = \{\mathrm{W_{j1}}, \mathrm{W_{j2}}, \dots, \mathrm{W_{jn}})$, and the membership function for the jth hyperbox, which is $0 \leq \mathrm{b_j}(\mathrm{A_h}, \mathrm{V_j}, \mathrm{W_j}) \leq 1$.

The membership function measures the degree to which the h^{th} input pattern, A_h, falls within the hyperbox formed by the min point, V_j, and the max point, W_j. Gabrys and Bargiela (2000) defined the membership function b_j as:

$$b_j(A_h, V_j, W_j) = min_{i=1.....n}(min(\lfloor 1 - f(a_{hi} - w_{ji}, \gamma)\rfloor, \lfloor 1 - f(v_{ji} - a_{hi,}\gamma\rfloor)), \quad (5.5)$$

Where f is the two-parameter ramp threshold function:

$$f(x,\gamma) = \begin{cases} 1 & if \quad x\gamma > 1 \\ x\gamma & if \quad 0 \leq x\gamma \leq 1 \\ o & if \quad x\gamma < 0 \end{cases} \quad (5.6)$$

The parameter γ is a sensitivity parameter that regulates how fast the membership values decrease when an input pattern is separated from the hyperbox core. When γ is large, the fuzzy set becomes more crisp, and when γ is small the fuzzy set becomes less crisp.

Proposed Fuzzy Membership Function:

Let one consider$V_j = W_j = A_j$, where the jth input pattern, A_j, from Equation 5.5 can be rewritten as:

$$b_j(A_h, A_j,) = min_{i=1.....n}(1 - f(|a_{hi} - a_{ji}|, \gamma)) \quad (5.7)$$

Now, $d_j = b_j$ in Equation 5.5 is a membership function measuring the degree of similarity between two patterns, thus the densest pattern for a given cluster will be the pattern that obtains the highest value for *D*.

A two-dimensional example is shown in Figure 5.4, where it is clear that the membership values decrease steadily with increasing distance from the pattern being analysed, $A_h =$

$[0.5, 0.5]$. The grayscale in Figure 5.4 represents the membership values, where values from 1 to 0 are symbolised by the grey tones ranging from white to black respectively.

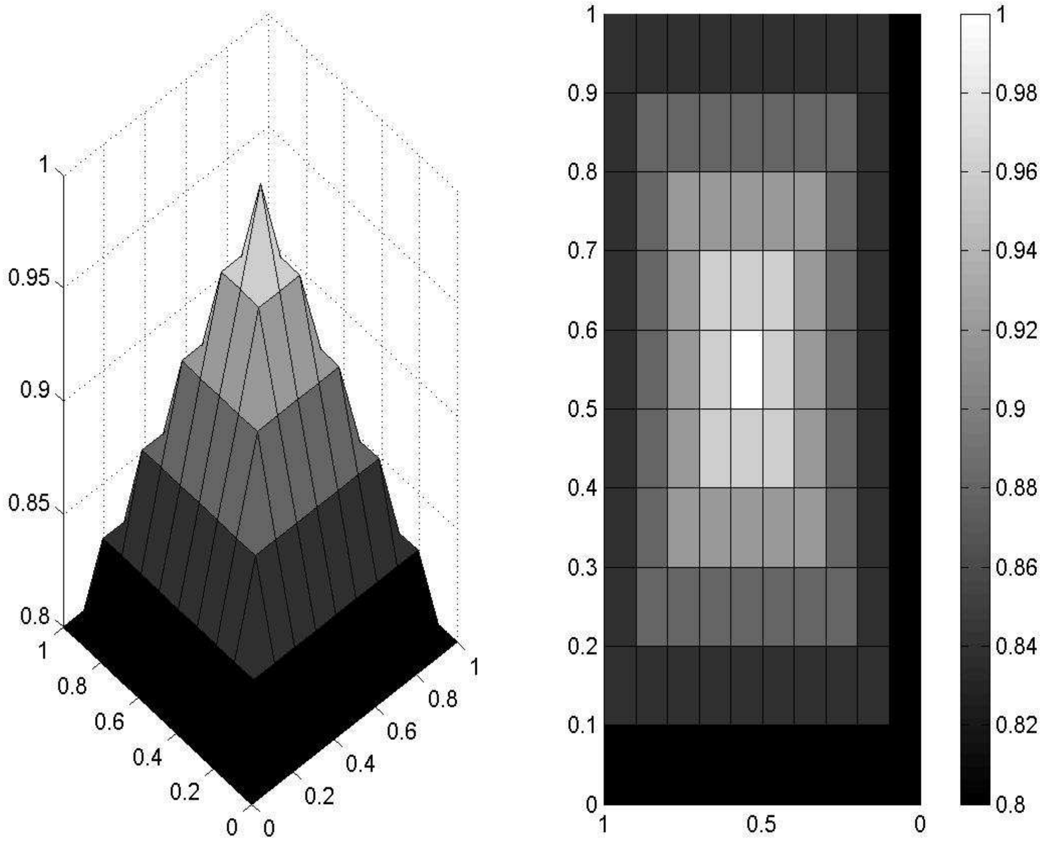


Figure 5.4: 2-dimentional example of membership function used in the density function.

5.4 Data grouping method (DGM)

Figure 5.5 illustrates the full scheme of the data grouping method (DGM) suggested in this chapter. The initial stage is to create clusters with the training data set as shown in Figure 5.1utilising the K-means algorithm. The number of clusters is initially set to the number of classes for a given data set. This number is acquired by setting a suitable value of K in the K-means algorithm. However, as will be shown later in section 5.6 Results and Discussions, it is more efficient to examine a higher number of clusters than the number of classes. Once the clusters are generated, it is imperative to rank each data point from the most dense to the least dense in each cluster. To achieve this, the density measure in Equation 5.3 is deployed with the suggested fuzzy membership function of Equation 5.7. The following phase is the data selection stage where one data point is selected from each cluster (taking into account its ranking). In the end an ordered data set is acquired and used for the training process in one of the RULES family algorithms. To ascertain the RULES family generalisation efficiency, a separate data set is used. This is known as the test set, and has been kept independent from the training process to enable measurement of the algorithm's classification accuracy percentage.

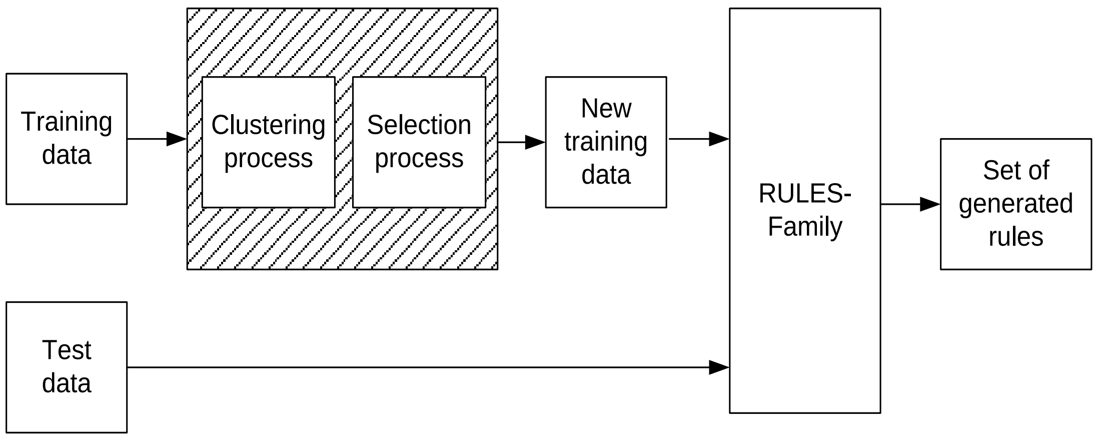


Figure 5.5: The proposed enhancement method for the RULES family algorithm.

5.5 Experiments

To measure how sensitive the rule generation process (training stage) is to different orders of pattern presentations, 30 randomly ordered presentations of the training data sets were used in the RULES family algorithms and compared to the performance obtained by the DGM. The RULES family algorithms that were used in this experiment were RULES 3 Plus (Pham and Dimov, 1997), RULES-5 (Pham, Bigot and Dimov, 2003) and RULES-5 Plus (Pham, Bigot and Dimov, 2004).

The DGM proposed in this work was tested using 10 benchmark data sets (breast cancer, haberman, ionosphere, iris, wine, glass, liver, zoo, pima indian's diabetes, tae) from the UCI Machine Learning Repository (Blake and Merz, 1998). A list of these data sets describing the number of attributes, classes and partitioning of the training and test sets is shown in Table 5.1.

Table 5.1: Description of data sets used to train and test the DGM

No	Data set	Attributes	Classes	Training set	Test set	Total
1	Breast Cancer	30	02	313	256	569
2	Haberman	03	02	169	137	306
3	Ionosphere	34	02	193	158	351
4	Iris	4	03	80	70	150
5	Wine	13	03	98	80	178
6	Glass	09	06	118	96	214
7	Liver	06	02	190	155	345
8	Zoo	16	07	56	45	101
9	Pima	08	02	422	346	768
10	Tae	05	03	84	67	151

5.6 Results and Discussions

The results of the experiments carried out on the data sets are shown in Tables 5.2 and 5.3 for the RULES-3 Plus algorithm, Tables 5.4 and 5.5 for the RULES-5 algorithm, and Tables 5.6 and 5.7 for the RULES-5 Plus algorithm. For each of the RULES family algorithms used, two tables presenting the DGM performance versus the performance of 30 random presentation orders are compared. For example, for RULES-3 Plus, Table 5.2 shows on the left side the worst classification percentage, the best classification percentage, and the average classification percentage using the test set for 30 random presentation orders of the training data. On the right side of the table is the classification performance obtained when the DGM was used on the training patterns, results using K-means (DGMK-means) clustering, as well as the number of clusters formed by K-means clustering technique. Table 5.3 shows the minimum, maximum, and most frequent rule that was obtained by the 30 random presentation orders of the training sets, as well as the number of rules generated by ordered data using the DGM.

