A Novel Rule Induction Algorithm with Improved Handling of Continuous Valued Attributes

A thesis submitted to the Cardiff University

For the degree of

Doctor of Philosophy

By

Dinh Trung Pham

School of Engineering, Cardiff University

United Kingdom

2012

Abstract

Machine learning programs can automatically learn to recognise complex patterns and make intelligent decisions based on data. Machine learning has become a powerful tool for data mining. A great deal of research in machine learning has focused on concept learning or classification learning. Among the various machine learning approaches that have been developed for classification, inductive learning from examples is the most commonly adopted in real-life applications.

Due to non-uniform data formats and huge volume of data, it is a challenge for scientists across different disciplines to optimise the process of knowledge acquisition from data with naïve inductive learning techniques. The overarching purpose of this research is to develop a novel and efficient rule induction algorithm a learning algorithm for inducing general rules from specific examples that can deal with both discrete and continuous variables without the need for data pre-processing.

This thesis presents a novel rule induction algorithm known as RULES-8 which utilises guidelines for the selection of seed examples, together with a simple method to form rules. The research also aims to improve current pruning methods for handling noisy examples. Another major concern of the work is designing a new heuristic for controlling the rule formation and selection processes. Finally, it concentrates on developing a new efficient learning algorithm for continuous output using fuzzy logic theory. The proposed algorithm allows automatic creation of membership functions and produces accurate as well as compact fuzzy sets.

Acknowledgements

This thesis would not have been accomplished without the guidance and support of a great number of people to whom I would like to convey my utmost gratitude.

My first and most special thanks go to Professor D. T. Pham for his supervision, guidance and advice from the early stage of my studies at previous MEC – Manufacturing Engineering Central of Cardiff University. Professor Pham has generously granted me his time and precious encouragement besides consistent and valuable academic assistance during the past three years. I am indebted to him not just because he has been there as a backbone of this thesis. I found him truly a source of ideas and passions in science which inspire and enrich my growth personally, intellectually, and professionally.

I am also especially grateful to Dr. Michael Packianather for his outstanding constructive suggestions for the completion of this thesis. Dr. Packianather has provided me every and the most needed encouragement and support in various ways as Professor Pham moved on in this career. I am benefited by advice and critical comments from Dr. Packianather who has kindheartedly invested time in reading and supporting this research. I am indebted to him more than he knows.

Many thanks go in particular to Dr. Samuel Bigot for the insights he has shared. His inputs have been a crucial contribution to this thesis. I am grateful in every possible way to Dr. Bigot for he also – together with Dr. Packianather – has done an extraordinary job of taking over my supervision from Professor Pham.

I would like to also express my gratitude to other faculty and staff at MEC, and now the School of Engineering at Cardiff University for their various support for me to pursue my academics.

I owe deeply to the wonderful friends who have been more than a source of encouragement. Many thanks go particularly to Dr. Le Chi Hieu, and many other friends who I apologise for not being able to mention one by one, for giving me such a pleasant time during research discussions as well as hanging out around England.

Above all, my family deserves my deepest thanks for giving me strength and endless encouragement and support to carry out this work. I would like to specially mention my Farther, Pham Dinh Vinh, who unfortunately, had only been able to see me through half way in this journey due to serious illness, was the first person who showed me the joy of science pursuit since I was a child. His spirit has been a *point d'appui* for me through the good and bad days and a kind, persistent reminder not to give up.

My special appreciation goes to my wife for her dedication and love. I owe her for unselfishly taking care of my sons so that I have my shoulders free for my research endeavor. I am fortunate to have her hand in marriage.

Last but not least, I would like to thank the Vietnamese government for their financial sponsorship for my academics at Cardiff University.

Declaration

This work has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed.....(Candidate) Date.....

Statement 1

This thesis is being submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Signed.....(Candidate) Date.....

Statement 2

This thesis is the result of my own investigations, except where otherwise stated. Other sources are acknowledged by footnotes giving explicit references.

Signed.....(Candidate) Date.....

Statement 3

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed.....(Candidate) Date.....

Contents

List of figures

List of tables

Notations

AI Artificial Intelligence

A_i The i^{th} attribute in an example.

ML Machine Learning

DM Data Mining

BPP Basic Post Pruning

C_E The class value in example E

CE The Closest Example to SE not belonging to the target class

$Cond_R^i$ The condition in rule R for the ith attribute

$Cond_j^k$ The k^{th} condition in the i^{th} disjunction

K_k^{out} The k^{th} fuzzy set created for the fuzzification of the output values, Tr(a(k), b(k), c(k))

F_k^i The fuzzy set employed in the fuzzy rule R to form a condition on the i^{th} attribute

F_R^{out} The output fuzzy set of the rule R

IPP Incremental Post Pruning

IREP Incremental Reduced Error Pruning

n The number of negative examples covered by the newly formed conjunction;

N The total number of negative examples (examples not belonging to the target class);

Nf The number of output fuzzy sets fixed by the user

NL The noise level

$N_{\_unclassified}$ The number of examples belonging to the target class and not classified by the rule set formed so far.

p The number of positive examples covered by the newly formed conjunction;

P The total number of positive examples (examples belonging to the target class);

$P_{\_unclassified}$ The number of examples belonging to the target class and not classified by the rule set formed so far.

S A set of instances.

SE A Seed Example

T A training set of examples

Th The noise threshold

T_PRSET The Temporary Partial Rule Set

Tr Triangular membership function

V_E^i The value of the ith attribute in example E

V_R^i The discrete value employed in the rule R to form a condition on the i^{th} discrete attribute

$V_{\max}^i$ The maximum known value of the i^{th} continuous attribute

$V_{\min}^i$ The minimum known value of the i^{th} continuous attribute

V_E^{out} The value of the continuous output in example E

$V_{\max}^{out}$ The maximum known value of the continuous output

$V_{\min}^{out}$ The minimum known value of the continuous output

Chapter 1

INTRODUCTION

1.1 Background

In recent years, robust advancements in the field of information technology have made the capacity of data collection and storage increase with incredible speed. Besides, the informationalisation, which is taking place rapidly on a large scale of different socioeconomic activities, has created an immense amount of information. Millions of different databases are being used that hold gigabytes or even terabytes of data. There is a need to tame those enormous sources of data into useful information and knowledge. Data mining, which is now no longer a new concept, has attracted a great deal of attention in the information industry and in society as a whole.

Because of the diversity of disciplines that contributes to data mining, data mining research is expected to generate a large variety of data mining systems. These systems can be categorised as database-oriented, statistics, machine learning, visualisation pattern recognition, neural networks, and so on (Jiawei and Micheline, 2000)

Machine learning is a branch of artificial intelligence, studying approaches that can automatically learn to recognise complex patterns and make intelligent decisions based on the available data. In real applications, inductive learning as an example of such approaches has been one of the most commonly adopted methods for classification besides concept learning and classification learning.

Perhaps the most well known inductive learning method is **decision tree induction** - a method in which a model is created to predict the value of the target variable based on

several input variables. A tree can be learned by splitting the source set into subsets based on an attribute value test. Because of the popularity of this representation technique, many decision tree algorithms have been developed. ID3 (Quinlan, 1986) is one of the most popular algorithms and has been improved several times by a number of researchers ever since it was invented. The most recent versions of this algorithm are C4.5 (Quinlan, 1993) and C5 (Rulequest Research, 2001). Both are integrated into a commercially available software package. Although the decision tree method has a high predictive accuracy and can be easily interpreted by users in using the decision tree for a model, the output attribute must be categorical. The problem is there can be a lot of attributes as the size of data increases, the trees created from numeric datasets can be complex and less efficient.

Another method is **rule induction** which represents classification knowledge in the form of a set of rules to describe each class. Like decision tree learning, there are many rule induction algorithms. Among them are AQ (Michalski, 1969; Michalski et al., 1986; Cervone et al., 2001; Michalski and Kaufman, 2001), CN2 (Clark and Niblett, 1989; Clark and Boswell, 1991) and RIPPER (Cohen, 1995). All these algorithms employ the same general method that was used for the first time in the AQ algorithm. AQ21 is the most recent version of the AQ family (Michalski and Wojtusiak, 2006).

The AQ family and some of the algorithms mentioned above have been improved from time to time and have been able to solve some drawbacks of the decision tree. However, since they extract rules and then remove the covered examples from a training set of examples, fragmentation had been one of the problems of these algorithms.

RULES (RULE Extraction System) is a family of simple inductive learning algorithm inspired by ideas from both AQ and CN2. The RULES family is different from the other algorithms in that it does not induce rules on a class-per-class basis but instead considers

the class of the selected seed example as the target class (Shehzad, 2009). It then attempts to induce rules that cover as many examples of the target class as possible using the rule evaluation function. At present, the RULES family has extended to Rules-7 (Pham and Khurram, 2010). Among members of the RULES family, Rules-5 (Pham and Samuel, 2004) is a noteworthy simple but efficient algorithm. It is also known as the Dyna algorithm. Its strength lies in its ability to handle continuous attributes. RULES-5 also employs a more efficient search mechanism as well as a new post-pruning technique (Pham and Bigot, 2004) in order to handle noisy data. Thanks to these advantages, RULES-5 has been successfully employed in different applications. However, it also has drawbacks that prevent it from being adopted for many real-life applications. Hence, there is the need for a new method that is able to achieve good accuracy, compact rule sets and natural induction.

1.2 Aim and objectives

The overarching aim of this thesis is to propose a novel rule induction algorithm, a simple and efficient method which is able to deal with either discrete or continuous variables without the need to preprocess data.

This research is based on RULES-5 (Pham and Bigot, 2004) developed at Cardiff University. This algorithm employs rule forming as a specific method for condition selection based on the consideration of distributions of examples. However, each seed example leads to the creation of a particular rule, and different sequences of seed examples can yield different rule sets. A new method will be developed to make sure that the sequence of seed examples leads to the best rule set. In addition, the new method also improves, the accuracy and simplification of rule forming, as well as expands its real life applications.

Finally, the resulting inductive learning algorithm will be further modified for handling continuous outputs.

To achieve the overall aim of the research, the following objectives were set:

- To survey current inductive learning techniques.

- To develop a new simple algorithm that has guidelines for the selection of seed examples.

- To design a new heuristic for controlling the rule formation and selection processes.

- To develop a new algorithm to handle continuous output using fuzzy logic.

1.3 Methodology

To provide background for this research, an in-depth review of the existing literature was carried out regarding inductive learning techniques and fuzzy logic for inductive learning. The review was intended to cover both discrete and continuous outputs.

It can be said that the inductive learning algorithms devised so far for extracting rules applied for discrete output have helped solve many real life problems. However, in today's world, the fact that data is accumulating in surmounting volume as well as diversity is exceeding the capability of the existing algorithms. It is necessary, therefore, to invent new algorithms or improve current ones for more effective and beneficial data mining. With respect to this demand in the field, this research presents a new algorithm, RULES-8, and compares it against its predecessor RULES-5 on a diverse population of datasets.

RULES-8 proves to be able to provide a decent result for discrete output, especially when it employs a new heuristic measure to evaluate rule quality. Based on RULES-8, another algorithm – TVFuzz – was designed to deal with continuous output. The new TVFuzz technique was also compared against the DynaFuzz using some real models.

1.4 Outline of the thesis

Chapter 1 provides a brief introduction to the research and states its objectives.

Chapter 2 reviews and gives background information about the research area, including the main inductive learning concepts and descriptions of number of existing algorithms based on these concepts.

Chapter 3 presents a novel rule induction algorithm called RULES-8 that utilises guidelines for the selection of seed examples and proposed a simple method to form rules. This chapter also details an improved pruning method for handling noisy examples.

Chapter 4 designs a new heuristic for controlling the rule formation and selection processes. The performance of the heuristic is compared with that of other heuristics.

Chapter 5 describes a new efficient learning algorithm for continuous output using fuzzy logic theory. This algorithm allows automatic creation of membership functions and produces accurate as well as compact fuzzy sets. It also resolves some drawbacks in RULES-5

Finally, chapter 6 summarises the thesis and proposes directions for further research.

Chapter 2

LITERATURE REVIEW

2.1 Preliminaries

Human history has entered its third millennium. The information accumulated over thousands of years has exceeded the capacity of human brains. A perpetuating concern in the science world has been how to make that huge mountain of data useful. Research in this regard has achieved some degree of success, yet scientists are far from being satisfied with what they can learn from the data. During the middle of the 1980s, the concept of Knowledge Discovery from Data (KDD) was conceived and began to help people explore potential knowledge and benefits of immense databases. KDD is a process including several phases among which data mining is the most essential (Jiawei and Micheline, 2000). This is the phase when new information is discovered. The process of knowledge discovery is itself the course of receiving, analysing, applying, and improving the achievements of previous discoveries. Various examples of this area of the science world are database technology, statistics, pattern recognition, information retrieval, neural networks, knowledge-based systems, artificial intelligence, high-performance computing, data visualisation, and so on.

Like other knowledge discovery tools, machine learning (ML) also has the central purpose of "learning from data." ML algorithms play an essential role in data mining. Most research in machine learning has focused on concept learning or classification learning. The mechanism of this kind of learning is the induction of the definition for a general category from specific positive and negative examples of that category. In real-world

application domains, inductive learning from examples is perhaps the most commonly adopted machine learning approache developed for classification.