Table 5.2 shows that for RULES-3 Plus, the performance obtained by DGMK-means algorithms is higher than the average value of the classification percentage of 30 random presentation orders of the training set. In particular, it can be seen that for the haberman, ionosphere, iris, wine, liver, zoo and tae data sets, the DGMK-means results obtain the same value as the best case found among the random presentation orders. As see in Figure B1: Classification performance in the RULES-3 Plus with random and order presentation.

Analysing Table 5.4 reveals that once again the DGMK-means, in this case of RULES-5, obtained a higher performance than the average value of the 30 random presentation orders of the training sets (and equal performance in the zoo and the iris data sets which did not present any variations). It is also of note that the DGMK-means with the breast cancer, haberman, ionosphere, wine, glass, liver, pima Indians and the tae data set, a performance higher was obtained than for the best case found by the 30 random presentation orders. As see in Figure B4: Classification performance in the RULES-5 with random and order presentation of the data

From Tables 5.2, 5.4 and 5.6, we see that the number of clusters used by the DGM is higher than the number of classes for each data set analysed. The reason for this is that better results were obtained by considering a higher number of clusters than the number of classes in each case. Not only in the classification performance does the random presentation of the training pattern generate significant variations in the results, but it is also clear from Tables 5.3, 5.5 and 5.7 that the number of rules is strongly affected as well.

In general terms, by studying all the tables, it was found that the random presentation order of the training patterns generates a variation in the classification percentage of approximately 6 % in the test set and approximately 18 % in the number of rules formed.

Table 5.2: Classification performance in the RULES-3 Plus with random and ordered presentation.

No	Data set	Worst case %	Best case %	Average %	K	DGMk-means %
1	BreastCancer	93.75	95.31	94.66	3	94.92
2	Haberman	48.91	75.18	63.33	3	75.18
3	Ionosphere	87.97	91.14	89.75	3	91.18
4	Iris	94.29	95.71	95.57	4	95.71
5	Wine	91.25	96.25	95.00	3	96.75
6	Glass	56.25	59.38	57.92	12	60.42
7	Liver	58.06	60.65	59.23	7	60.60
8	Zoo	86.67	86.67	86.67	7	86.67
9	Pima	71.68	73.41	72.60	7	73.12
10	Tae	46.27	47.76	46.87	3	47.76

Table 5.3: Number of rules formed by RULES-3 Plus with random and ordered presentation of the data.

No	Data set	Min-rules	Max-rules	Frequent -rules	DGMK-means -rules
1	BreastCancer	36	44	42	43
2	Haberman	43	43	43	43
3	Ionosphere	41	48	45	41
4	Iris	11	13	12	11
5	Wine	22	27	24	25
6	Glass	51	54	53	52
7	Liver	80	84	84	80
8	Zoo	7	9	8	8
9	Pima	183	193	186	185
10	Tae	33	34	33	33

Table 5.4: Classification performance of the RULES-5 algorithm with random and ordered presentation of the data.

No	Data Set	Worst case %	Best case %	Average %	K	DGMk-means %
1	BreastCancer	92.58	96.48	94.60	6	97.27
2	Haberman	58.39	75.18	71.82	4	74.45
3	Ionosphere	89.87	93.67	91.65	5	94.30
4	Iris	95.71	95.71	95.71	3	95.71
5	Wine	87.45	97.50	92.54	9	98.75
6	Glass	38.54	61.48	46.98	11	50.00
7	Liver	63.87	72.26	67.10	4	70.32
8	Zoo	88.89	88.89	88.89	7	88.89
9	Pima	73.12	77.17	74.94	4	77.75
10	Tae	38.80	47.76	43.43	3	46.27

Table 5.5: Number of rules formed by RULES-5 with random and order presentation of the data.

No	Data Set	Min-rules	Max-rules	Frequent-rules	DGMK-means -rules
1	BreastCancer	12	20	19	17
2	Haberman	45	58	50	50
3	Ionosphere	17	23	18	18
4	Iris	6	10	10	7
5	Wine	7	14	10	9
6	Glass	29	35	33	35
7	Liver	40	55	48	46
8	Zoo	7	9	8	9
9	Pima	87	120	98	86
10	Tae	24	34	24	23

Table 5.6: Classification performances in the RULES-5 Plus with random and ordered presentation of the data.

No	Data set	Worst case%	Best case%	Average %	K	DGMk-means %
1	BreastCancer	91.80	96.09	94.13	4	94.53
2	Haberman	60.58	74.45	71.33	3	73.72
3	Ionosphere	87.34	94.94	92.03	5	94.30
4	Iris	94.29	95.71	95.33	5	95.71
5	Wine	86.25	97.50	92.54	9	98.75
6	Glass	34.38	56.25	45.10	12	55.21
7	Liver	64.52	72.26	67.48	7	70.32
8	Zoo	88.89	88.89	88.89	7	88.89
9	Pima	74.57	79.48	76.76	4	79.19
10	Tae	40.30	74.76	42.84	3	46.27

Table 5.7: Number of rules formed by RULES-5 Plus with random and ordered presentation of the data.

No	Data Set	min-rules	max-rules	frequent -rules	DGMK-means -rules
1	BreastCancer	3	14	7	10
2	Haberman	24	37	32	28
3	Ionosphere	8	18	9	11
4	Iris	3	6	5	4
5	Wine	6	10	7	7
6	Glass	24	32	28	30
7	Liver	38	47	43	40
8	Zoo	7	8	8	8
9	Pima	52	61	55	61
10	Tae	22	24	24	22

5.7 Summary

This chapter has presented a method for the selection of optimum examples for representing a data set. Clustering algorithm is presented in this chapter and one technique in particular, based on neural network clustering, were described. This technique groups together similar examples prior to selecting examples from each cluster. The DGM was tested with ten data sets where the performance was greater than the average classification percentage achieved by thirty random presentation orders of the training set.

Chapter 6: Conclusions and Future Work

During the initial phase of this study, a review was conducted covering the basic topics in machine learning, data mining and inductive learning algorithms. The review identified a number of areas that required further investigation and prompted the research presented in this thesis. This chapter ends the thesis by summarising the main conclusions of the study and proposing areas that require further exploration.

6.1 Conclusions

This section will re-examine the objectives stated in Chapter 1 to show that they have all been met.

The main reason for developing the data ordering methods presented in this thesis is to reduce the variation in the generalisation performance that can be caused by ordering the training pattern presentations differently. This work has focused on the pre-processing of data, and in particular the pre-processing steps used in the RULES family of learning algorithms. The data ordering method (DOM) works by sorting the examples before the rule is introduced into the inductive learning algorithm. It gives priority to the example with the highest information content. The method consists of three main steps: (i) the entropy for each attribute value is calculated; (ii) the entropy value for each example is calculated by adding all the entropy values of the associated attributes; (iii) examples are reordered according to their entropies. In general, the results obtained by the data ordering method gave smaller rule sets compared to the original ordering method and three random ordering methods **(Objective 1)**. The proposed technique succeeded in improving the performance of reducing the number of rules created by the three algorithms. However in terms of the processing time, an additional computational effort was required making the method slightly more time consuming.

The fundamental rationale for the Data Sorting Method is the fact that a possible cause of variation is the significance of outlier data and/or data with only a minimum number of neighbours in the pattern's input space. This occurs when the data is presented in the initial phase of the rule forming process, instead of in the final phase, causing the inductive algorithm to generate less compact and less accurate rule sets. For these reasons, a method of ordering the training data set during the rule learning process was developed; this data pre-processing DSM comprises three tasks: (i) changing the order of training objects, (ii) changing the order of training attributes, and (iii) changing the order of the training data set after each rule is produced **(Objective 2)**. The performance of the proposed algorithm was better than all other orders. The ordering method was more efficient in terms of computational effort and the training time was also reduced significantly.

This work draws attention to two significant points that are deemed to be the most effective at measuring the quality of the algorithm. These two points include improvements in the number of rules created and in the classification accuracy. These terms have been used by Pham and Dimov (1997b) to show the quality of the RULES family algorithms.

Clustering was introduced in order to minimise variability in the generalisation, and the number of rules generated by the inductive learning algorithms. From the RULES family, a new data grouping method known as DGM has been proposed. The DGM is comprised of three main steps. The initial step entails clustering with the training patterns. In this case the clustering technique known as the K-means algorithm and was utilised. The second step ranks each data point from each cluster based on its level of importance, such as its density value. At this stage a new density function has been suggested which considers the rectangular distribution in which the algorithms in the RULES family classify patterns. The third step includes the selection of data from each cluster. The pattern with the maximum density value is selected in each iteration until all patterns are exhausted. The RULES-3 Plus and RULES-5 Plus algorithms from the RULES family were utilised **(Objective 3)**. This method gave good performance compared to the case which had no pre-processing. In a number of cases the performance was greater than the average classification percentage achieved by thirty random presentation orders of the training set. In other cases, the DGM outperformed the best classification percentage found in the 30 random presentation orders, and in some cases it equalled the best case.

6.2 Future Research Directions

As illustrated in Chapter 2, data mining is more significant than simply applying data modelling algorithms to data. It is a crucial step in getting the data ready by using various data pre-processing techniques. These include feature selection and clustering techniques. Unfortunately, a lack of integrated tools and methodologies to perform these tasks still exists. It is therefore important that data pre-processing techniques exploit the capabilities of the underlying database management system, data warehouse and data engineering.

- In Chapter 3 and Chapter 4, only the size of the models (i.e. the number of rules extracted) is used as a performance indicator. Future work will include validation of the created models using a separate test data set.

- In Chapter 3 and Chapter 4, missing and numeric (as opposed to nominal) attributes are not handled. Applying the proposed data ordering method to domains that involve missing and numeric attribute values will form future work.

- In Chapter 3 and Chapter 4, feature selection draws particular attention to the selection of beneficial features from a set of original available features. Real data mining applications require the construction of new features from available features. Currently, there are no automatic or even semi-automatic tools for carrying out this transformation. To address this issue, tools from the fields of statistics and machine learning should be combined with data visualisation and domain knowledge.

- The discretisation procedure deployed in the RULES family of algorithms is characterised by extreme simplicity, and as such the procedure is not applicable to all values. New discretisation procedures are required to speed up the learning process by permitting various quantisation levels for distinct numerical attributes incorporating adaptive discretisation.

- The current version of the RULES family algorithm, referred to as Rule Pruning, is easy to use but it limits the number of attributes and examples that can be utilised. The latest version of RULES family, which is able to deal with any kind of data without restrictions, should substantially maximise the rule-generating facility. Moreover, working solely with nominal values unambiguously restricts the application for real world data. The method should therefore be stretched so that it can deal with both numerical attributes and nominal values.

- The measure deployed in the RULES family is complicated and does not yield high levels of accuracy. This measure plays a significant role in rules, and as such it is essential that a measure is developed which is simple, easy to comprehend and precisely associated with rule accuracy.

- Future research will analyse in more detail the ideal number of clusters deployed by the clustering phase of the DGM, together with an exploration of other data selection procedures once the data in each cluster is ranked. Moreover, the application of this method should be considered in other classification systems which are sensitive to the presentation order of the training patterns.

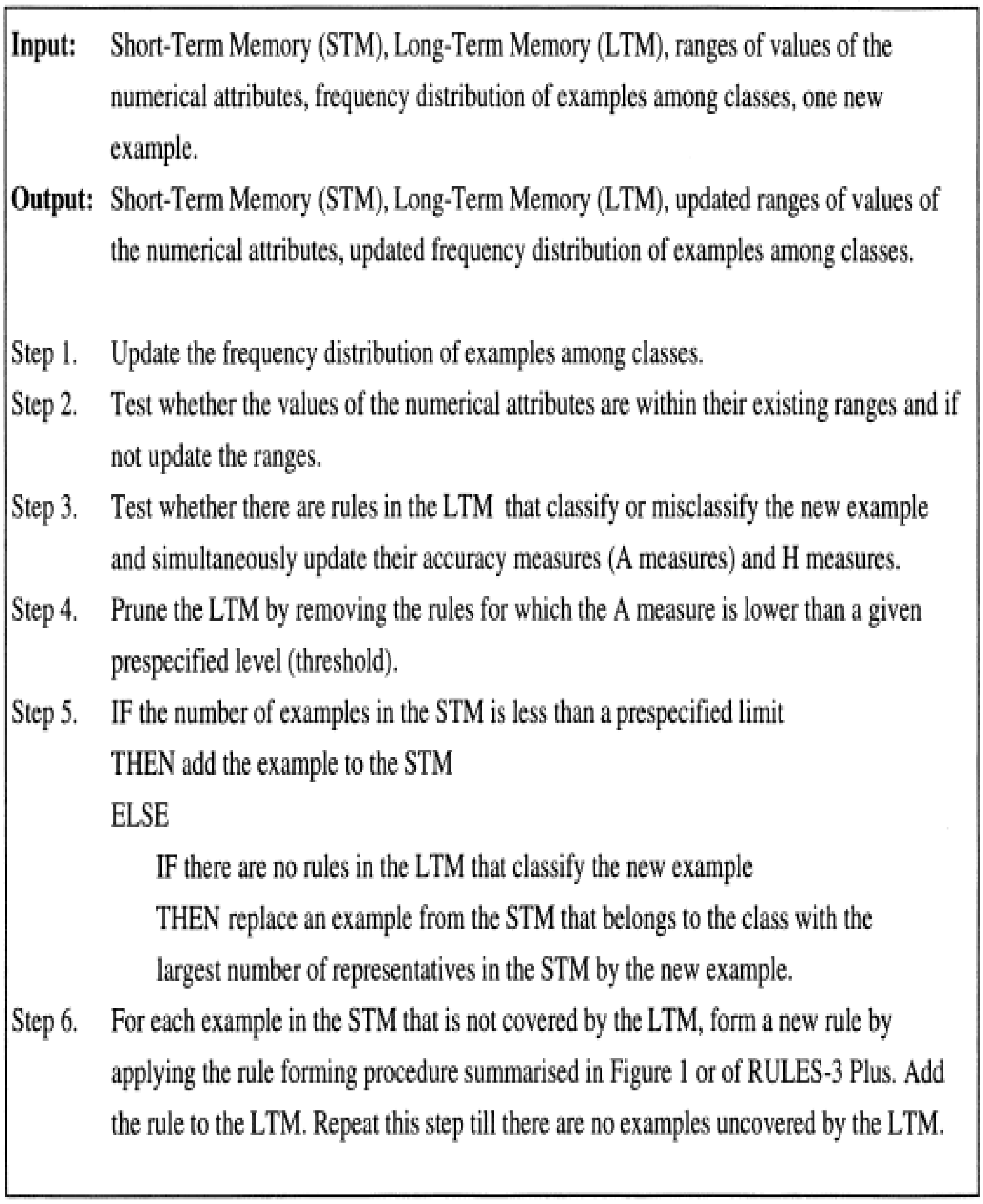
- The methods must be extended for handling nominal attribute values as well as numerical attribute values.

- The clustering technique used is unable to deal with the difficulty of unknown attribute values frequently encountered in the real world. A more suitable clustering technique capable of handling unknown attributes is needed.

- In Chapter 5, cross-validation could be used instead of the random splitting of examples between training and testing when running experiments.

APPENDIX A

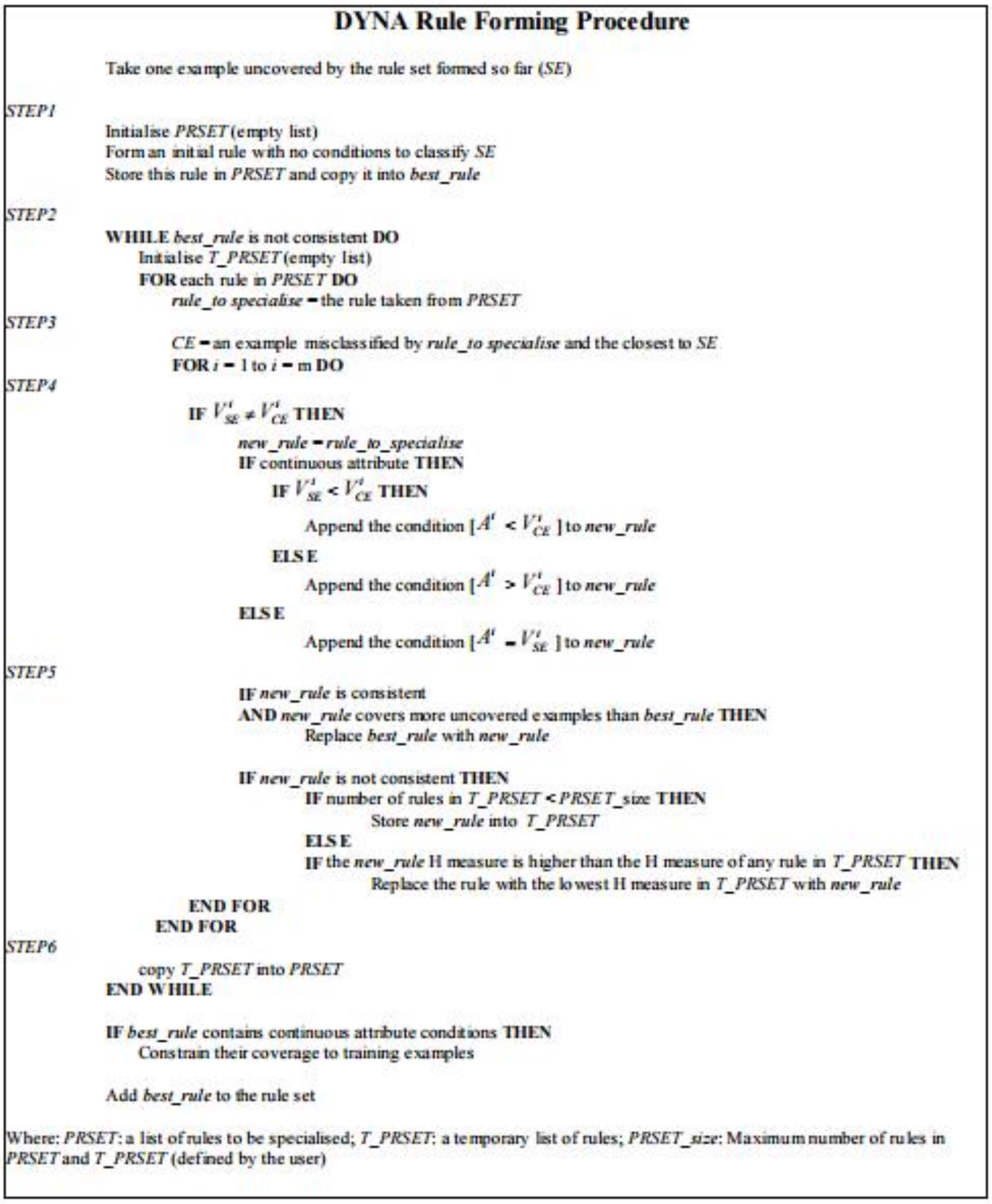
A PSEUDO-CODE OF THE RULES FAMILY ALGORITHM



```
Input:   Short-Term Memory (STM), Long-Term Memory (LTM), ranges of values of the
         numerical attributes, frequency distribution of examples among classes, one new
         example.
Output:  Short-Term Memory (STM), Long-Term Memory (LTM), updated ranges of values of
         the numerical attributes, updated frequency distribution of examples among classes.

Step 1.  Update the frequency distribution of examples among classes.
Step 2.  Test whether the values of the numerical attributes are within their existing ranges and if
         not update the ranges.
Step 3.  Test whether there are rules in the LTM that classify or misclassify the new example
         and simultaneously update their accuracy measures (A measures) and H measures.
Step 4.  Prune the LTM by removing the rules for which the A measure is lower than a given
         prespecified level (threshold).
Step 5.  IF the number of examples in the STM is less than a prespecified limit
         THEN add the example to the STM
         ELSE
             IF there are no rules in the LTM that classify the new example
             THEN replace an example from the STM that belongs to the class with the
             largest number of representatives in the STM by the new example.
Step 6.  For each example in the STM that is not covered by the LTM, form a new rule by
         applying the rule forming procedure summarised in Figure 1 or of RULES-3 Plus. Add
         the rule to the LTM. Repeat this step till there are no examples uncovered by the LTM.
```