Inductive learning is simply known as a process of acquiring knowledge by drawing inductive inferences from data. The study of inductive learning is mainly motivated by the desire to automate the process of knowledge acquisition during the construction of expert systems. This chapter presents a background on machine learning with a focus on inductive learning developed for prediction and classification models. The chapter is organised as follows: Section 2.2 presents a framework for knowledge discovery including input and output format as well as discovery method. Section 2.3 describes in detail inductive learning approaches to classification including major decision tree and rule induction algorithms. Section 2.4 presents some basic concepts of fuzzy logic and existing algorithms for automatic fuzzy rule generation from numerical examples. A summary of the chapter is presented in section 2.5

2.2 A framework for knowledge discovery

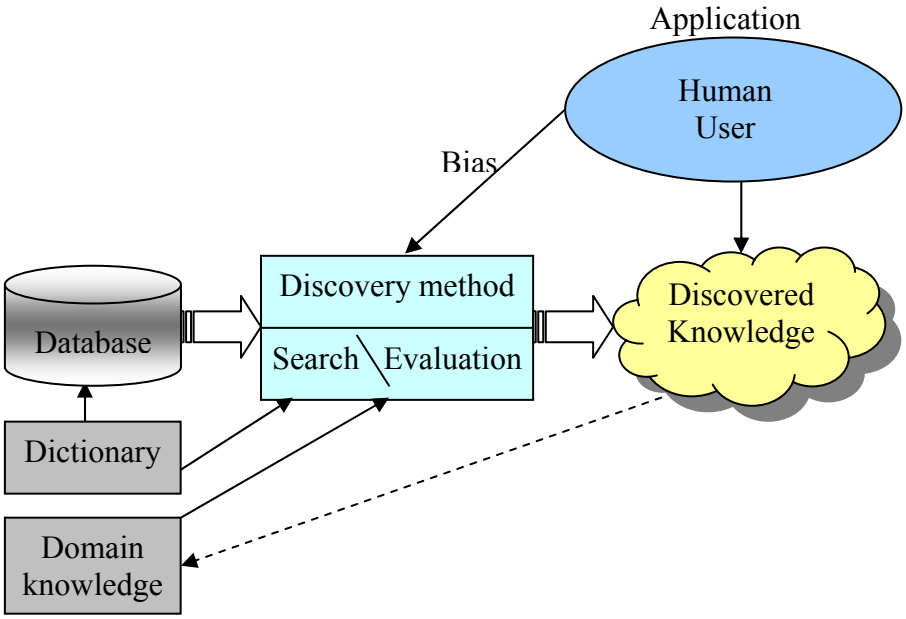


Figure 2.1: A framework for knowledge discovery in databases

Figure 2.1 illustrates the basic components of a prototypical system for knowledge discovery in database. In this model, the input includes raw data from the database, information from the data dictionary, additional domain knowledge, and a set of user defined biases that provides a high-level focus. All these will be computed and evaluated by the discovery method so that new knowledge is discovered. Discovered knowledge is the output and can be directed to the user or back into the system as new domain knowledge.

As can be seen in Figure 2.1, the discovery method is the central process designed to extract knowledge from data. This activity usually involves two processes, namely identifying and describing noteworthy patterns in a concise and meaningful manner. The identification process, also referred to as unsupervised learning in ML, categorises or clusters records into subclasses that reflect patterns inherent in the data. The descriptive process, in turn, summarises relevant qualities of the identified classes. This process is known as supervised learning in ML.

Pattern Identification: Discovering pattern classes is a problem of pattern identification or clustering. There are two basic approaches to this problem: traditional numeric methods and conceptual clustering. Traditional methods of clustering come from cluster analysis and mathematical taxonomy (Dunn and Everitt 1982). These algorithms produce classes that have a maximum level of similarity within classes but a minimum level of similarity between classes. Various measures of similarity have been proposed, most based on Euclidean measures of distance between numeric attributes. Accordingly, these algorithms only work well on numeric data. An additional drawback is their inability to use background information, such as knowledge about similar cluster shapes. There have been attempts in conceptual clustering to overcome these problems. These methods work with

nominal and structured data and determine clusters not only by attribute similarity but also by conceptual cohesiveness, as defined by background information.

Although successful under certain conditions, these methods do not always equal the human ability to identify useful clusters, especially when dimensionality is low and visualisation is possible. This situation has prompted the development of interactive clustering algorithms that combine the computer's computational powers with the human user's knowledge and visual skills.

Concept Description: Describing the useful pattern classes once having been identified is a more important task than just simply enumerating them. In machine learning, this process is known as supervised concept learning from examples, i.e. to derive an intentional description of a class given a set of objects labeled by class. Empirical learning algorithms, the most common approach to this problem, work by identifying commonalities or differences among class members. Well-known examples of this approach include decision tree inducers (Quinlan 1986), rule induction (Michalski et al 1969), neural networks (Rummelhart and McClelland 1986), and genetic algorithms (Holland et al. 1986).

Some learning approaches, such as explanation-based learning (Mitchell, Keller, and Kedar-Cabelli 1986), require a set of domain knowledge (called a domain theory) in order to explain why an object falls into a particular class. Other approaches combine empirical methods and knowledge-based ones. The main drawback of empirical methods is their inability to use available domain knowledge. This failure can result in descriptions that encode obvious or trivial relationships among class members. Discovery in large, complex databases clearly requires both empirical methods to detect the statistical regularity of patterns and knowledge-based approaches to incorporate available domain knowledge.

Because different tasks require different forms and amount of information, they often influence discovery algorithm selection or design. The next section discusses a learning method developed for classification model.

2.3 Inductive learning for classification model

The concept of learning has been tackled by many authors. Simon (1983), on a broader term, defined learning as changes in the system that are adaptive in the sense that they enable the system to do the same task or tasks drawn from the same population more efficiently and more effectively the next time. Shavlik and Dietterich (1990) narrowed it down to describe inductive learning, which is a process accomplished by reasoning from supplied examples to produce general rules.

Inductive learning can be categorised as either supervised inductive learning or unsupervised inductive learning (Afify, 2004). In supervised learning a supervisor gives direct feedback to the learner about the appropriateness of its performance. This is in sharp contrast to unsupervised learning where this kind of feedback is absent. Since this thesis focused on supervised inductive learning procedures, all the algorithms discussed in this section refer to supervised inductive learning methods.

2.3.1 Decision tree learning

The major purpose of the decision tree method is to select step by step an attribute to decompose training set into several subsets until there remains a unique class in each subset. The result of this method generally takes the form of a tree, with class names as leaves and other nodes representing attribute-based tests for possible outcomes as branches.

Let S represent an example set and $A = \{A^1, A^2, ...A^n\}$ be the condition attribute set with observable value sets $V^i = \{V_1^i, V_2^i, ..., V_n^i\}$ respectively. C is the decision attribute with domain $C = \{C_1, C_2, ...C_k\}$. The general decision tree inductive learning procedure is as follows:

Step1: Select attribute i^{th} to decompose example set S into $S_{i1}, S_{i2}, ..., S_{in}$ (n is possible value of attribute i^{th}). For a continuous attribute A^i, a binary test is carried out, and a corresponding branch $V^i < V_t^i$ is created, with a second branch corresponding to $V^i > V_t^i$, where V_t^i is a threshold in the domain of A^i)

Step 2: For each subset S_{ik}, if there is a unique class in it, then stop decomposing, and label the node as a class. Otherwise, continue to decompose S_{ik} in the same way as described in step 1.

Figure 2.2: The general Decision tree inductive learning procedure

Table 2.1 shows an example data set and Figure 2.2 displays a decision tree constructed from this data.

Table 2.1 Training set for the Alarm problem

Example	Sensor_1	Sensor_2	Alarm
1	-1	0	OFF
2	0	0	OFF
3	-1	1	ON
4	0	1	OFF
5	0	1	OFF
6	1	1	ON
7	-1	0	OFF
8	-1	1	ON

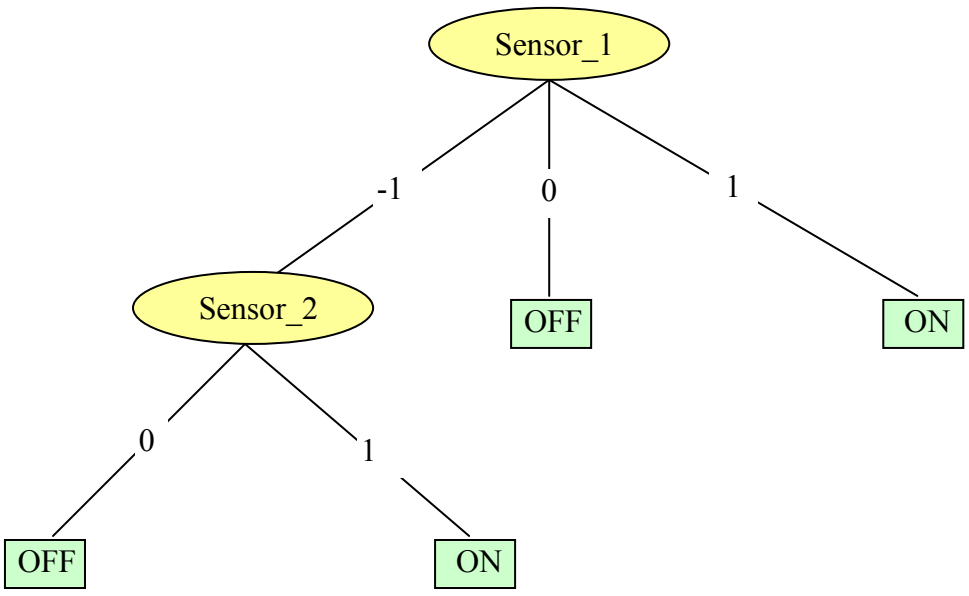


Figure 2.3: Representation of the formed decision tree

In order to classify an object, one starts at the root of the tree, evaluate the test, and takes the branch appropriate to the outcome. The process continues until a leaf is encountered, at which time the object is asserted to belong to the class named by the leaf.

The earliest Decision tree learning system is Concept Learning System (CLS). It was introduced by Hunt et al (1966). In the CLS series, Hunt decomposes S by using a heuristic look-ahead method which utilises values which appear most frequently. CLS has nine versions, numbered from CLS1 to CLS9. The main difference between the first eight versions and the latest version CLS9 is that the former versions use only binary decomposition while CLS9 can provide non-binary decomposition.

In 1983, Quinlan presented the ID3 inductive learning algorithm [Quinlan 1983], also a descendant of the CLS. ID3 uses an information entropy measure to guide the decomposition.

Information entropy is defined as:

$$I(S) = -\sum_{i} p(C = C_i, S) \log_2 p(C = C_i, S) \qquad (2.1)$$

$$E(A_i, S) = \sum_{\substack{A_i = v_{ik} \\ v_{ik} \in V_i}} \frac{|S_j|}{|S|} I(S_j) \qquad (2.2)$$

Where $p(C = C_i, S)$ be the proportion of instance in S

I(S) is the whole information entropy in set S

E(Ai,S) denotes the information entropy when S is divided based on the condition attribute Ai.

The information gain is defined as:

$$Gain(A_i, S) = I(S) - E(A_i, S). \qquad (2.3)$$

ID3 chooses the optimal decomposition at a node by maximising the information gain (Pitas et al. 1992).

By using the information gain as a guide, ID3 tends to choose the decomposition based on the condition attribute that has more values. This might cause ID3 to miss more general decision trees. Therefore, Quinlan introduced the information gain ratio to overcome this weakness (Quinlan 1986a). The information gain ratio is defined as:

$$Gain\_ratio(A_i, S) = \frac{I(S) - E(A_i, S)}{IV(C_i)} \qquad (2.4)$$

Where $IV(A_i)$ denotes the degree of randomness of the distribution of the examples in S when partitioned using Ai

$$\mathrm{IV}(\mathrm{A_i}) = -\sum_{\substack{\mathrm{A_i}=\mathrm{v_{ik}} \\ \mathrm{v_{ik}} \in \mathrm{V_i}}} \frac{|\mathrm{S_j}|}{|\mathrm{S}|} \log_2 \left(\frac{|\mathrm{S_j}|}{|\mathrm{S}|}\right) \qquad (2.5)$$

ID3 was also developed into a series of learning algorithms known as the ID3 family.

ID3-IV is a version of the ID3 family, modified with the information gain ratio (Quinlan 1986a) and named by Cheng (Cheng et al. 1988). Cheng et al. (1988) continued to introduce Generalised ID3, or GID3. GID3 uses the information gain ratio and only generates a new branch (i.e. forms a new subset) when encountering a relevant value. The relevance of a value is evaluated by a user-determined tolerance level. This modification is intended to avoid over-specific decision trees.

PRISM is based on ID3 but uses a modified information entropy measure which tries to reduce redundant condition values in the learning output (Cendrowska 1988).

Subsequent to ID3 were ID4 (Schlimmer and Fisher 1986), ID5 (Utgoff 1988), ID5R and ID5R-hat (Utgoff 1989).

C4.5 is an industrial version of ID3 (Quinlan 1993). It has the same basic structure as ID3-IV. It features pruning of decision trees and then conversion of the pruned decision trees into rule sets. In addition, C4.5 can handle continuous values, noise and missing values.

Breiman et al. (1984) present CART (Classification and Regression Trees). CART is designed for handling continuous-valued examples. In such examples, all the condition attributes have continuous values and the decision attribute has discrete values. As a Decision tree inductive learning method, CART can only generate binary decision trees. It uses the *Gini* index of diversity denoted by i(t). As a measure of node impurity to guide the decomposition of the example set at a node t, i(t) is defined as follows:

$$i(t) = \sum_{j \neq k} p(j \mid S)\, p(k \mid S) \qquad (2.6)$$

Where S denotes the set at node t, j and k are classes in set S and p(j|S), p(k|S) denotes the probability of examples of class j and k in set S in turn.

Let S represents a continuous-valued example set and $A = \{A^1, A^2, ...A^n\}$ be the condition attribute set. The learning process of CART can be briefly described as follows:

At a node t, CART searches through the condition attributes $\{A^1, A^2, ...A^n\}$ one by one. For each condition attribute, it finds the best decomposition by maximising the decrease of impurity. Then, it selects the best decomposition from n candidates.

Clearly, at a node, the best decomposition is dependent on the measure of impurity. When that measure changes, the best decomposition will also change.

Crawford presents an extension of CART (Crawford 1990), OC1, which is able to generate oblique decompositions (Murthy et al. 1994). The oblique decomposition is the decomposition that is not parallel to coordinate axes in example sets. A smaller decision tree can be generated by using oblique decomposition.

2.3.2 Rule induction

Mitchell (1978) introduced the Candidate-Elimination algorithm, which served as the basis to develop the Rule induction method. The Rule induction method is to establish a hypothesis rule space which is based on a given example set and then to refine (search through) the hypothesis rule space to find more general rules. The hypothesis rule space is also called the version space.

Among the rule inductive methods devised based on the Candidate-Elimination algorithm is Cohen and Feigenbaum's (1982) Extension-Against method. The algorithm is described as follows.

Let S be an example set. The examples are categorised as either positive or negative according to their classes. If an example belongs to the class of interest, it is considered positive. Examples belonging to all other classes are considered negative.