Figure A1: Incremental induction procedure in RULES-4, Pham and Dimov (1997a).



DYNA Rule Forming Procedure

```
        Take one example uncovered by the rule set formed so far (SE)

STEP1
        Initialise PRSET (empty list)
        Form an initial rule with no conditions to classify SE
        Store this rule in PRSET and copy it into best_rule

STEP2
        WHILE best_rule is not consistent DO
            Initialise T_PRSET (empty list)
            FOR each rule in PRSET DO
                rule_to specialise = the rule taken from PRSET
STEP3
                CE = an example misclassified by rule_to specialise and the closest to SE
                FOR $i = 1$ to $i = m$ DO
STEP4
                    IF $V^i_{SE} \neq V^i_{CE}$ THEN
                        new_rule = rule_to_specialise
                        IF continuous attribute THEN
                            IF $V^i_{SE} < V^i_{CE}$ THEN
                                Append the condition [$A^i < V^i_{CE}$] to new_rule
                            ELSE
                                Append the condition [$A^i > V^i_{CE}$] to new_rule
                        ELSE
                                Append the condition [$A^i = V^i_{SE}$] to new_rule
STEP5
                        IF new_rule is consistent
                        AND new_rule covers more uncovered examples than best_rule THEN
                                Replace best_rule with new_rule

                        IF new_rule is not consistent THEN
                                IF number of rules in T_PRSET < PRSET_size THEN
                                        Store new_rule into T_PRSET
                                ELSE
                                IF the new_rule H measure is higher than the H measure of any rule in T_PRSET THEN
                                        Replace the rule with the lowest H measure in T_PRSET with new_rule
                    END FOR
                END FOR
STEP6
            copy T_PRSET into PRSET
        END WHILE

        IF best_rule contains continuous attribute conditions THEN
            Constrain their coverage to training examples

        Add best_rule to the rule set
```

Where: *PRSET*: a list of rules to be specialised; *T_PRSET*: a temporary list of rules; *PRSET_size*: Maximum number of rules in *PRSET* and *T_PRSET* (defined by the user)

Figure A2: RULES-5 Plus forming procedure (Bigot 2002).

APPENDIX B

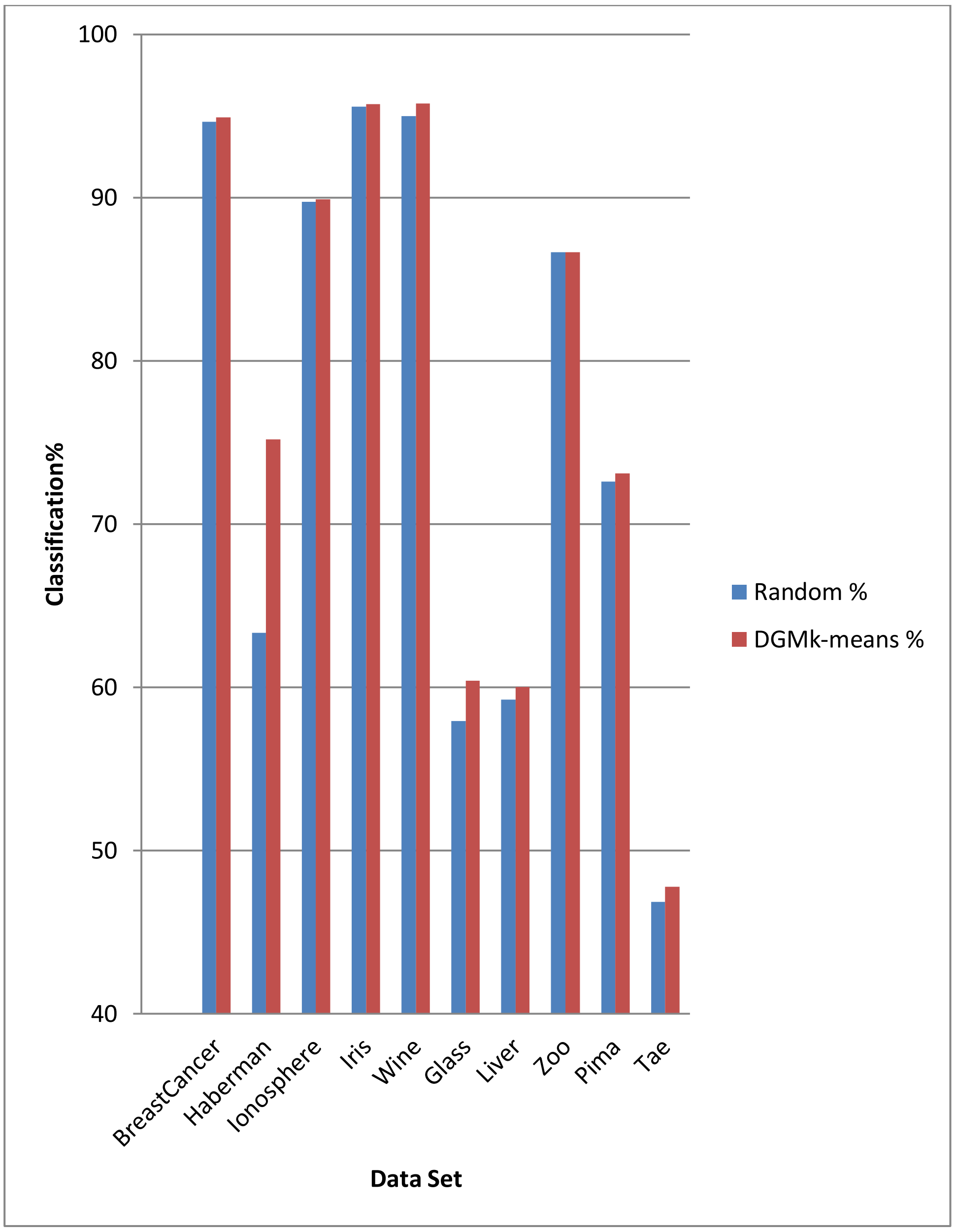


Figure B1: Classification performance in the RULES-3 Plus with random and order presentation.

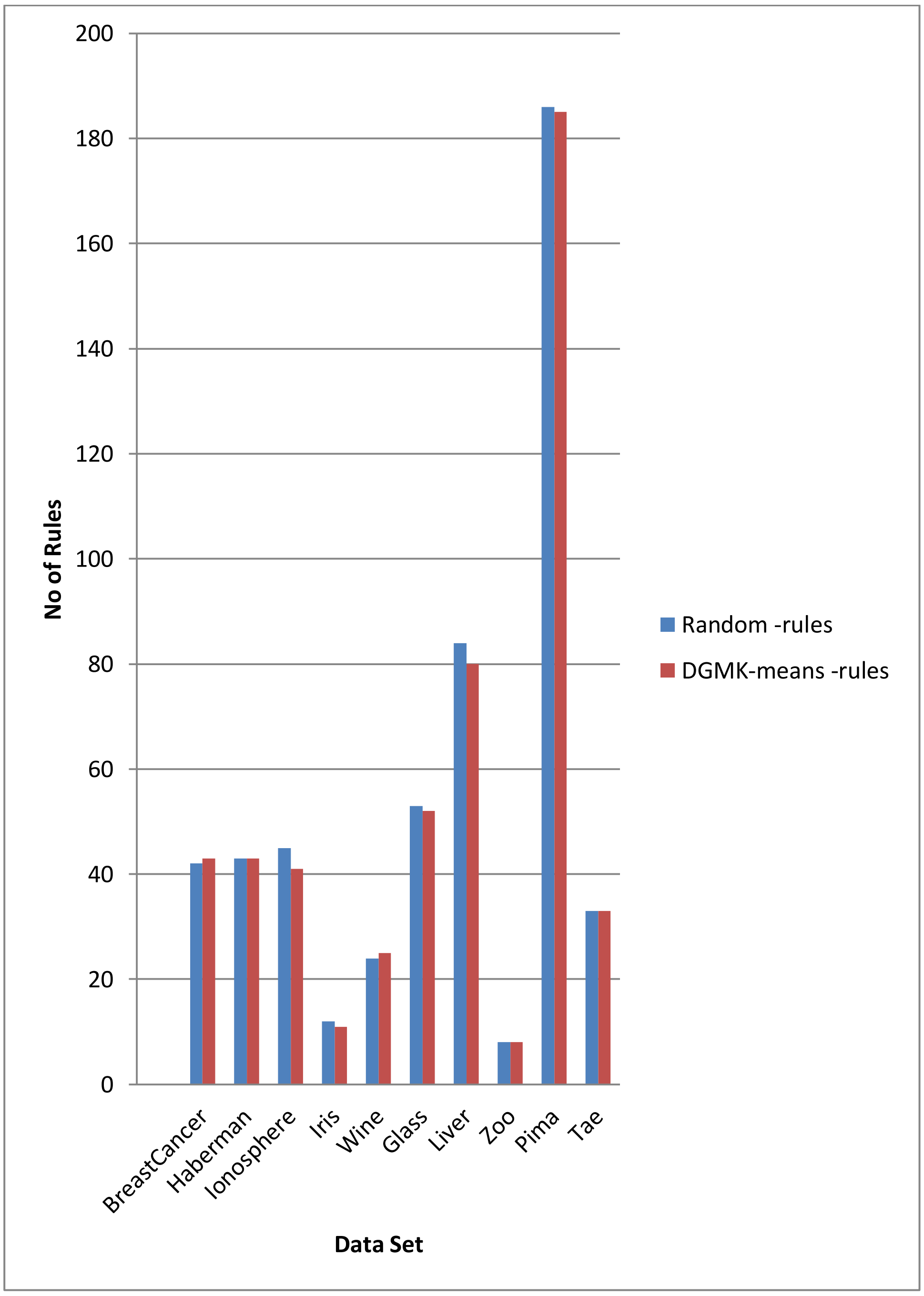


Figure B2: Number of rules formed by RULES-3 Plus with random and order presentation of the data.

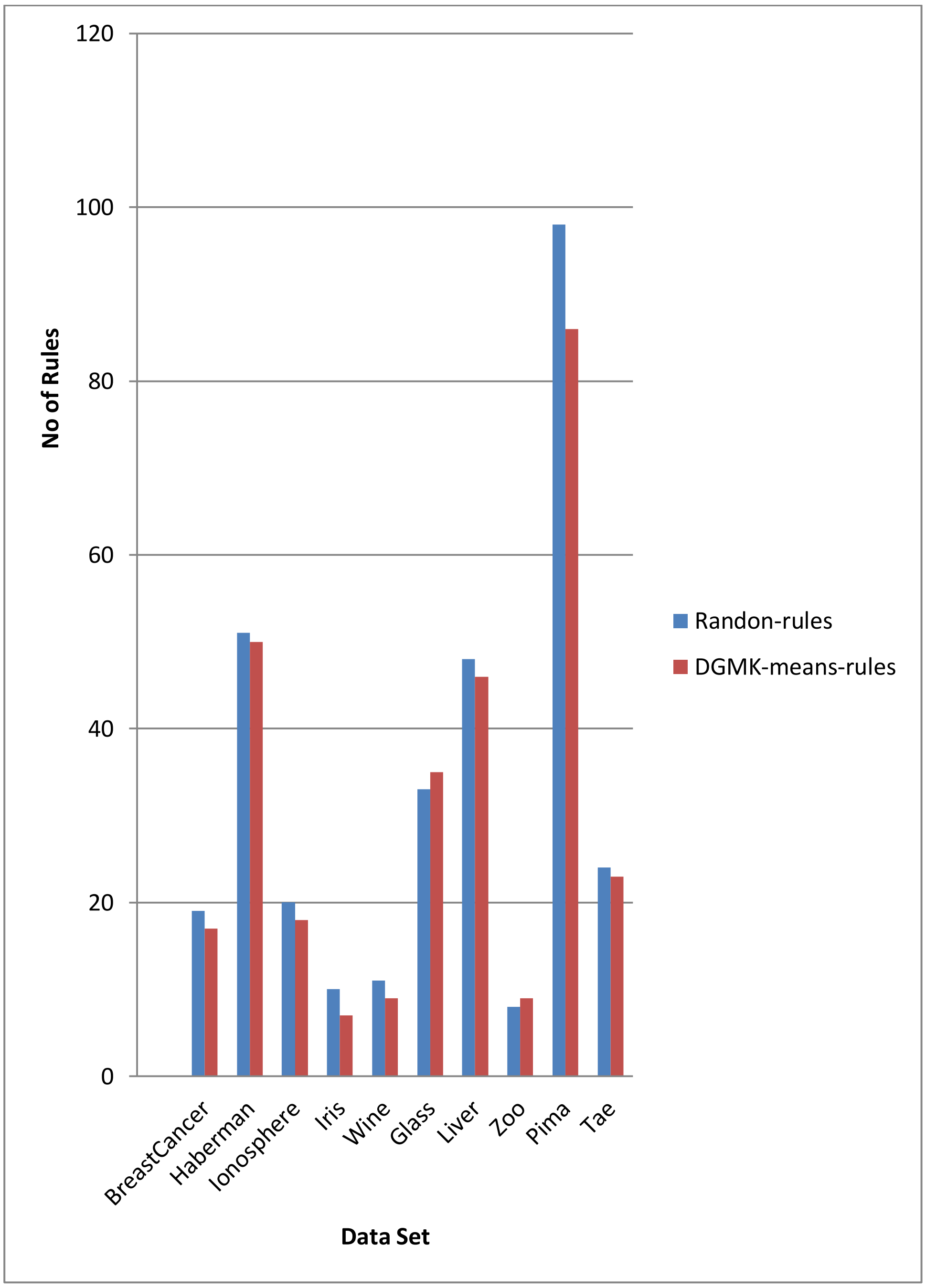


Figure B3: Number of rules formed by RULES-5 with random and order presentation of the data.

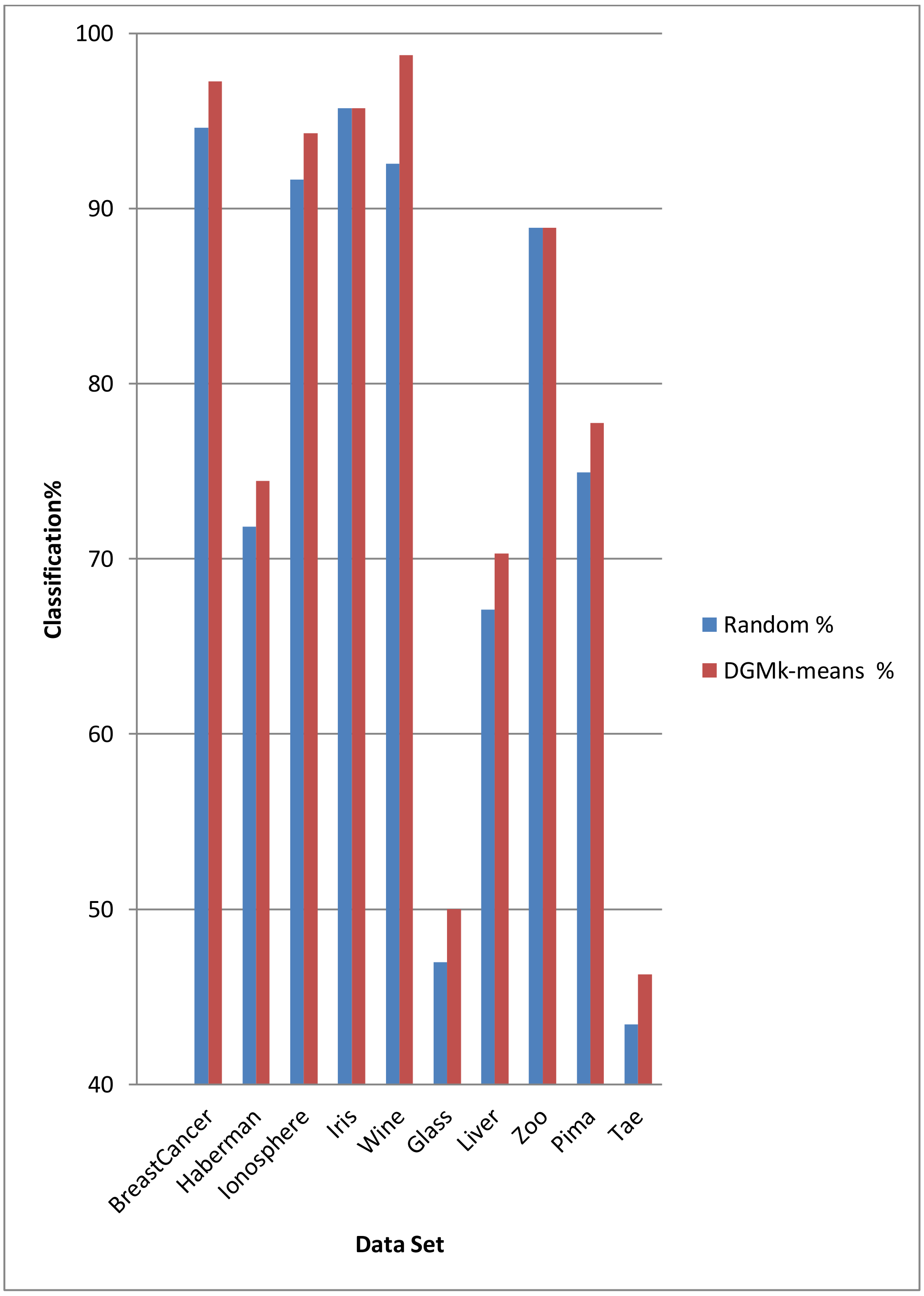


Figure B4: Classification performance in the RULES-5 with random and order presentation of the data.