H denotes the version space. Initially, H contains all the possible concepts which are based on the given positive examples. Then, as examples are presented, candidate concepts are eliminated from H. The elimination process goes on until only one kind of concept for the same class remains in H, and this is the desired kind of concept. The procedure can be described in more details as follows:

When a positive example is presented, H will be generalised, that is, specific concept descriptions are removed from H. When a negative example is presented, H will be specialised, that is, very general concept descriptions are removed from H. In this way, H gradually shrinks until only the desired concept descriptions remain. The learned concept is represented in the form "IF [description] THEN [decision]".

This algorithm can find the most acceptable concepts based on the given example set. However, for a large example set, it can be very difficult or even impossible to construct the initial H.

Perhaps one of the best known Rule induction methods is AQ algorithm. Credited to Michalski (1975), this algorithm is similar in principle to the Candidate-Elimination algorithm. The main difference is that, initially, H contains the null description (the most general concept) only. Let an example set S be divided into a positive set S^+ and a negative set S^-. The examples in S^+ are in the same class, and $S^+ \cup S^-$=S and $S^+ \cap S^-$=$\varnothing$. The basic idea in the AQ algorithm is to find a cover CV($S^+ | S^-$) which covers S^+ against S^-,

that is, it separates S^+ from S^-. A simplified version of the AQ algorithm is as Figure 2.3 follows:

Step1: Randomly select a "SEED" example e from the positive examples set S^+.

Step 2: Generate an orderly disjoint "START" G(e| S^-), e$\in S^+$. The star is against the set S^-.

Step 3: Find the "*best rule"* from the START according to user-defined criteria. Remove the examples covered by this rule from S^+.

Step 4: If the positive example set S^+ is not empty, return to step 1 and continue the procedure. Otherwise, the obtained rules constitute a complete and consistent concept of S^+.

Figure 2.4: A simplified version of the AQ algorithm procedure

In the most general sense, the START G(e| S^-) of example e, e$\in S^+$, is a set of all possible alternative non-redundant descriptions of example e that do not cover examples in S^-. S^- thus acts as a constraint on the possible descriptions of e. The START can be generated using a number of methods (Michalski 1983) which constitute the kernel of the AQ algorithm.

The advantages of AQ are two-folds: first, because the search is based on a Seed Example (SE), AQ will find the shortest descriptions for a concept; second, AQ can control the concepts forming procedure by using some user-defined criteria.

Various AQ based inductive learning algorithms have been developed such as AQ11 (Michalski 1983) and AQ14-NT (Pachowicz and Bala 1991). AQ15 is an incremental version of AQ11 (Michalski et al. 1986). AQ15-GA incorporates a genetic algorithm technique to guide the search (Vafaie and DeJong 1994). Some multistage versions of AQ based algorithms, AQ17-DCI, AQ17-FCLS, AQ17-HCI and AQ17, have also been reported (Wnek and Michalski 1994). Over the past decades, AQ algorithms have continued to be developed into a series of learning algorithms called the AQ family. The most recent member of the AQ family is AQ21 (Wojtusiak and Michalski, 2006).

Another renowned method is CN2 (Clark and Niblett, 1989). It is a rule induction algorithm named after its authors. CN2 attempts to combine the good features of the decision tree algorithm ID3 with the rule induction algorithm AQ. CN2 uses a subset of the expression language VL1 used in AQ and also retains the beam search strategy of AQ. However, it differs from AQ in that it does not rely on specific examples during search but instead considers all specialisations of a complex similar to ID3, which considers all attributes in order to find the best one to split at a particular node. The specialisation of a complex involves either adding a new conjunctive term or removing a disjunctive element

in one of its selectors. Because of this top-down search for complexes, the CN2 algorithm is able to incorporate a cutoff method similar to the one used in decision tree pruning that can halt specialisation of a complex when no further statistically significant specialisations can be found. This results in an extension of the search space to include rules that do not perform perfectly on the training data.

Similarly to ID3, to handle continuous attributes CN2 algorithm divides the range of values of each attribute into discrete subranges and then creates two thresholds on the attribute at subrange boundaries. One advantage is that CN2 also includes any missing values for both discrete and continuous-valued attributes. In case of discrete attributes, the missing value is replaced with the most commonly occurring value of that attribute in the training data. For continuous attributes, the mid-value of the most commonly occurring subrange is used to replace any missing values.

The original version of CN2 produces an ordered set of rules. In the case where an example satisfies none of the rules, it is classified using the default rule at the end of the list, which assigns it the class label of the most frequently occurring class within the training data. CN2 uses two rule evaluation measures, namely entropy and statistical significance. The latter is measured using the likelihood ratio statistic (Kalbfleish, 1979) which is defined as:

$$LikelihoodRatio(F,E) = 2\sum_{i=1}^{n} f_i \log \frac{f_i}{e_i} \qquad (2.7)$$

The function assesses the significance of the complex by comparing the observed frequency f_i of examples satisfying the complex among classes with the expected frequency e_i if the rule made random predictions.

The CN2 algorithm has been modified in a later version (Clark and Boswell, 1991), which enables it to generate an unordered set of rules. It also replaced the entropy rule evaluation measure with the Laplace expected error estimate which is given by:

$$LaplaceAccuracy(n_{class}, n_{covered}, k) = \frac{n_{class} + 1}{n_{covered} + k} \qquad (2.8)$$

Where

n_{class} is the number of examples of the target class covered by the rule

$n_{covered}$ is the total number of examples covered by the rule

k is the number of classes

The modified algorithm also incorporates a stopping criterion to check if the Laplace estimate of the best complex is better than that of the default rule. If this is the case, the induction continues for the current class. Otherwise, the new complex is not deemed to bring about an improvement and so rule generation for the current class terminates. Since the new version of CN2 generates an unordered set of rules, so a conflict resolution approach is also adopted in order to resolve any clashes that might occur. In case a new example satisfies more than one rule predicting different classes, a probabilistic method is

used in which the distribution of covered examples of each rule among classes is summed to find the most probable class.

Among the more recent methods of rule induction is the RULES family. RULES (RULe Extraction System) is a family of simple inductive learning algorithms which inherit ideas from both AQ and CN2 algorithms. The RULES family is different from the other algorithms in that it does not induce rules on a class-per-class basis but considers the class of the selected seed example as the target class. It then attempts to induce a rule that covers as many examples of the target class as possible using the rule evaluation function. In addition, the RULES family only marks the examples covered by previous rules instead of removing them.

RULES (Pham and Aksoy, 1993), RULES-2 (Pham and Aksoy, 1995b) and RULES-3 (Pham and Aksoy, 1995a) were the first three algorithms in the family. Later, Pham and Dimov developed a new rule induction algorithm RULES-3 Plus (Pham and Dimov, 1997b) that incorporated the beam search strategy instead of greedy search and used a new rule evaluation measure called the H-measure (Lee, 1994), which is defined as:

$$H = \sqrt{\frac{p+n}{P+N}}[2 - 2\sqrt{\frac{p}{p+n}.\frac{P}{P+N}} - 2\sqrt{(1-\frac{p}{p+n})(1-\frac{P}{P+N})}] \qquad (2.9)$$

Where, P is the total number of positive examples (examples belonging to the target class);

N is the total number of negative examples (examples not belonging to the target class);

p is the number of positive examples covered by the newly formed rule;

n is the number of the negative examples covered by the newly formed rule.

At present, the RULES family has extended to Rules-7 (Shehzad, 2010). Among members of the RULES family, Rules-5 (Bigot, 2004) is a noteworthy simple but efficient algorithm.

2.3.3 Rules-5 algorithm

Rules-5 (Bigot, 2004) is also known as the Dyna algorithm, which is described in Figures 2.5 and 2.6. In the Dyna algorithm, the search strategy employed is based on seed examples. Each seed example leads to the creation of particular rules, and different sequences of seed examples can yield different rule sets. Thus, a guideline for selection of the seed examples needs to be developed in the later version.

It should be noticed that in the Dyna algorithm, a new heuristic for rules evaluation is designed. The heuristic is defined as follows:

$$S = \frac{p}{p+n}\frac{p_{-new}}{P_{-unclassified}}\left(1-\frac{n}{N}\right) \qquad (2.10)$$

Where: P is the total number of positive examples (examples belonging to the target class);

N is the total number of negative examples (examples not belonging to the target class);

p is the number of positive examples covered by the newly formed rule;

n is the number of the negative examples covered by the newly formed rule.

p_new is defined as the number of positive examples covered by the newly formed conjunction of conditions and not covered by previously created rules

P_unclassified is defined as the number of examples belonging to the target class and not classified by the rule set formed so far

Step 1: Select Seed example (SE), SE is considered a positive example

Step 2: Find conditions of a new rule according to a consideration of distribution of examples by looking for the closest example (CE) not belonging to the target class and covered by the rules formed so far

Step 3: If there are uncovered examples, go back to *Step 1*

Figure 2.5 Dyna rule forming procedure

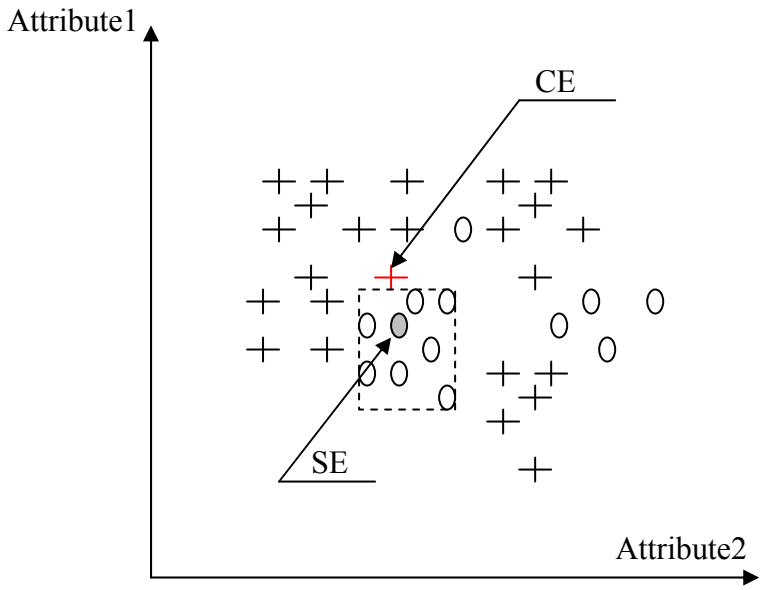


Figure 2.6: Graphical representation of the coverage of the new rules

Most inductive learning algorithms dealing with discrete classes use a similar input data structure. The following section discusses another trend of learning for handling continuous classes.

2.4 Learning fuzzy logic from examples

Fuzzy set theory has been in existence for almost half a century and has been proved extremely useful in many control applications as well as non-control applications requiring decision-making in uncertain environments. This section summarises some basic concepts of fuzzy logic and existing algorithms for fuzzy inductive algorithms.

2.4.1 Fuzzy logic basic concepts

- *A fuzzy set* F defined on U is characterised by its membership function $\mu_F(u) \in [0,1]$ (Klir and Folger, 1992), $\mu_F(u)$ is the degree of membership. $\mu_F(u) = 0$ means that u does not belong to the fuzzy set and $\mu_F(u) \in]0,1]$ means that u belongs to the fuzzy set with a degree of certainty of $\mu_F(u)$.

- *Membership function* $\mu_F(u)$ maps the degree to which an element *u* belongs to a fuzzy subset F from domain U to the range [0,1]

Supposing that U is the set of all non-negative integers and F is a fuzzy subset of U labeled "approximately 10," then, the fuzzy subset can be represented by a membership function, $\mu_{10}(u)$. Figure 2.7 depicts a possible definition of the membership function.According to this fuzzy subset, the number 10 has a membership value of 1.0 (i.e., 10 is exactly l0), and the number 9, a membership value of 0.5 (i.e., 9 is roughly 10 to the degree of 0.5). Note that a membership function in general can be linear, trapezoidal, convex curve, or in many other forms (Lee 1990, Dummerrnuth 1991) but to facilitate computation, triangular forms are widely adopted because they are easy to define as Tr(a,b,c) (Figure 2.8) This representation will be employed in this study.

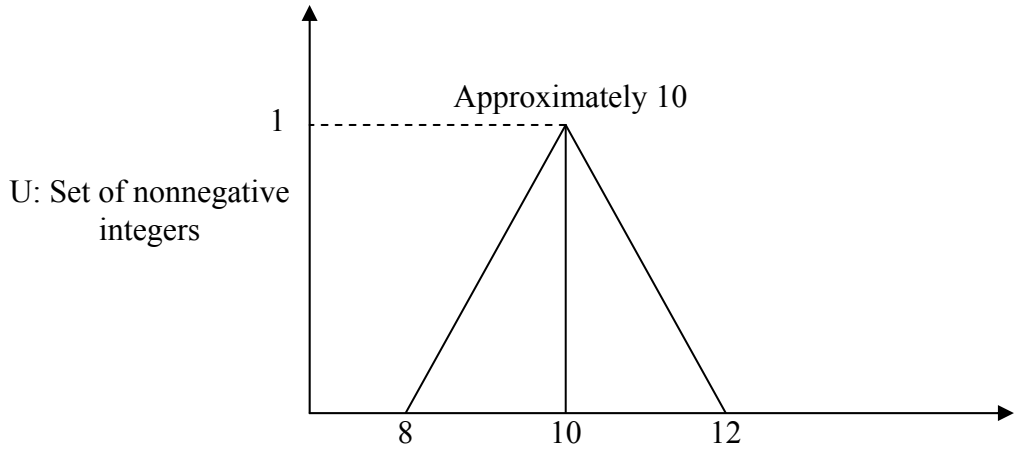


Figure 2.7: MF defining the concept "approximately 10" on a fuzzy system.