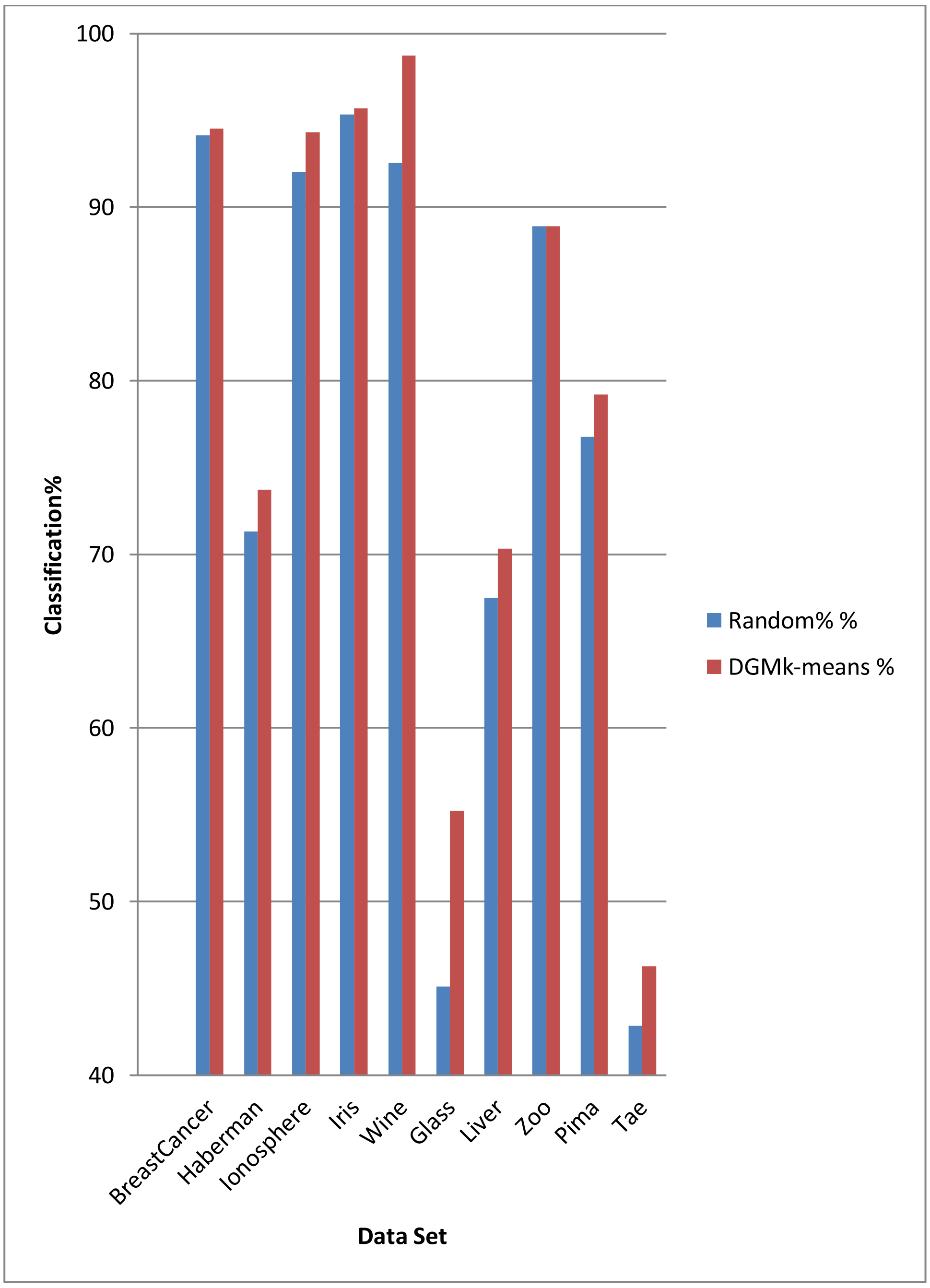


Figure B5: Classification performances in the RULES-5 Plus with random and order presentation of the data.

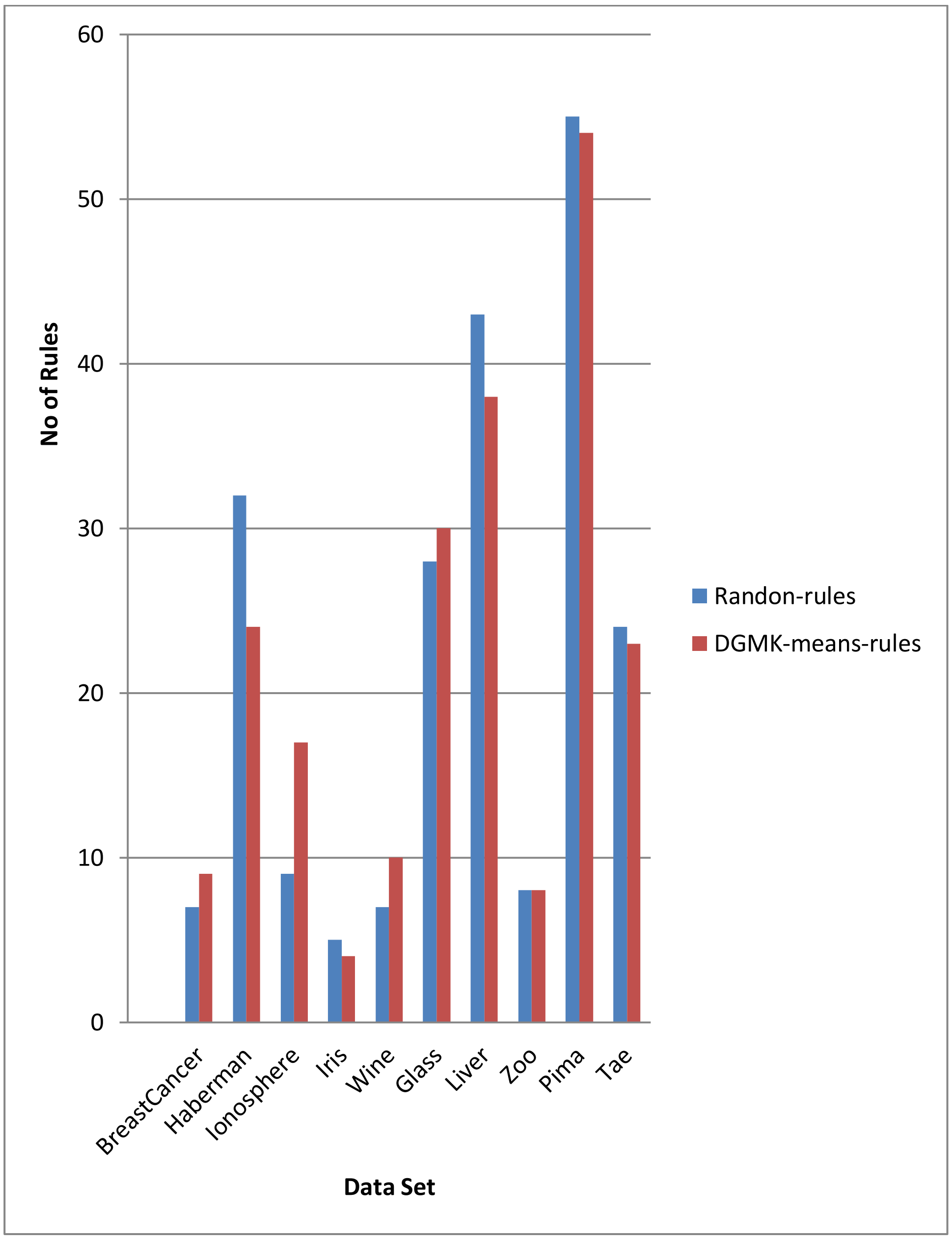


Figure B6: Number of rules formed by RULES-5 Plus with random and order presentation of the data.

References

Afify, A. A. (2004) Design and Analysis of Scalable Rule Induction Systems. *PhD Thesis*, Systems Engineering Division, University of Wales, Cardiff, UK.

Agrawal, R., Imielinski, T. and Swami, A. (1993) Mining Association Rules between Sets of Items in Large Databases. *Proc. of ACM SIGMOD Conf.*, Washington D.C., pp. 207-216.

Agrawal, R. and Srikant, R. (1994) Fast Algorithms for Mining Association Rules. *Proc. of the 20th VLDB Conf.*, Santiago, Chile, pp. 487-499.

Aha, D. W., Kibler, D. and Albert, M. K. (1991) Instance-Based Learning Algorithms. *Machine Learning*, Vol. 6, pp. 37-66.

Aksoy, M. S. (2008) A Review of the Rules Family of Algorithms. *Mathematical and Computational Applications*, Vol. 13, pp. 51-60.

Aksoy, M. S. (2005) Applications of RULES-3 Induction System. *IPROMS2005 virtual conference*, Cardiff, UK.

Alexander, D. (2008) Data mining. Available online at: *http://www.laits.utexas.edu/~anorman/BUS.FOR/course.mat/Alex/* [Accessed on 24/04/2009].