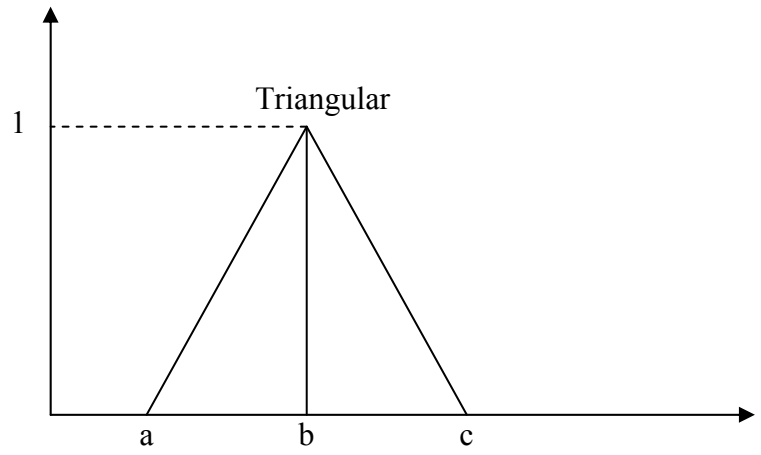


Figure 2.8: Triangular membership functions

- A fuzzy rule R is composed of a number of possible fuzzy conditions on each of the m attributes and an output fuzzy set (F_R^{out}) it can be represented as follows: $Cond_R^1 \wedge ... \wedge Cond_R^i \wedge ... \wedge Cond_R^m \rightarrow F_R^{out}$

2.4.2 Fuzzy logic system

Fuzzy Logic Systems (FLS) are one of the main developments and successes of fuzzy logic. They are motivated by the biological brain's ability to learn, reason and generalise using noisy or uncertain information (Lei 1999).

A general structure of a fuzzy logic system consists of four units including fuzzification, defuzzification, inference engine, and rule base (Dadone 2001; Lee 1990a; Passino and Yurkovich 1998). A general structure of a fuzzy logic system is described in Figure 2.9

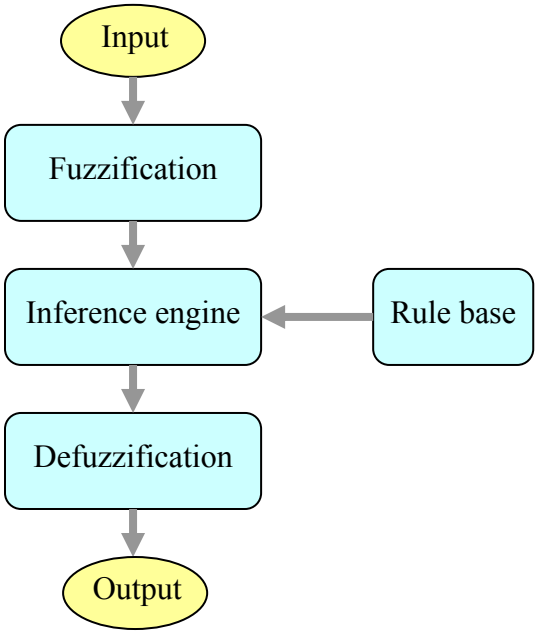


Figure 2.9: A general structure of a fuzzy logic system

To illustrate these steps, the following example (Grabot, 1998) will be used. A fuzzy rule set is needed in order to control a car moving towards a wall, bringing it close to the wall

in the most efficient way. There are two attributes, the speed of the car ($Speed \in [0km/h, 60km/h]$), and its distance from the wall ($\text{Distance} \in [0m, 60m]$), and one output, the deceleration ($Brake \in [0m/s^2, 12m/s^2]$).

a. Fuzzification

Fuzzification is the process which translates measured values into real values between 0 and 1. It also assigns these values degrees of truth, usually called membership degrees, for the linguistic values of the input linguistic variables. For the instance above, the tree parameters are decomposed as follows (Figure 2.10):

The linguistic variables of attribute Speed is divided into the following fuzzy sets: zero, low, moderate and high.

The linguistic variables of attribute Distance is divided into the following fuzzy sets: zero, short, moderate and long.

The linguistic variables of attribute Brake is divided into the following fuzzy sets: zero, soft, medium, hard and max.

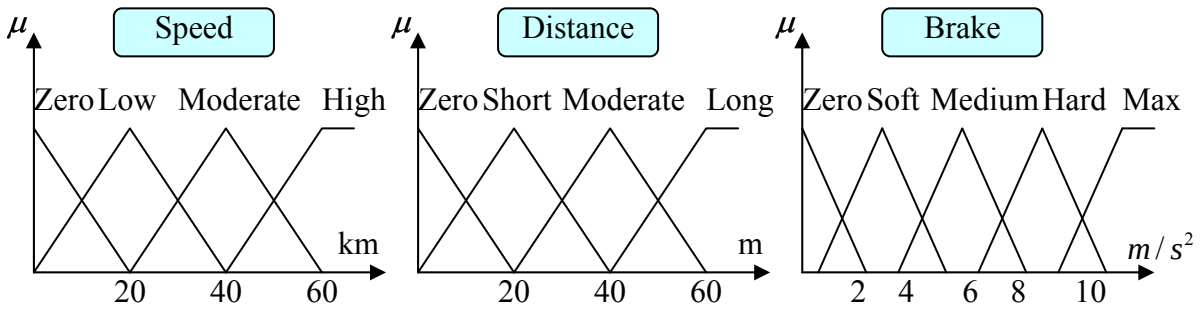


Figure 2.10: Decompose in membership function

b. Rule base

The rule base of a fuzzy logic system consists of a set of fuzzy IF-THEN rules. To help the generation of all rules, the typical method of multidimensional matrix representation called the decision table. Each cell in the matrix is called a fuzzy subspace and represents a possible rule when linked with a particular output. Table 2.2 shows the decision table generated from the previous example.

Table 2.2: Decision table

Speed \ Distance	Zero	Short	Moderate	Long
Zero	Zero	Zero	Zero	Zero
Low	Max	Medium	Soft	Zero
Moderate	Max	Hard	Medium	Zero
High	Max	Hard	Medium	Zero

Rule base set includes:

1. IF [Speed = Zero] and [Distance = Zero] THEN [Brake = Zero]

2. IF [Speed = Zero] and [Distance = Short] THEN [Brake = Zero]

3. IF [Speed = Zero] and [Distance = Moderate] THEN [Brake = Zero]

4. IF [Speed = Zero] and [Distance = Long] THEN [Brake = Zero]

5. IF [Speed = Low] and [Distance = Zero] THEN [Brake = Max]

6. IF [Speed = Low] and [Distance = Short] THEN [Brake = Medium]

….

c. Reference engine

Since fuzzy logic systems are stimulated by the biological brain's capability to make decisions, the inference engine or fuzzy reasoning is considered a method of cloning a human decision making process of judging and giving a proper fuzzy output depending on the inputs and the rule base

d. Defuzzification

Defuzzification is the mapping from the linguistic fuzzy output defined over an output universe into a crisp output space (Awadalla 2005). There are many defuzzification strategies. The most common strategies are the Weighted Average, Mean of Maxima and Centroid.

- Weighted Average method used the formula:

$$output = \frac{\sum_{R=1}^{r} \mu\_rule_R(E).C_{out}}{\sum_{R=1}^{r} \mu\_rule_R(E)}$$

Where E is the new example,

C_{out} is the centre of the output fuzzy set of the considered rule

r is the total number of rules

- The Mean of Maxima method (MoM): This method selects the mean of output values within the group of possible output fuzzy sets that correspond to the highest membership degree. If more than one solution exists, the mean is taken. A graphical example can be seen in Figure 2.11.

- The Centre of Gravity method (CoG): This method was first suggested by Zadeh (Roychowdhury and Wang, 1996). The output value is obtained by assessing the centre of gravity of the resulting group of fuzzy sets (considering overlapping or not). A graphical example is shown in Figure 2.12. This method can be highly computationally costly since the centre of gravity might not always be easy to assess, in particular when considering overlapping.

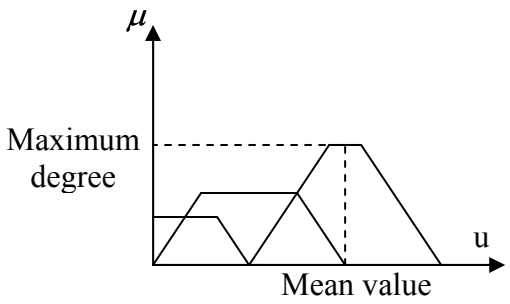


Figure 2.11: M.o.M defuzzification

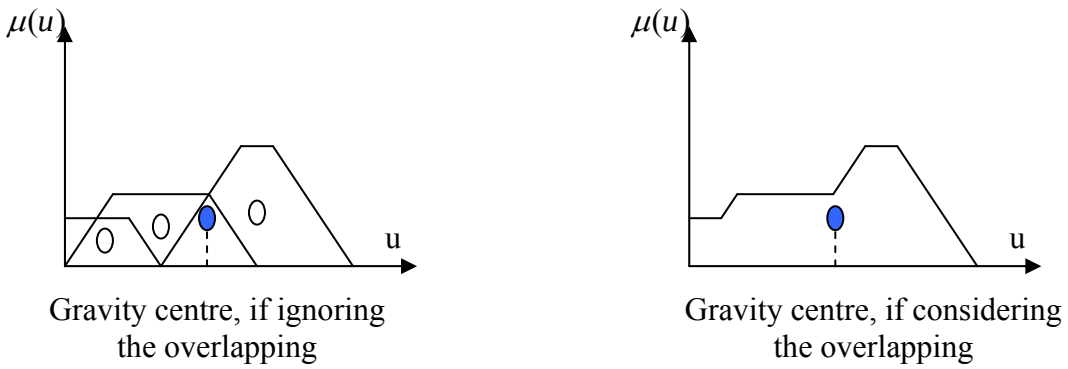


Figure 2.12: C.o.G defuzzification

2.4.3 Generating fuzzy rule from numerical data

Perhaps the most popular method is the Wang and Mendel algorithm (Wang and Medel, 1991). The method determines a mapping from the input space to the output space based on the combined fuzzy rule base using a defuzzification procedure. Follows is a detailed description of this method:

Step 1: Divide the Input and Output Spaces into Fuzzy Regions

Step 2: Generate Fuzzy Rules from the Given Data Pairs

Step 3: Assign a Degree to Each Rule

Step 4: Create a Combined FAM Bank

Step 5: Determine a Mapping based on the FAM Bank

Figure 2.13: Wang and Medel procedure intended to generate fuzzy rules

The mapping was proven to be capable of approximating any real continuous function to an arbitrary accuracy. However, as mentioned by the authors, there is a problem of "growing memory": when more training examples become available, more rules are generated and the selection of the best rules becomes difficult. To help this selection, Delgado and Gonzalez use a method based on the definition of frequencies in each fuzzy domain, which allows one to identify if any possible rule is a "true rule"(Delgado and Gonzalez, 1993).

Similar to Wang and Mendel's algorithm, Nozaki and Ishibuchi (1997) proposed to use a particular heuristic method to automatically generate fuzzy if-then rules from numerical data. The fuzzy if-then rules with non-fuzzy singletons (i.e., real numbers) in the consequent parts are generated by assessing single real numbers (instead of membership functions) which are to be stored in each cell of the decision table. Since there is no defuzzification step involved, Nozaki and Ishibuchi's algorithm is quite simple. However, the problem of growing memory still remains.

Sebag and Schoenauer(1994) observed that the problem of designing membership functions might be just as complex as designing fuzzy rules. Both methods above need to pre-define membership functions, which is actually not an easy task. Various methods for automatic creation of membership functions have been devised. However, all still cannot solve the existing problems, not to mention the rise of new problems. One of the challenges is the demand of post-processing the large fuzzy rule sets formed to acquire more compact rule sets.

To resolve the problems for automatic membership functions design, Hong and Lee (1996) proposed a method, by which the fuzzification of output is performed using a clustering procedure that regards examples in the training set (T) with close output values as belonging to the same fuzzy set. Appropriate membership functions are then assigned to represent each fuzzy set. Initial membership functions are assigned to attributes in the form of a triangle base equal to a small interval predefined by a user. After attributes have received their initial membership functions, the decision table is built using the examples in T. The decision table is then simplified through the process of merging membership functions from which the final rule sets can be extracted.

Nevertheless, a problem still exists with the merging process regardless of the attempt to improve it by the authors Hong and Lee (1999). That is, it can be highly computationally expensive as the number of attributes increases. Hong and Chen (2000) also attempted to develop a method to simplify the initial membership functions. Their method has been proven to be more efficient and accurate, yet it does not reduce considerably the computational cost. This method, however, generates a set of fuzzy rules where the membership functions have been automatically created and the universes of discourse are not equally partitioned. It should be noted that it is still a challenge when the number of attributes increases.

Another type of algorithms based on machine learning techniques was developed by Shann and Fu (1995). This algorithm uses the neural network structure that allows the use of the error back-propagation learning algorithm for fuzzy rule generation. However, one of the weaknesses of this algorithm is that the membership functions need to be predefined. There are many other algorithms using neural networks and most of them use a method similar to Shann and Fu's algorithm.

In Yuan and Shaw (1995), continuous attributes are first fuzzified using human experts or techniques such as fuzzy clustering, and then the membership functions are employed by each attribute as possible branches during the construction of the decision tree. However, this algorithm is designed for classification problems only and it is not clear how it would handle real or fuzzy outputs. In addition, the membership functions also need to be predefined.

Perhaps the most common way of creating fuzzy decision trees is to use a method similar to the FILM algorithm (Jeng et al., 1997) where a crisp decision tree is first created using ID3 and then fuzzification operations are applied to modify it. The membership functions

are created automatically, but the models can only be used for problems with discrete outputs; fuzzy logic is only employed to handle vagueness and ambiguity in the attribute values.

Other methods also exist. Wang et al. designed an algorithm based on the PRISM learning strategy (Wang et al., 1999). More recently, a method was proposed using a combination of inductive learning and genetic algorithms (Castro et al., 2001). However, like many others (Ravi et al., 2001; Wang et al., 2001), all these fuzzy inductive learning methods have been developed only for classification problems where the fuzzy concepts are used to deal with noise, uncertainty and imprecision in the attribute values.

In 2004, Bigot released a new technique called DynaFuzz that can automatically create input membership functions. Inheriting the advantages of its predecessors – theDyna and DynaSpace inductive learning algorithms, DynaFuzz can generate more compact and more accurate fuzzy rule sets. However, in the present world, given that databases are becoming more diverse with more features as well as bigger quantity, there is an increasing need for improving existing algorithms and developing new ones. The need for further research on the automation of creating output membership functions is also of no exception.

2.5 Summary

This chapter has given background information on different machine learning algorithms with attention focused on inductive learning. The basic concepts of inductive learning algorithms have been described and the two main types of these algorithms currently available presented. The chapter has also presented the basic concepts of fuzzy logic relevant to fuzzy rule generation. Finally, existing algorithms for automatic rule generation based on these concepts are discussed.