An, A. and Cercone, N. (1999) Discretization of Continuous Attributes for Learning Classification Rules. *Proc. of the 3rd Pacific-Asia Conf. on Methodologies for Knowledge Discovery and Data Mining*, pp. 509-514.

Ansari, S., Chetlur, S., Prabhu, S., Kini, N. G., Hedge, G. and Hyder, Y. (2013) An Overview of Clustering Analysis Techniques used in Data Mining. *International Journal of Emerging Technology and Advanced Engineering.* vol. 3, no. 12, pp. 284–286

Aronis, J. and Provost, F. (1997) Increasing the Efficiency of Data Mining Algorithms with Breadth-first Marker Propagation. *Proc. of the 3rd Int. Conf. on Knowledge Discovery and Data Mining*, Newport Beach, CA. pp. 119–122.

Axelson, D. E. (2012) Data Preprocessing for Chemometric and Metabonomic Analysis. *Createspace Independent Publishing Platform*.

Barber, D. (2012) *Bayesian Reasoning and Machine Learning*. Cambridge University Press.

Bay, S. D. (2000) Multivariate Discretization of Continuous Variables for Set Mining. *Proc. of the 6th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, Boston, Massachusetts, United States, pp. 315-319.

Bayardo, R. J. (1998) Efficiently Mining Long Patterns from Databases. *Proc. of the 1998 ACM SIGMOD Int. Conf. on Management of Data*, Seattle, Washington, United States, pp. 85-93.

Berry, M. J. A. and Linoff, G. S. (2004) *Data Mining Techniques: For Marketing, Sales, and Customer Relationship Management*. New York: Wiley Computer Publishing.

Bigot, S. (2002) New Techniques for Handling Continuous Values in Inductive Learning. *PhD Thesis*, Systems Engineering Division, University of Wales, Cardiff, UK.

Bishop, C. M. (2006) *Pattern Recognition and Machine Learning, a Matlab Companion*. Springer.

Blake, C. L. and Merz, C. J. (1998) *UCI Repository of Machine Learning Databases*. Irvine, University of California, USA. Available online at: http://www.ics.uci.edu/~mlearn/MLRepository.html [Accessed on 28/05/2009].

Barber, D. (2012) *Bayesian Reasoning and Machine Learning*. Cambridge University Press.

Bell, J. (2014) *Machine Learning: Hands-On for Developers and Technical Professionals*. John Wiley & Sons.

Braha, D. (2001) *Data Mining for Design and Manufacturing: Methods and Applications*. Kluwer Academic Publishers, Boston.

Bramer, M. (2007) *Principles of Data Mining (Undergraduate Topics in Computer Science).* Springer.

Burdick, D., Calimlim, M. and Gehrke, J. (2001) MAFIA: A Maximal Frequent Itemset Algorithm for Transactional Databases. *Proc. of 17th Int. Conf. on Data Engineering*, pp. 443-452.

Brownlee, J. (2013) *A Tour of Machine Learning Algorithms*. Available online at: http://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/ [Accessed on 19/08/2015].

Busse, G. (2005) *Rule Induction*. Available online at: http://link.springer.com/chapter/10.1007%2F0-387-25465-X_13 [Accessed on 22/08/2015].

Calderon, T. G., Cheh, J. J. and Kim, I. (2003) How large corporations use data mining to create value. *Management Accounting Quarterly (winter)*, pp. 1-11.

Cao, F., Ester, M., Qian, W. and Zhou, A. (2006) *Density-Based Clustering over an Evolving Data Stream with Noise*. Available online at: http://www.siam.org/meetings/sdm06/proceedings/030caof.pdf [Accessed on 15/04/2014].

Carpenter, G. A., Grossberg, S., Markuzon, N., Reynolds, J. H. and Rosen, D. B. (1992) Fuzzy ARTMAP: A Neural Network Architecture for Incremental Supervised Learning of Analogy Multidimensional maps. *IEEE Trans. Neural Networks*, Vol. 3, No. 5, pp. 698-713.

Carter, C. and Catlett, J. (1987) Assessing Credit Card Applications Using Machine Learning. *IEEE Expert Intelligent Systems and Their Applications*, FALL, pp. 71-79.

Cervone, G., Franzese, P. and Allen, P. K. K. (2010) Algorithm Quasi-optimal (AQ) Learning. Available online at: file:///C:/Users/Khizer/Desktop/2010_WIRES_AQLearning_Cervone.pdf [Accessed on 28/11/2012].

Cervone, G., Panait, L. A. and Michalski, R. S. (2001) The Development of the AQ20 Learning System and Initial Experiments. *Proc. of the 10th Int. Symposium on Intelligent Information Systems*, Zakopone, Poland.

Chan, Y., Talburt, J. and Talley, T. M. (2009) *Data Engineering: Mining, Information and Intelligence*. Springer Science and Business Media.

Chang, C., Lin, N. P. and Jan, N. Y. (2009) An Axis-Shifted Grid-Clustering Algorithm. *Tamkang Journal of Science and Engineering*, Vol. 12, No 2, pp. 183−192.

Chen, W., Chen, Y., Guo, B. (2013) *Density-Based Logistic Regression*. Available online at: http://www.cse.wustl.edu/~ychen/public/kdd13.pdf [Accessed on 15/11/2013].

Datta, A. Pal S, (2000) A Connectionist Model for Convex-Hull of Planar. *Set Neural Networks* Vol. 13, pp. 377-884

Dale, M. B., McBratney, A. B., and Russell, J. S. (1989) On the Role of Expert Systems and Numerical Taxonomy in Soil Classification. *Journal of Soil Science*, Vol. 40, pp. 223-234.

Flach, P. (2012) *Machine Learning: The Art and Science of Algorithms that Make Sense of Data*. Cambridge University Press.

Ferreira, L. and Hitchcock, D. B. (2011) *A Comparison of Hierarchical Methods for Clustering Functional Data*. Available online at http://www.stat.sc.edu/~hitchcock/compare_hier_fda.pdf [Accessed on 15/04/2014].

Freitas, A. A. (2002) *Data Mining and Knowledge Discovery with Evolutionary Algorithms*. Springer Science and Business Media.

Gabrys, B. and Bargiela, A. (2000) General Fuzzy Min-Max Neural Network for Clustering and Classification. *IEEE Trans. On Neural Networks*, Vol. 11, No. 3, pp. 769-783.

Giudici, P. (2003) *Applied Data Mining: Statistical Methods for Business and Industry*. John Wiley and Sons, England.

Halkidi, M., Batistakis, Y. and Vazirgiannis, M. (2001). On Clustering Validation Techniques. *Journal of Intelligent Information Systems.* Vol. 17, No. 2, pp. 107-145

Han, J. and Kamber, M. (2001) *Data Mining: Concepts and Techniques*. Academic Press, USA.

Han, J. and Kamber, M. (2006) *Data Mining Concepts and Techniques*. Morgan Kaufmann Publishers.

Han, J., Kamber, M. and Pei, J. (2011) *Data Mining: Concepts and Techniques*. Elsevier.

Hastie, T., Tibshirani, R. and Friedman, F. (2009) *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer.

Hong, T. P. and Lee, C. Y. (1996) Induction of fuzzy rules and membership functions from training examples. *Fuzzy Sets Syst*., Vol. 84, pp. 33–47. Available online at: http://www.elsevier.com/locate/fss [Accessed on 10/01/2013].

International Federation of classification societies Conference. (2004) Classification, Clustering and Data Mining Applications. *Proceedings of the meeting of the international federation of classification societies (IFCS),* Illinois Institute of Technology, Chicago, Springer Science and Business Media, pp. 3.

Jang, J. S. R., Mizutani, E. and Sun, C. T. (1997) *Neuro-Fuzzy and Soft-Computing: A Computational Approach to Learning and Machine Intelligence*. Prentice-Hall.

Jiang, D., Pei, J. and Zhang, A. (2010) *DHC: A Density-based Hierarchical Clustering Method for Time Series Gene Expression Data*. Department of Computer Science and Engineering, State University of New York at Buffalo.

Kersting, K. (2006) *An Inductive Logic Programming Approach to Statistical Relational Learning*.IOS Press.

Kibler, D., and Aha, D. (1987) Learning Representative Exemplars of Concepts: An Initial Case Study. *Proceedings of the Fourth International Workshop on Machine Learning,* Irvine, CA: Morgan Kaufmann. pp. 24-30.

Klösgen, W. and Sytkow, J. M. (2002*). Handbook of Data Mining and Knowledge Discovery*. Oxford University Press, New York.

Kqanungo, T., Netanyahu, N. S. and Wu, A. Y. (2002) An Efficient k-Means Clustering Algorithm: Analysis and Implementation, *Ieee Transactions On Pattern Analysis And Machine Intelligence*, Vol. 24, No.7, pp. 881-892.

Kohonen, T. (1989) *Self-Organization and Associative Memory, 3rd ed*. Springer-Verlag

Koyu, V. and Deshpande, B. (2014) *Predictive Analytics and Data Mining: Concepts and Practice with Rapid Miner*. Morgan Kaufmann.

Kung, S. Y. (2014) *Kernel Methods and Machine Learning*. Cambridge University Press.

Larose, D. T. (2004) *Discovering Knowledge in Data: An Introduction to Data Mining*. Wiley-Inter science.

Lavrač, N., Motoda, H., Fawcett, T., Holte, R., Langley, P. and Adriaans, P. (2004) Introduction: Lessons Learned From Data Mining Applications and Collaborative Problem Solving. *Machine Learning,* Vol. 57, pp. 13-34.