Chapter 3

RULE 8: A NOVEL RULE INDUCTION LEARNING ALGORITHM

3.1 Preliminaries

Ian and Eibe (2005) observed, "We are overwhelmed with data. The amount of data in the world, in our lives, seems to go on and on increasing and there is no end in sight." Achievements of digital archive technology have especially reinforced this observation. Everyday, a huge amount of data is being accumulated in all social, economic, technological, and production activities and is becoming highly valuable resources which could support or lead people to new understandings in socio-economic networking, in manufacturing as well as in scientific research activities. A challenge for scientists across different disciplines has been how to optimise the process of acquisition of knowledge from data.

Inductive learning is a form of data analysis that can be used to extract models that are able to describe important data classes or predict future data trends. Such analysis can provide a better understanding of the data at large. In machine learning, many inductive methods have been proposed. They can be divided into two main categories, namely decision tree induction and rule induction.

In rule induction, the search strategy employed is based on seed examples. Each seed example leads to the creation of particular rules, and different sequences of seed examples can yield different rule sets. In the current version of these algorithms there is no guideline for the selection of seed examples. The first uncovered example found in the training set (T) is always selected, and therefore the rule set created will depend on the storage order of the examples in T.

This chapter presents RULES-8, a proposed new rule induction algorithm that addresses the weaknesses of the predecessors. In particular, it selects a candidate attribute-value instead of a seed example to form a new rule to make sure that the candidate attribute-value leads to the best rule. The conjunction is also formed by incrementally adding conditions which are selected by utilising a specific heuristic measure. In addition, a rule simplification technique is also improved to create more compact rule sets and minimises overlapping between rules.

The chapter is organised as follows: Section 3.2 gives a detail description of the new rule induction algorithm. Section 3.3 discusses in detail the problems identified with Pruning techniques in Dyna algorithm. An improvement pruning technique will also be presented in this section. The next section, Section 3.4, provides the results obtained from an experimental evaluation of RULES-8 on some benchmark datasets. Finally, section 3.5 summarises and concludes the chapter.

3.2 The Novel Learning Algorithm

3.2.1 Representation and Basis concepts

Like its predecessors, RULES-8 also extracts IF-THEN rules directly from a set of examples called the training set (T). Each example is described by a vector of attribute-value pairs, together with a specification of the class that belongs. Uncovered attribute-value pairs are attribute-values in uncovered examples.

Conditions on nominal attributes are equality tests of the form $[A_i = v_{ij}]$, where A_i is the attribute and v_{ij} is one of its valid values. Conditions on continuous attributes are

inequalities of the form [$A_i > t_{i1}$] or [$A_i \leq t_{i2}$], where t_{i1} and t_{i2} are two thresholds in the domain of attribute A_i.

Seed attribute-value is a candidate condition, which if being applied to a rule, the newly-created rule will be capable of covering the most examples

An attribute-value constitutes a condition. If the number of attributes is n_a, a rule may contain between one or n_a conditions, each of which must be a different attribute-value. Only the conjunction of conditions is permitted in a rule and therefore the attributes must all be different if the rule comprises more than one condition.

The class to be learned is called the target class. Examples of the target class in the training set are called positive examples. Examples in the training set that do not belong to the target class are called negative examples.

A rule is said to cover an example if the example satisfies all the rule conditions. A rule is said to be consistent if it covers none of the negative examples in the training set, and it is complete if it covers all the positive instances in the training set.

3.2.2 Learning algorithm description

Unlike its predecessors in the RULES family and other rule induction techniques, the new learning algorithm begins by selecting a seed attribute-value. Based on the seed attribute-value, this algorithm employs a specialisation process searching a general rule by incrementally adding new conditions to them.

To select the best conjunction of conditions, a heuristic H measure is also used to assess the information content of each newly formed rule. It can be defined as follows:

$$H = \sqrt{\frac{p+n}{P+N}}[2 - 2\sqrt{\frac{p}{p+n}.\frac{P}{P+N}} - 2\sqrt{(1-\frac{p}{p+n})(1-\frac{P}{P+N})}] \quad (3.1)$$

Where

P is the total number of positive examples (examples belonging to the target class);

N is the total number of negative examples (examples not belonging to the target class);

p is the number of positive examples covered by the newly formed rule;

n is the number of the negative examples covered by the newly formed rule.

The following figure describes the procedures of the newly proposed inductive learning algorithm.

Rule Forming Procedure

Step 1: Select randomly one uncovered attribute-value from each attribute to form an array $SETAV = [A_{.1}\ A_{.2}\ ..\ \ A_{.j}]$, then mark them in the training set T.

Step 2: Form array expression T_EXP from SETAV

Step 3: Compute H measure for each expression in T_EXP and sort them out according to the accuracy of expressions and then add them to the PRSET (highest H measure). If the H measure of the newly formed rule is higher than the H measure on any rules in the PRSET, the new rule replaces the rule having the lowest H measure.

Step 4: Select an expression with a potential condition (A_{ij}) having the highest H measure to find the seed attribute-value (A_{is})

IF the expression having the highest H measure covers correctly all covering example THEN the potential condition (A_{ij}) is selected as a seed attribute-value (A_{is})

Assume that it covers n examples

- Removes this expression from the PRSET

- Go to *step 7*

ELSE go to *step 5*

Step 5: Check uncovered attribute-values

IF there are not uncovered attribute-values THEN Stop

ELSE check uncovered attribute-values

IF there are uncovered attribute-values that have not been marked

THEN go to *step 1*

ELSE go to *step 6*

Step 6: Form a new array SETAV by combining the potential attribute-value with other attribute value in the same example $SETAV = [A_{ij} + A_{i1}; A_{ij} + A_{i2}; .. \quad A_{ij} + A_{ik}]$

Go to *step 2*

Step 7: Form a set of conjunctions of condition SETCC by combining A_{is} with other attribute-values, $SETCC = [A_{is} + A_{i1}; A_{is} + A_{i2}; .. \quad A_{is} + A_{ik}]$

Step 8: Form array rule Temporary Rule Set (T_RSET) from SETCC

Step 9: Compute H measure for each expression of T_RSET and sort them out according to the consistency of expressions and then add them to the NRSET (highest H measure). If the H measure of the newly formed rule is higher than the H measure on any rules in the NRSET, the new rule replaces the rule having the lowest H measure.

Step 10: Select an expression with the conjunction of conditions (A_{isc}) having the highest H measure to test the consistency. Assume that it covers m examples

IF m> n THEN A_{isc} is considered as a new seed attribute-value

Go to *step 6*

ELSE add the new rule to RuleSet

Go to *Step 5*

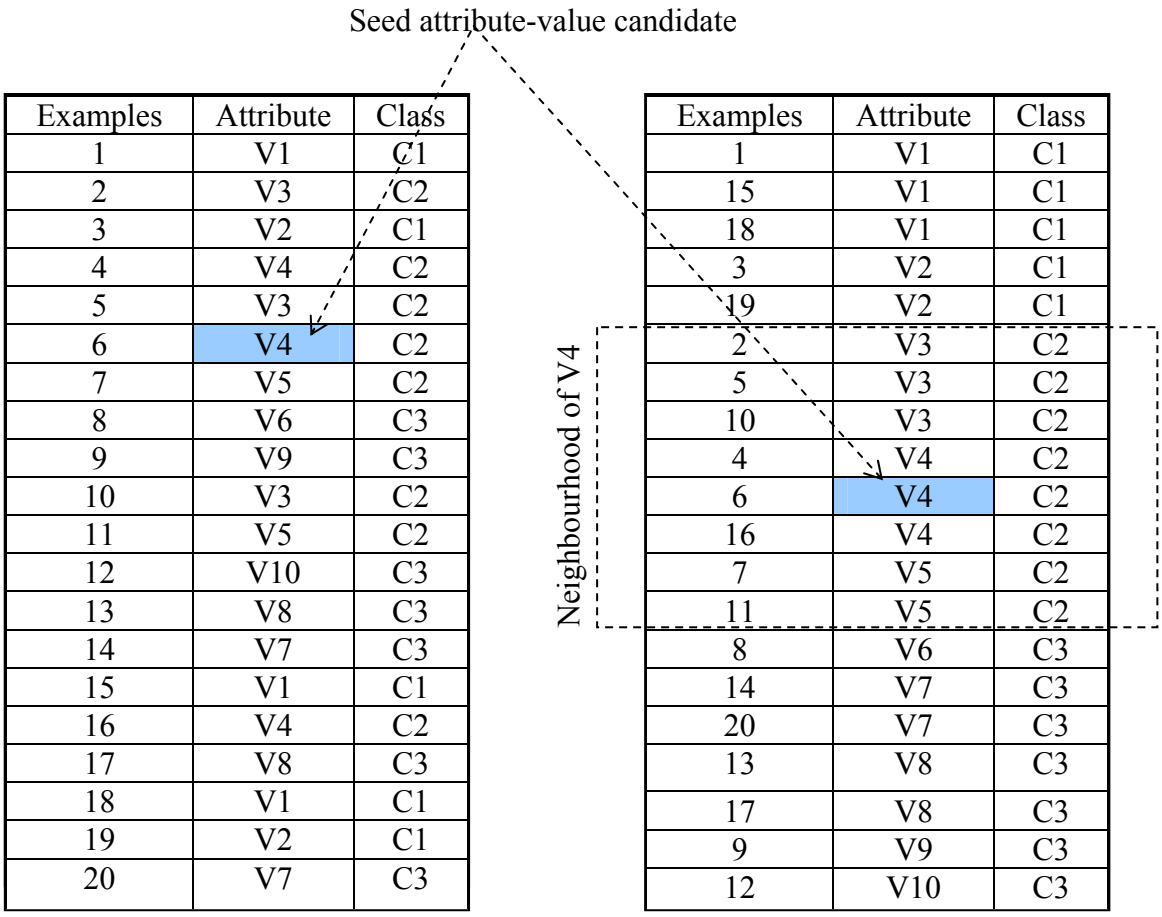
Figure 3.1: A pseudo-code description of rule forming procedure

3.2.2.1 Seed attribute-value selection

The proposed new mechanism, like other covering methods, searches for rules that best cover the examples from the target class meanwhile exclude examples belonging to other classes. Pham and Dimov (1997a) developed RULES-3 Plus in which the learning process employs a beam search method and considers all conditions extractable from, starting with condition having the highest H measure to form a rule. However, there are downsides in terms of speed and accuracy when applying to large amount of examples and values other than discrete values since the search space grows exponentially with the number of the descriptive attributes for the data.

In the proposed new algorithm, the learning process considers each attribute-value a state of search space. Hence, if a data set consists of *e* examples, and there are *n* attributes of each example, the search space includes *e* * *n* states. To find a seed attribute-value, a hill climbing method is used by selecting randomly one attribute-value from each attribute to form an array of SETAV. An array of rules is constructed accordingly. The attribute-value in a rule having the highest H measure is the seed attribute-value. In this way, the shortcoming of the speed of beam search method using in RULES-3 Plus has been overcome.

In the case of a continuous attribute, the seed attribute-value will be determined based on the size of neighbourhood. The size of neighbourhood is the range of attribute-values belonging to the target class. This can be clarified with the help of Figure 3.2, which shows typical data containing three classes and having ten values for a particular attribute. An attribute-value V4 is selected as a potential attribute-value; after employing a specialisation process, the range of attribute-values from V3 to V5 are found [Vmax, Vmin], the format of the formed condition will be $V_{\min} \le V \le V_{\max}$



Examples	Attribute	Class
1	V1	C1
2	V3	C2
3	V2	C1
4	V4	C2
5	V3	C2
6	V4	C2
7	V5	C2
8	V6	C3
9	V9	C3
10	V3	C2
11	V5	C2
12	V10	C3
13	V8	C3
14	V7	C3
15	V1	C1
16	V4	C2
17	V8	C3
18	V1	C1
19	V2	C1
20	V7	C3

Examples	Attribute	Class
1	V1	C1
15	V1	C1
18	V1	C1
3	V2	C1
19	V2	C1
2	V3	C2
5	V3	C2
10	V3	C2
4	V4	C2
6	V4	C2
16	V4	C2
7	V5	C2
11	V5	C2
8	V6	C3
14	V7	C3
20	V7	C3
13	V8	C3
17	V8	C3
9	V9	C3
12	V10	C3

Figure 3.2: Illustration of the concept size of neighbourhood

3.2.2.2 The conjunction of condition selection

In the newly proposed learning algorithm, the condition in the temporary rule having the highest H measure is selected as a seed attribute-value. The condition of a rule is formed by incrementally adding new attribute-values. The process continues until a rule is generated, covering the number of examples that is not higher than that before adding more conditions.

Assuming that the seed attribute-value is A with the range from $a_{\min}$ to $a_{\max}$ and another attribute needs to combine is B. The following procedure is to determine the new condition when combining the seed attribute-value and another attribute:

The condition forming procedure

Step 1: Initial condition of the rule $[a_{\min}, a_{\max}]$

Step 2: Determine the range of attribute B given a = $[a_{\min}, a_{\max}]$, and following the target class. For example, b = $[b_{\min}, b_{\max}]$

Step 3: Determine the range of attribute A, given b = $[b_{\min}, b_{\max}]$, and following the target class. For example, a = $[a_{\min-new}, a_{\max-new}]$

IF $[a_{\min-new} < a_{\min}]$ or $[a_{\max-new} > a_{\max}]$

THEN $[a_{\min} = a_{\min-new}]$ and $[a_{\max} = a_{\max-new}]$

Go to *step 2*

ELSE stop process

New condition are $[a_{\min}, a_{\max}]$ and $[b_{\min}, b_{\max}]$

In case of discrete attribute $a_{\min} = a_{\max}$ and $b_{\min} = b_{\max}$

Figure 3.3: A pseudo-code description of the condition forming procedure

Example: The distribution of dataset is showed in figure 3.4, the blue boxes are the distribution of negative examples and the yellow boxes are the distribution of positive examples (the target class). Assuming that the seed attribute-value is selected with the range from a_1 to a_2. The condition forming procedure when combining the seed attribute-value (A) with another attribute (B) is explained with reference to the steps given in Figure 3.3.