Lirov, Y., Rodin, E. Y. and Ghosh, B. K. (1989) Automated Learning by Tactical Decision Systems in Air Combat. *Computer and Mathematics with Application*, Vol. 18, pp. 151-160.

Liu, B. (2007) *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data*. Springer.

Loveman, G. (2003) Diamonds in the data mine. *Harvard Business Review,* pp. 109-123.

Madhulata, T. S. (2012) An Overview on Clustering Methods. *IOSR Journal of Engineering,* Vol. 2, No. 4, pp. 719-725.

Maimon, O and Rokach, L. (2006) *Data Mining and Knowledge Discovery Handbook*. Springer Science & Business Media.

Maimon, O. and Rokac (2007) *Soft Computing for Knowledge Discovery and Data Mining*. Springer Science & Business Media

Markov, Z. and Larose, D. T. (2007) *Data Mining the Web: Uncovering Patterns in Web Content, Structure, and Usage*. Wiley-Inter science.

Mathkour, H. I. (2010) RULES3-EXT Improvements on RULES-3 Induction Algorithm. *Mathematical and Computational Applications*, Vol. 15, pp. 318-324.

Matignon, R. (2007) *Data Mining Using SAS Enterprise Miner*. Wiley-Interscience.

Mehrotra, K., Mohan, C.K., and Ranka, S. (1997) *Elements of Artificial ¬Neural ¬Networks*.

MIT Press, Cambridge, MA

Michalski, R. S. (1990) A Theory and Methodology of Inductive Learning. *Readings in Machine Learning,* J. W. Shavlik and T. G. Dietterich, eds, pp. 70-95, Morgan Kaufmann, San Mateo, California.

Michalski, R. S. (1996) The Multi-purpose Incremental Learning System AQ15 and its Testing Application to Three Medical Domains. *Proc. National Conf. on AI,* Philadelphia, PA., August, pp. 1041-1044.

Mirkin, B. (2005) *Clustering for Data Mining: A data Recovery Approach*. CRC press.

Modesitt, K. L. (1987) Space Shuttle Main Engine Anomaly Data and Inductive Knowledge Based Systems: Automated Corporate Expertise. *Proc. Conf. Artificial Intelligence for Space Applications*, Huntsville, Alabama, NASA CP-2492, pp. 203-212

Mohri, M., Rostmaixadeh, A and Talwalkar, A. (2012) Foundations of Machine Learning, MIT Press.

Monostori, L. (2002) AI and Machine Learning Techniques for Managing Complexity, Changes and Uncertainties in Manufacturing. *Proc. of the 15th Triennial World Congress*, Barcelona, Spain, pp. 119-130.

Murtagh, F. and Contreras, P. (2011) *Methods of Hierarchical Clustering*. Available online at: http://arxiv.org/pdf/1105.0121.pdf [Accessed on 15/12/2014].

Murphy, K, P. (2012) *Machine Learning: A Probabilistic Perspective*. MIT Press.

Pappa, G, L and Freitas, A, A. (2009) *Automating the Design of Data Mining Algorithms: An Evolutionary Computation Approach*. Springer Science & Business Media.

Park, N. H. and Lee, W. S. (2004) *Statistical Grid-based Clustering over Data Streams*. Available online at: http://sigmod.org/publications/sigmod-record/0403/A15.park-lee.pdf [Accessed on 15/08/2014].

Perner, P. (2009) *Machine Learning and Data Mining In Pattern Recognition: 6Th International Conference*. Springer Science & Bsuiness Media.

Pham, D. T. and Afify, A. A. (2005) RULES-6: A Simple Rule Induction Algorithm for Supporting Decision Making. *Industrial Electronics Society, IECON 31st Annual Conf. of IEEE*, Raleigh, North Carolina, USA, pp. 2184-2189.

Pham, D. T. and Afify, A. A. (2007) Clustering Techniques and Their Applications in Engineering: Intelligent Systems Laboratory. *Mechanical Engineering Science*, Vol. 221, Part C.

Pham, D. T., Afify, A. A. and Dimov, S. S. (2002) Machine Learning in Manufacturing. *Proc. of the 3rd CIRP Int. Seminar on Intelligent Computation in Manufacturing Engineering (ICME 2002)*, Ischia, Italy, pp. III-XII.

Pham, D. T. and Aksoy, M. S. (1993) An Algorithm for Automatic Rule Induction. *Artificial Intell. Eng*., Vol. 8, pp. 277-282.

Pham, D. T. and Aksoy, M. S. (1995a) RULES: A Simple Rule Extraction System. *Expert Systems with Applications*, Vol. 8, pp. 59-65.

Pham, D. T. and Aksoy, M. S. (1995b) A New Algorithm for Inductive Learning. *Journal of Systems Eng.*, Vol. 5, pp. 115-122.

Pham, D. T. and Bigot, S. (2003) RULES-5: A Rule Induction Algorithm for Classification Problems Involving Continuous Attributes. *Proc. Inst. Mechanical Engineers*, Part C, Vol. 217, pp. 1273-1285.

Pham, D. T., Bigot, S. and Dimov, S. S. (2004) A Rule Merging Technique for Handling Noise in Inductive Learning, *Proceedings of the Institution of Mechanical Engineers, Part C, Journal of Mechanical Engineering Science*, Vol. 218, pp. 1255- 1268.

Pham, D. T., Bigot, S. and Dimov, S. S. (2006) RULES-F: A Fuzzy Inductive Learning Algorithm. *Proceedings of the Institution of Mechanical Engineers, Part C, Journal of Mechanical Engineering Science*, Vol. 220, No. 9, pp. 1433-1447.

Pham, D. T. and Dimov, S. S. (1997) An Algorithm for Incremental Inductive Learning. *Proc. Instn Mech. Engrs, Part B: J. Engineering Manufacture,* Vol. 211(B3), pp. 239-249.

Pham, D. T. and Dimov, S. S. (1997) An Efficient Algorithm for Automatic Knowledge Acquisition. *Pattern Recognition,* Vol. 30, No. 7, pp. 1137-1143.

Quinlan, J. R. (1987) Inductive Knowledge Acquisition: A Case Study. In: *Applications of Expert Systems,* Ed: J. R. Quinlan, Turing Institute Press, pp. 157-173.

Quinlan, J. R. (1988). Induction, Knowledge and Expert Systems. In: *Artificial Intelligence Developments and Applications,* Eds J. S. Gero and R. Stanton, Amsterdam, North-Holland, pp. 253-271.

Rizzi, A., Panella. M, Frattale Mascioli, (2002) Adaptive Resolution Min-max Classifiers. *IEEE Transaction on Neural Network*, Vol. 13, No. 2, pp. 402–414.

Sajja, P. S. and Akerkar, R. (2012) Intelligent Technologies for Web Applications, CRC Press.

Selby, R. W. and Porter, A. A. (1988) Learning From Examples: Generation and Evaluation of Decision Trees for Software Resource Analysis. *IEEE Transactions on Software Engineering,* Vol. 14, pp. 1743-1756.

Simpson, P. K. (1992) Fuzzy Min_Max Neural Networks - Part 1: Classifications. *IEEE Transactions on Neural Networks,* Vol. 3, No. 5, pp. 776-786.

Simpson, P. K. (1993) Fuzzy Min_Max Neural Networks - Part 1: Classifications. *IEEE Transactions on Neural Networks,* Vol. 1, No. 1, pp. 32-45.

Sra, S., Nowozin, S. and Wright, S, J. (2012) *Optimization for Machine Learning*. MIT Press.

Teknomo, K. (2007*) K-Means Clustering Tutorial*. Available online at: http://croce.ggf.br/dados/K%20mean%20Clustering1.pdf [Accessed on 14/09/2014].

Thorpe, J. C., Marr, A. and Slack, R. S. (1989) Using an Expert System to Monitor an Automatic Stock Control System. *Journal of the Operational Research Society*, Vol. 40, pp. 945-952.

Tou, J. T. and Gonzalez, R. C. (1974) *Pattern Recognition Principles*. Addison-Wesley Publishing Company, Reading, Massachusetts.

Wagstaff, K. L. (2012) *Machine Learning that Matters*. Jet Propulsion Laboratory, California Institute of Technology, CA.

Wagstaff, K., Cardie, C., Rogers, S. and Schroedl, S. (2001) Constrained K-means Clustering with Background Knowledge. *Proceedings of the Eighteenth International Conference on Machine Learning,* pp.577-584.

Wang, L. X. and Mendel, J. M. (1991) Generating Fuzzy Rules from Numerical Data, With Applications. *Technical Report TR USC-SIPI #169*. Signal and Image Processing Institute, University of Southern California, CA, USA, 1991.

Witten, I. H. and Frank, E. (2000) *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufman Publishers, USA.

Witten, I. H. and Frank, E. (2005) *Data Mining: Practical Machine Learning Tools and Techniques, 2nd Edition*. Morgan Kaufman.

Wojtusiak, J., Michalski, R. S., Kaufman, K. A. and Pietrzykowski, J. (2006) The AQ21 Natural Induction Program for Pattern Discovery: Initial Version and its Novel Features. *18th IEEE International Conference on Tools with Artificial Intelligence*, pp. 523-526.

Wu, J. (2012) *Advances in K-means Clustering: A Data Mining Thinking*. Springer Science and Business Media.

Zaki, M. J. and Meira, W. (2014). *Data Mining and Analysis: Fundamental Concepts and Algorithms*. Cambridge University Press.

Zhou, Z., Li, H. and Yang, Q. (2007). Advances in Knowledge Discovery and Data Mining. *11th Pacific-Asia Conference,* Springer Science and Business Media.