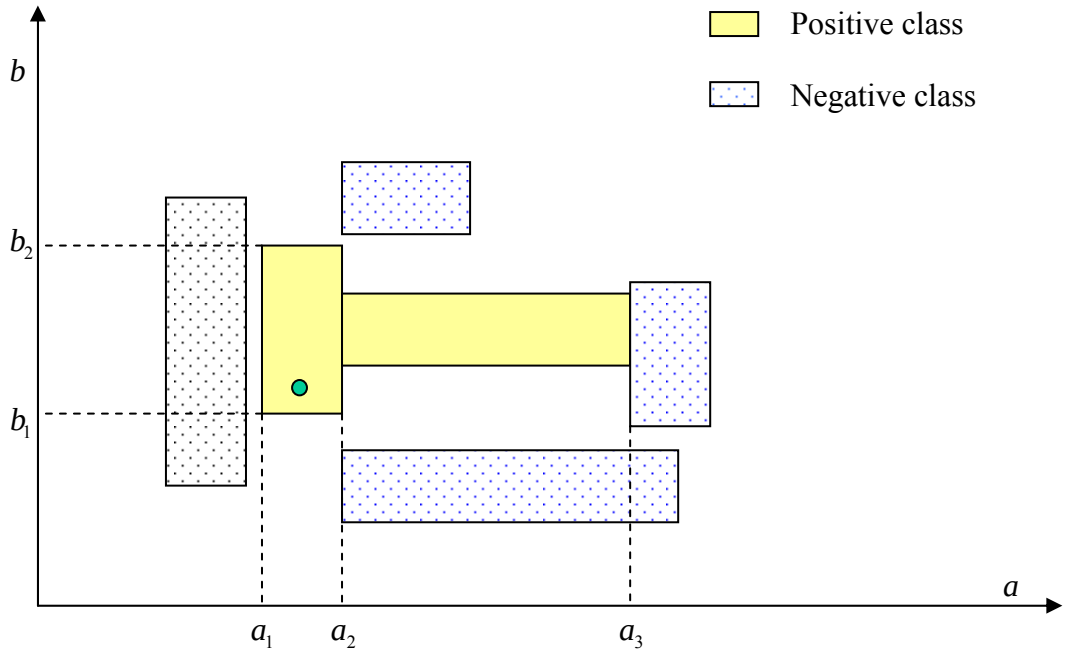


Figure 3.4: The conjunction selection between two continuous attribute-values

Step 1: Initial condition of the rule $[a_{\min} = a_1, a_{\max} = a_2]$

Step 2: Determine the range of attribute B given A = $[a_1, a_2]$ and following the target class

Result B = $[b_1, b_2]$

Step 3: Determine the range of attribute A given B = $[b_1, b_2]$, and following the target class

Result A = $[a_1, a_3]$

Due to $[a_3 > a_2]$ THEN $[a_{min} = a_1]$ and $[a_{max} = a_3]$

Go to *step 2*

- Determine the range of attribute B given A = $[a_{min} = a_1, a_{max} = a_3]$ and following the target class.

Result B = $[b_1, b_2]$

- Due to $[b_{min} = b_1, b_{max} = b_2]$ constant

End process

New condition are $[a_{min} = a_1, a_{max} = a_3]$ and $[b_{min} = b_1, b_{max} = b_2]$

3.2.2.3 Illustrative learning algorithm

To illustrate how the rule is formed in the new algorithm, the training data shown in Table 3.1 is used. The problem involves four attributes, namely Heat Treatment (HT), Material (M), Tolerance (T) and Finish (F) which have output values respectively as follow:

- Heat Treatment: Yes, No

- Material: Steel-3135, Aluminum, Steel_1045

- Tolerance: It can be a range of continuous values from 7 to 14.

- Finish: High, Medium, Low

The output system can be from R1 to R4. The rule forming procedure is explained with reference to the steps given in Figure 3.1

Step 1: A Set of Attributes and Values is constructed by selecting randomly attributes-values from 4 attributes, for examples SETAV = {[Heat Treatment = Yes], [Material = Steel_3135], [Tolerance = 10], [Finish = Low]} which can be clarified as table 3.2 below:

Table 3.2: Data set

SETAV

Examples	Heat Treatment	Material	Tolerance	Finish	Route
1	Yes	Steel_3135	10	Medium	R2
2	No	Aluminum	12	Medium	R3
3	Yes	Steel_3135	8	Medium	R2
4	No	Steel_1045	14	Low	R3
5	No	Aluminum	7	High	R4
6	Yes	Steel_1045	9	Medium	R1
7	No	Aluminum	9	Medium	R3
8	No	Aluminum	10	High	R4
9	No	Aluminum	10	Low	R3
10	Yes	Steel_1045	7	Medium	R1
11	No	Steel_1045	7	Low	R3

Step 2,3: Form array expression T_EXP using the conditions stored in SETAV, the following expressions are formed and stored in PRSET:

1. IF [Material = Steel_3135] THEN [Route = R2]; H = 0.489175

2. IF [Finish = Low] THEN [Route = R3]; H =0.340278

3. IF [Heat Treatment = Yes] THEN [Route = R2]; H = 0.07102

4. IF [Tolerance = 10] THEN [Route = R4]; H = 0.015947

Step 4: A rule is produced as the first expression in PRSET applies to only one class. Therefore [Material = Steel_3135] is a seed attribute-value

NewRule: IF [Material = Steel_3135] THEN [Route = R2]; H = 0.489175

Step 6: Form a set of conjunctions of conditions SETCC = {[Material = Steel_3135] and [Tolerance = 14], [Material = Steel_3135] and [Finish = Medium], [Material = Steel_3135] and [Heat Treatment = No]}

Steps 7,8: Form array rule Temporary Rule Set (T_RSET) and NRSET then as follows:

1: IF [Material = Steel_3135] and [Finish = Medium] THEN [Route = R2]; H =0.489167

2: IF [Material=Steel_3135] and [Heat Treatment= No] THEN [Route = R2]; H =0.489167

3: [Material = Steel_3135] and [Tolerance = 14] THEN [Route = R2]; H =0.345893

Step 9: The expressions 1 and 2 have the highest H measure but they cover 2 examples that are not bigger than NewRule. Thus, NewRule is added to RuleSet.

Rule 1: IF [Material = Steel_3135] THEN [Route = R3]; H = 0.489175

Rule 1 can classify examples 1 and 3

The process continues to Step 5 and goes to *Step1*

Step 1: Four attributes-values continue to be selected to form SETAV. For example,

SETAV = {[Heat Treatment=No], [Material=Steel_1045], [Tolerance=9], [Finish = High]}

Step 2: Form array expression T_EXP from SETAV

1. IF [Heat Treatment = No] THEN [Route = R4], H = 0.012138

2. IF [Material = Steal_1045] THEN [Route = R1], H = 0.07102

3. IF [Tolerance = 9] THEN [Route = R1], H = 0.050219

4. IF [Finish = High] THEN [Route = R4], H = 0.489167

Step 3: PRSET is adjusted as follows:

1. IF [Finish = High] THEN [Route = R4], H = 0.489167

2. IF [Finish = Low] THEN [Route = R3]; H =0.340278

3. IF [Heat Treatment = Yes] THEN [Route = R2]; H = 0.07102

4. IF [Tolerance = 9] THEN [Route = R1], H = 0.050219

The first expression applies to only one class. Like other steps to form Rule 1, a new rule is obtained.

Rule 2: IF [Finish = High] THEN [Route = R4], H = 0.489167

Rule 2 covers examples 5, 8.

The process continues to *Step 5* and goes to *Step1* and then another rule is obtained

Rule 3: IF [Finish = Low] THEN [Route = R3]; H =0.340278

Rule 3 covers examples 4, 9 and 11.

The process continues to *Step 5* and goes to *Step1*

Step 1: Form SETAV = {[HT = Yes]; [M = Aluminum]; [T = 11]; [F = Medium]}

Step 2: Form array expression T_EXP from SETAV

1. IF [Heat Treatment = Yes] THEN [Route = R1], H = 0.07102

2. IF [Material = Aluminum] THEN [Rote = R3], H = 0.014384

3. IF [Tolerance = 7] THEN [Route = R1], H = 0.015947

4. IF [Finish = Medium] THEN [Route = R3], H = 0.011415

Step 3: The new PRSET

1. IF [Heat Treatment = Yes] THEN [Route = R1], H = 0.07102

2. IF [Material = Steel_1045] THEN [Route = R1]; H = 0.07102

3. IF [Heat Treatment = No] THEN [Route = R3], H = 0.056521

4. IF [Finish = Medium] THEN [Rote = R1], H = 0.022552

Step 4: The expression having the highest H measure covers more than one class. Thus, the process proceeds to step 5 and then continue to step 1.

The process repeats until there are no more uncovered attribute-values that have not been marked. The following is the PRSET at that time.

1. IF [Heat Treatment = Yes] THEN [Route = R1], H = 0.07102

2. IF [Material = Steal_1045] THEN [Rote = R1], H = 0.07102

3. IF [Material = Aluminum] THEN [Route = R3], H = 0.014384

4. IF [Finish = Medium] THEN [Rote = R3], H = 0.011415

The potential attribute-value is a condition in the expression having the highest H measure. It is [Heat Treatment = Yes]

Step 6: Form a new array set SETAV by combining potential attribute-value with other attribute-value. Following is a new SETAV

SETAV = {[Heat Treatment = Yes] and [Material = Steel_1045], [Heat Treatment = Yes] and [Tolerance = 9], [Heat Treatment = Yes] and [Finish = Medium]}.

Step 2,3: A new PRSET is formed as follows:

1: IF [Heat Treatment = Yes] and [Material = Steel_1045] THEN [Route = R1]; H =0.489167

2: IF [Heat Treatment = Yes] and [Tolerance = 9] THEN [Route = R1]; H =0.489167

3: [Heat Treatment = Yes] and [Finish = Medium] THEN [Route = R1]; H =0.345893

Step 4: A rule is produced as the first expression in the PRSET applies to only one class. Therefore, [Heat Treatment=Yes] and [Material=Steel_1045] can be a seed attribute-value.

NewRule : IF [Heat Treatment = Yes] and [Material = Steel_1045] THEN [Route = R1],

H =0.489167

Step 6: Form a set of conjunctions of conditions SETCC = {[Heat Treatment = Yes] and [Material = Steel_1045] and [7≤Tolerance ≤9], [Heat Treatment = Yes] and [Material = Steel_1045] and [Finish = medium]}

Steps 7, 8: Form array rule Temporary Rule Set (T_RSET) and NRSET then as follows:

1. IF [Heat Treatment = Yes] and [Material = Steel_1045] and [Finish = Medium] THEN [Route = 1], H = 0.489167.

2. IF [Heat Treatment = Yes] and [Material = Steel_1045] and [7 ≤ Tolerance ≤ 9] THEN [Route = 1], H = 0.022552.

Step 9: The expression 1 has the highest H measure but it covers 2 examples that are not higher than the number of examples covered by NewRule. Thus, NewRule can be added to RuleSet.

Rule 4: IF [Heat Treatment = Yes] and [Material = Steel_1045] THEN [Route = R1],

H =0.489167

Rule 4 covers examples 6 and 10.

The process continues to *Step 5*.

The expression having the highest H measure in the PRSET at that time is

IF [Material=Aluminum] THEN [Route = R3], H = 0.014384

The potential attribute-value is [Material = Aluminum] and goes to *Step 6*

Step 6: Form a new array SETAV by combining the potential attribute-value with another attribute-value. SETAV = {[Material = Aluminum] and [Heat Treatment = No], [Material = Aluminum] and [9 ≤ Tolerance ≤ 10], [Material = Aluminum] and [Finish = Medium]}

Steps 2,3,4: The process finds out a seed attribute-value [Material = Aluminum] and [Finish = Medium].

With this seed attribute value, the process continues some steps as in forming Rule 4 above. The outcome is a new rule as follows and is added to *RuleSet:*

Rule 5: IF [Material = Aluminum] and [Finish = Medium] THEN [Route = R3],

H = 0.277843

Rule 5 covers examples 2 and 7.

There are no remaining unclassified examples in the dataset, the procedure stops at that time. The final RuleSet includes 5 rules (Rule 1, Rule 2, Rule 3, Rule 4, and Rule 5)

3.3 Rule Simplification

Like predecessors in RULES family and other inductive learning algorithms, the presence of noisy data is also a problem in the learning process. In dealing with noisy data so far, the pruning technique is a standard method that can avoid overfitting the training data set as well as reducing the error and complexity of induced models.

Learning in the presence of noisy data sometimes leads to the output of learning that comprises a large number of rules with low coverage. To address these drawbacks, Bigot (2004) presented the post-pruning technique that creates a new rule by merging two existing rules 1 (R1) and rule 2 (R2) covering examples from the same class.

By this way, if a condition exists for a particular attribute in two existing rules, R1 and R2, a new rule is created by merging them. The merging of conditions between R1 and R2 is performed by carrying out the following operations:

Continuous attribute: If a condition exists for a particular attribute in both rules $[V_{\min_{R1}^{i}} \le A^i \le V_{\max_{R1}^{i}}]$ *and* $[V_{\min_{R2}^{i}} \le A^i \le V_{\max_{R2}^{i}}]$. The new condition in the new rule is

$$\left[V_{\min_{new-rule}^{i}} \le A^i \le V_{\max_{new-rule}^{i}}\right]$$

Where

$$V_{\min^i_{new-rule}} = \min\left[V_{\min^i_{R1}}, V_{\min^i_{R2}}\right]$$
$$V_{\max^i_{new-rule}} = \max\left[V_{\max^i_{R1}}, V_{\max^i_{R2}}\right]$$

Otherwise, no condition will be formed for this attribute in the newly formed rule.

Discrete attributes: If a condition exists for a particular attribute in both rules and if the attribute values in the conditions of both rules are the same ($V^i_{R1} = V^i_{R2}$), then the same attribute value is used to form a condition in the newly formed rule. Otherwise, no condition will be formed for this attribute in the newly formed rule.

As one can see, Dyna algorithm is an efficient rule simplification method. However, during the merging process, all rules are used to form a new rule have been removed from rule set. Thus, the attempt to optimise a given rule set for more general ones may be trapped at local optimal solution. For instance, a rule set (RSet) shown in Figure 3.5 contains 8 rules. RSET = {Rule 1, Rule 2, Rule 3, Rule 4, Rule 5, Rule 6, Rule 7, Rule 8}. The noise level is set to 10% (Th =0.9)

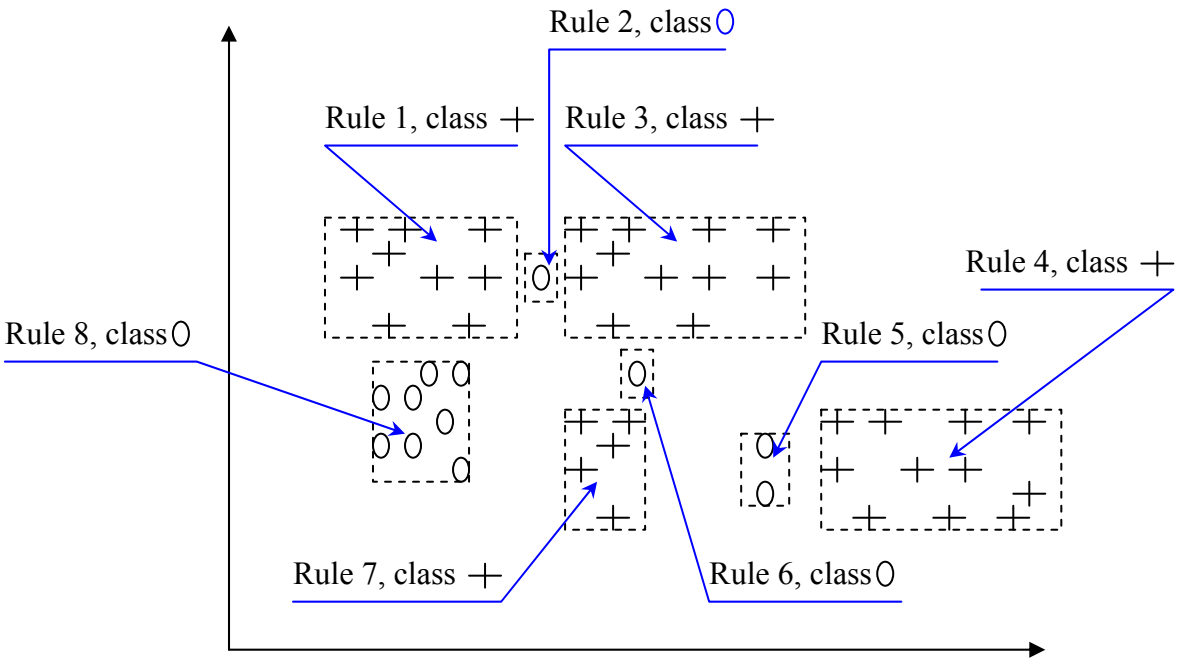


Figure 3.5: Before Pruning, 8 Rules, consistency = 1.00

The Basic Post-Pruning (BPP) (Bigot, 2004) procedure is applied, beginning with Rule 1 as the first Rule-to-be-Merged (R2M) covering examples belonging to class +.

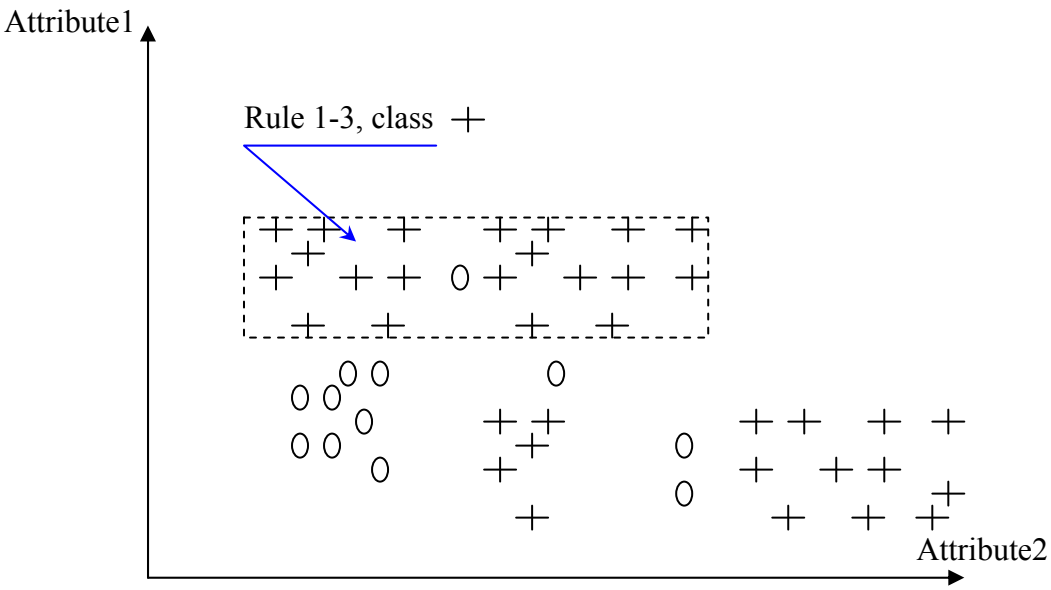


Figure 3.6: Rule 1 merged with Rule 3, consistency = 0.95

The result of merging Rule 1 with another rule in the same class in the PSet is Rule 1-3 (Rule 1 merged with Rule 3) having the highest consistency measure (0.95 > Th) (Figure 3.6). As the procedure in the Dyna Algorithm Rule 1-3 is added to the RSet, and Rule 1 and Rule 3 are removed at the same time. By repeating the process above, Dyna Algorithm will result in the Final-RSet including 7 rules {Rule1-3, Rule 4, Rule 5, Rule 6, Rule 7, Rule 8}

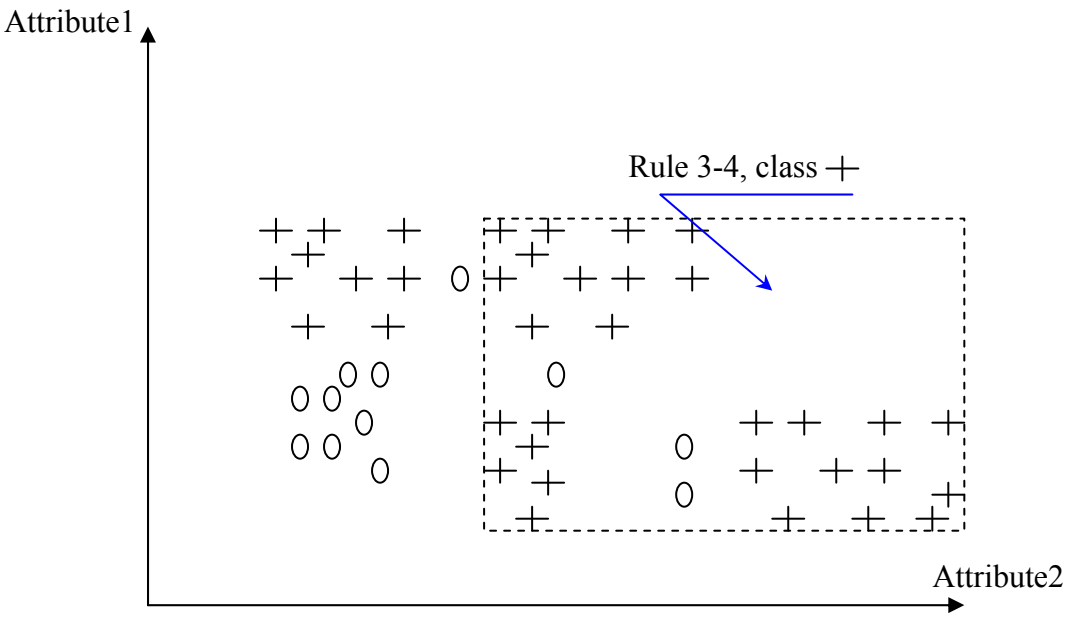


Figure 3.7: Rule 3 merged with Rule 4, consistency = 0.903

With Dyna algorithm, the rule set has reduced the number of rules to six rules instead of eight rules. However, a merging between Rule 3 and Rule 4 can be employed (Figure 3.7). Therefore, the rule set will include three rules instead of six rules as employed by Dyna.

In order to address the above-mentioned issues, both post-pruning techniques in Dyna algorithm will be developed. These techniques are described in detail in the next sections.

3.3.1 Proposed pruning technique development

3.3.1.2 Basic post pruning (BPP) Improvement

The new pruning technique optimises the process of rules simplification by only marking rules that are used to form new rules instead of removing them like Dyna algorithm has employed. However, this technique is also applied after a rule set has been formed and used for the entire rule set (RSet). To avoid being trapped at the local optimal solution, two temporary sets are set up, called the New-RSet (a set of rules in the same class with a rule being taken to be merged) and the T-RSet (a set of rules used only for merging with R2M). By this way, the New-RSet and the RSet are stopping criteria for the algorithm. The complete BPP procedure is presented below:

Step 1: Initialise New-RSET (empty list), T-RSET (empty list), Best-RSET (empty list)
WHILE there is a rule in RSet DO
Step 2 Take an R in RSet and remove all rules in the same class with R to New-RSET
T-RSET = New-RSET
WHILE there is a rule in New-RSET DO
R2M is the first rule in New-RSET
Step 3 Merge R2M with each rule in T-RSET and compute the consistency of the newly formed rule
new-rule = the rule with the highest consistency measure resulting from the merging
IF the new-rule consistency measure ≥ Th THEN
- Remove all rules used for its formation from New-RSET
- Add the new-rule to T-RSET and New-RSet at the same time
ELSE - Add R2M to Best-RSET
- Remove R2M from New-RSET
AND WHILE
AND WHILE

Step 4 Initialise Final-RSet (empty list)

WHILE there are examples not classified by Final-RSet DO

Step 5 new_rule = the rule in Best-RSet covering the largest number of

examples not covered by Final-RSet formed so far

Add new_rule to Final-RSet

END WHILE

Figure 3.8: The Basic Post-Pruning Procedure (BPP)

3.3.1. Incremental Post-Pruning (IPP) Improvement

In contrast to the BPP algorithm, whenever a new rule is formed, the IPP techniques will be started. The new IPP is different from its predecessor technique in that all rules used for merging new rule are also marked instead of being removed from RSet. Moreover, during creating Final-RSet, only unmarked rules in RSet are used. The developed IPP technique is described as follows.

WHILE there is an uncovered example DO

Best_Rule = a new rule is formed

IF Th = 1 THEN Stop

ELSE

R2M = Best_Rule

Step 1: Merge R2M with each rule for the same class in RSet

new_rule = the rule with the highest consistency measure resulting from the mergers

Step 2: IF new_rule has a consistency measure $\geq$ Th THEN

Marks all rules used for its formation from RSet

Add new_rule into RSet

R2M = new_rule

Go back to *Step*

ELSE Add R2M into RSet

END WHILE

Step 3: Initialise Final-RSet (empty list)

WHILE there are examples uncovered by the Final-RSet DO

Step 4: new_rule = the unmarked rule in RSet covering the largest number of examples that are still not covered by the Final-RSet formed so far

Add new_rule to Final-RSet

END WHILE

Figure 3.9: The Incremental Post-Pruning Procedure (IPP)

3.3.2 Illustrative problem

In order to illustrate the pruning process carried out by the algorithm given above, the new BPP procedure is applied to the rule set (RSet) shown in Figure 3.5. The noise level is set to 10% (Th = 0.9) for this illustrative problem.

Step 1: Initialise new rule sets New-RSET = { }, T-RSET = { }, Best-RSET = { }. RSet contains 8 Rules, RSet = {Rule1, Rule 2, Rule 3, Rule 4, Rule 5, Rule 6, Rule 7, Rule 8} The pruning procedure starts.

Step 2: Rule1 covering example belonging to class + is taken to be pruned

New-RSet = {Rule1, Rule 3, Rule 4, Rule 7};

T-RSet = {Rule1, Rule 3, Rule 4, Rule 7};

The first R2M is Rule 1

Step 3: The result of merging R2M with Rule 3 (Rule 1-3), Rule 4 (Rule 1-4) and Rule 7 (Rule 1-7) are shown in Figure 3.6, Figure 3.8 and Figure 3.9 respectively. The best rule, with the highest consistency, resulting from this merged is Rule1-3. Thus, new-rule = Rule 1-3

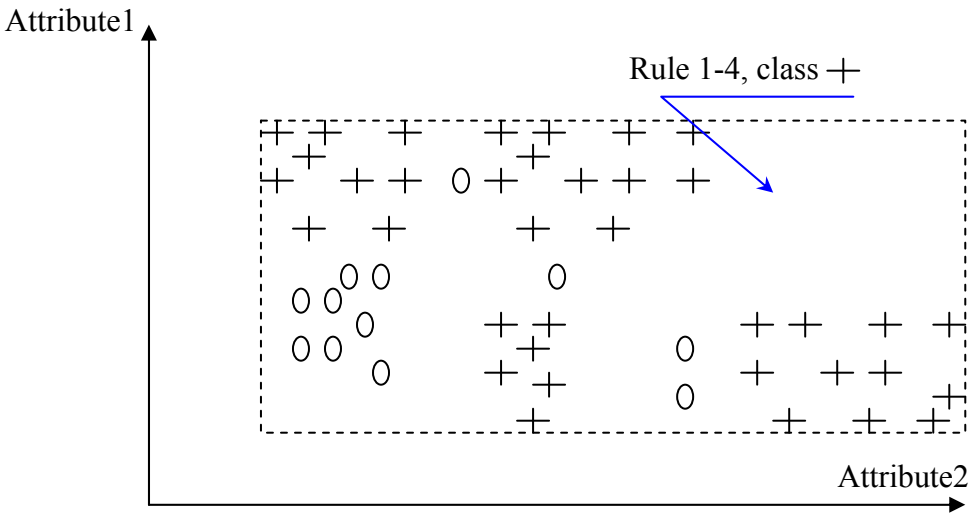


Figure 3.10: Rule 1 merged with Rule 4, consistency = 0.76

The consistency of new-rule is higher than Th (consistency = 0.95, Th = 0.9). Therefore Rule1 and Rule 3 are removed from New-RSet, and new-rule is added to T-RSet and New-RSet

New-RSet = {Rule 4, Rule 7, Rule1-3; *T-RSet* = {Rule 1, Rule 3, Rule 4, Rule 7, Rule1-3}

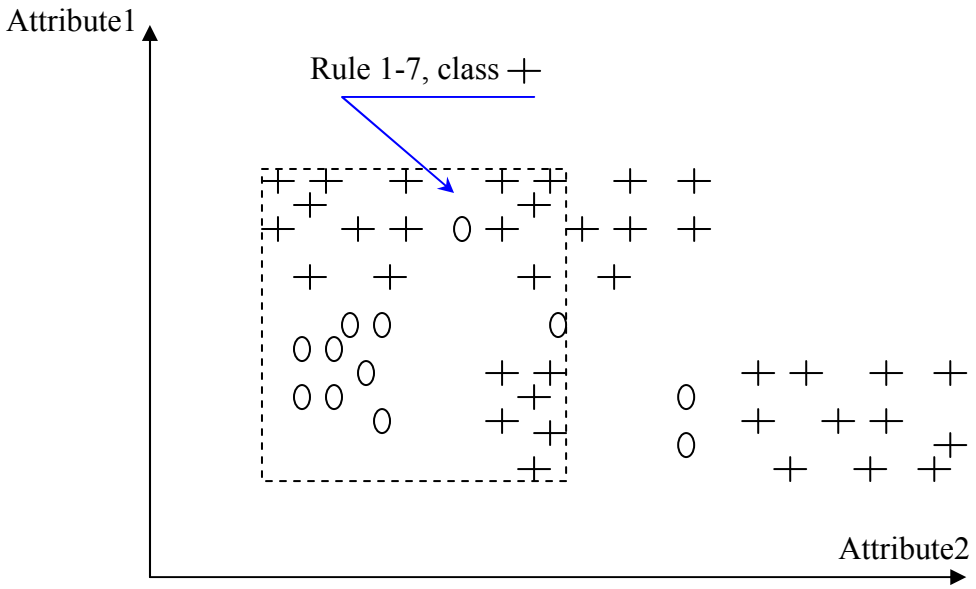


Figure 3.11: Rule 1 merged with Rule 7, consistency = 0.67

There are still rules in New-RSet so the algorithm goes back to step 2. R2M = Rule 4.

Step 3: R2M can be merged with other rules in T-RSet, resulting from that merging are Rule 4-1 (Merging Rule 4 with Rule 1), Rule 4-3 (merging Rule 4 with Rule 3), Rule 4-7 (merging Rule 4 with Rule 7) and Rule 4-1-3 (merging Rule 4 with Rule 1-3) that shown in Figure 3.7, Figure 3.10, Figure 3.11 and Figure 3.7 respectively.

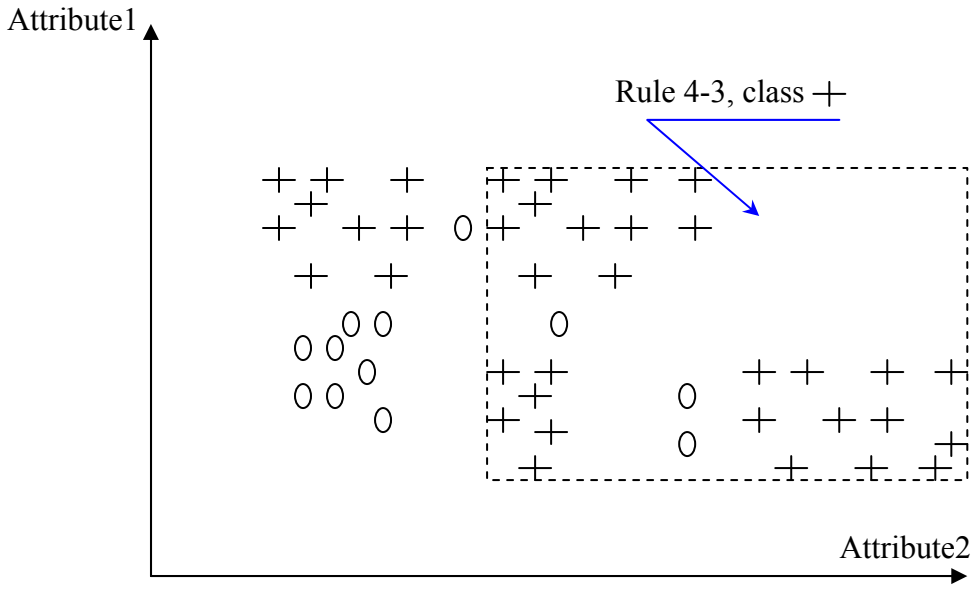


Figure 3.12: Rule 4 merged with Rule 3, consistency = 0.903

The best rule, with the highest consistency, resulting from this merging is Rule 4-3. New-rule = Rule 4-3. The consistency of new-rule is higher than Th (consistency = 0.903, Th = 0.9) therefore Rule 4 is removed from New-RSet and new-rule is added to T-RSet and New-RSet at the same time.

T-RSet = {Rule 1, Rule 3, Rule 4, Rule 7, Rule1-3, Rule 4-3}

New-RSet = (Rule7, Rule 1-3, Rule 4-3}

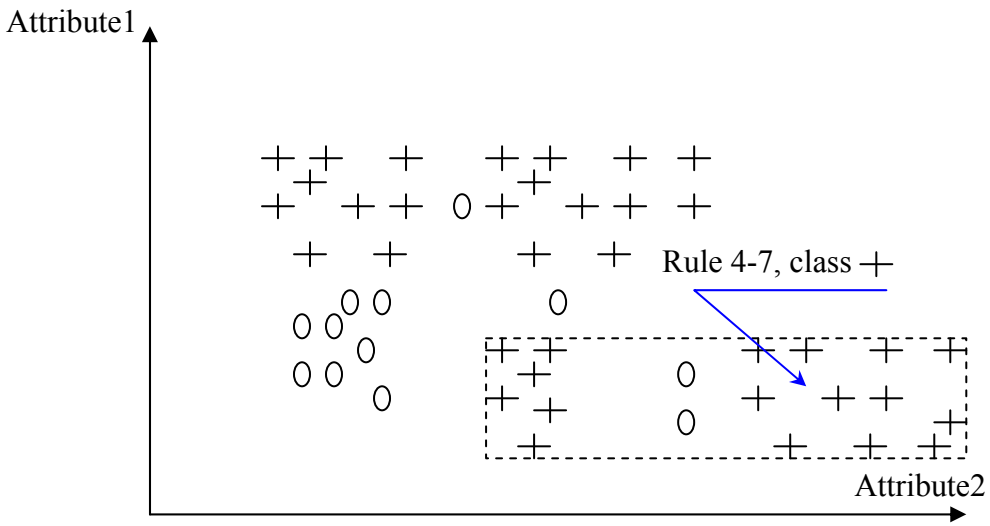


Figure 3.13: Rule 4 merged with Rule 7, consistency = 0.894

There are still rules in *New-RSet* and the algorithm, goes back to step 2 again, R2M = Rule7

Step 3: Rule 7 can be merged with other rules in *T-RSet*, the best rule, with the highest consistency measure, resulting from this merging is Rule 7-4-3 (Rule 7 merged with Rule 4-3) (Figure 3.12), new-rule = Rule 7-4-3 (consistency = 0.903)

The consistency of new-rule is higher than Th (consistency = 0.903, Th = 0.9). At that time New-RSet and T-RSet are revised as below

New-RSet = {Rule 1-3, Rule 7-4-3)

T-RSet = (Rule 1, Rule 3, Rule 4, Rule 7, Rule 1-3, Rule 4-3, Rule 7-4-3}

There are still rules in New-RSet so the algorithm continuously goes back to step 2, R2M = Rule 1-3.

Step 3: Similar with the previous steps, the best rule, with the highest consistency measure, resulting from merging R2M with other rules in T-RSet is Rule 1-3-4-7 (Rule 1-3 merged with Rule 4-7) (Figure 3.13), new-rule = Rule 1-3-4-7 (consistency = 0.76)

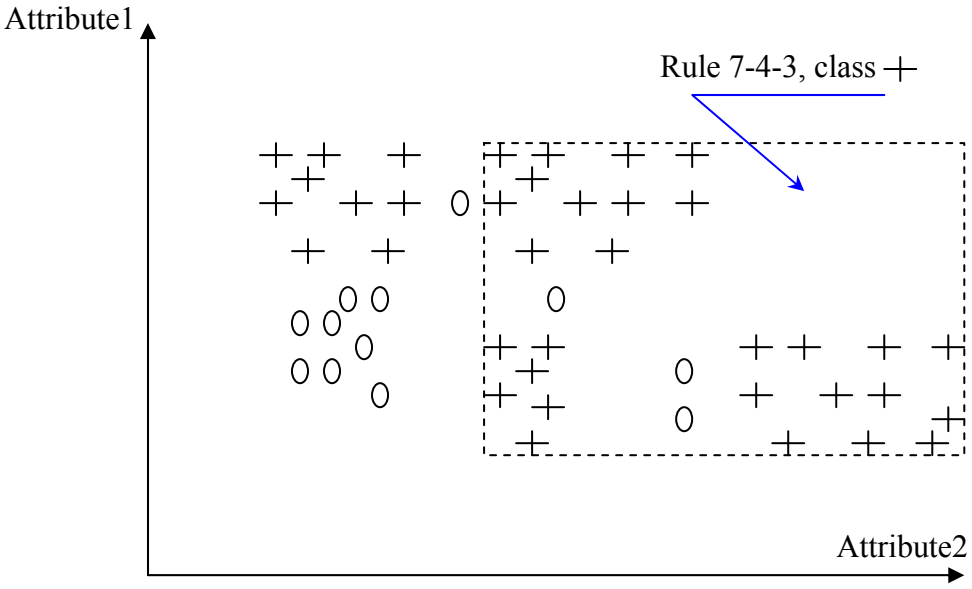


Figure 3.14: Rule 7 merged with Rule 4-3, consistency = 0.903

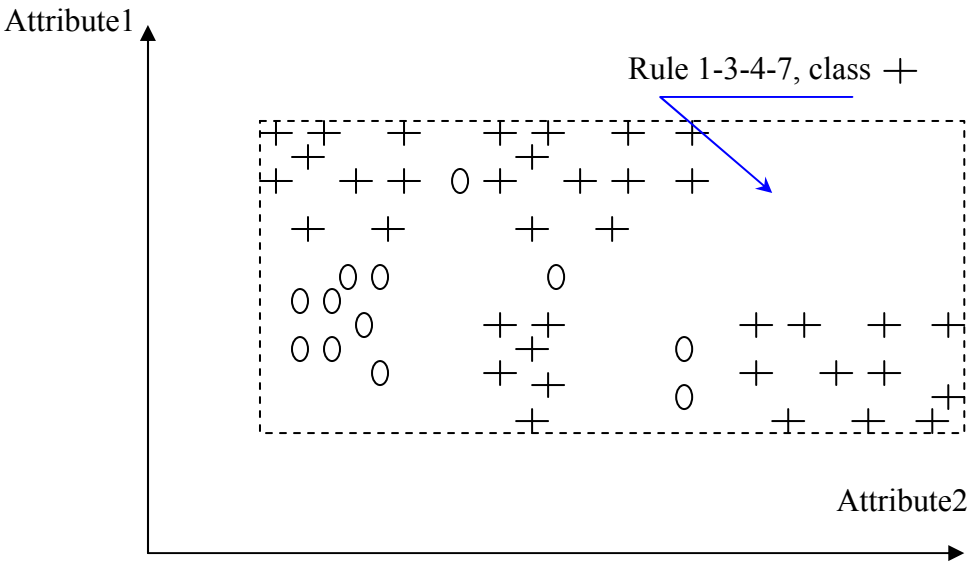


Figure 3.15: Rule 3-1 merged with Rule 4-7, consistency = 0.76

The consistency of new-rule is lower than Th (consistency = 0.76, Th = 0.9) and therefore R2M is stored in Best-RSet and Rule 1-3 is removed from New-RSet.

New-RSet = {Rule 7-4-3}

T-RSet = {Rule 1, Rule 3, Rule 4, Rule 7, Rule 1-3, Rule 4-7, Rule 7-4-3}

Best-RSet = {Rule 1-3}

RSet = {Rule 2, rule 5, Rule 6}

By repeating these steps until New-RSet and then RSet are empty, Best-RSet is formed (Figure 3.14). Best-RSet = {Rule 1-3, Rule 2, Rule 7-4-3, Rule 5, Rule 6, Rule 8}

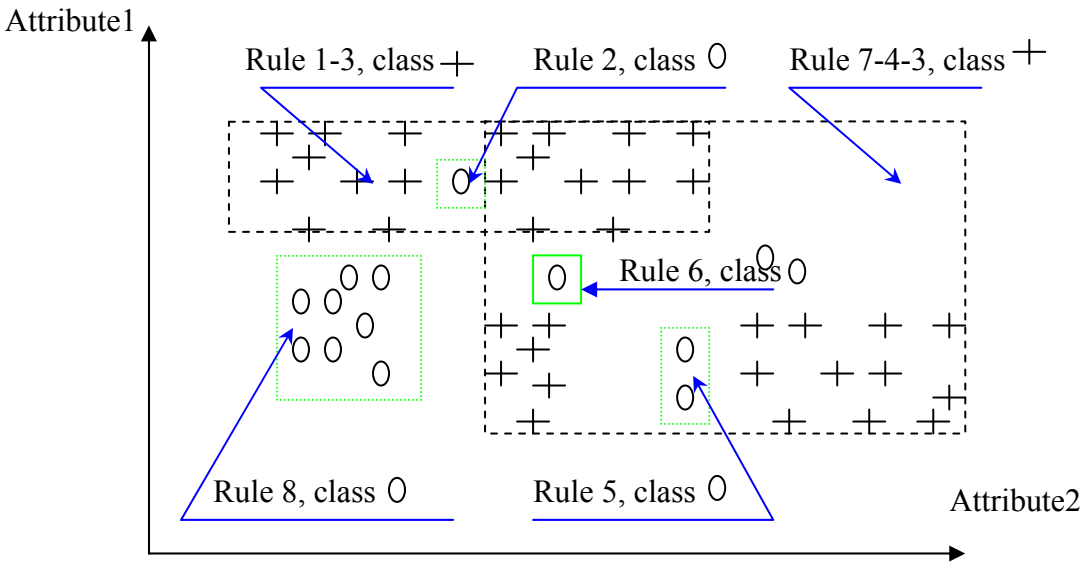


Figure 3.16: Best-RSet, 6 Rules

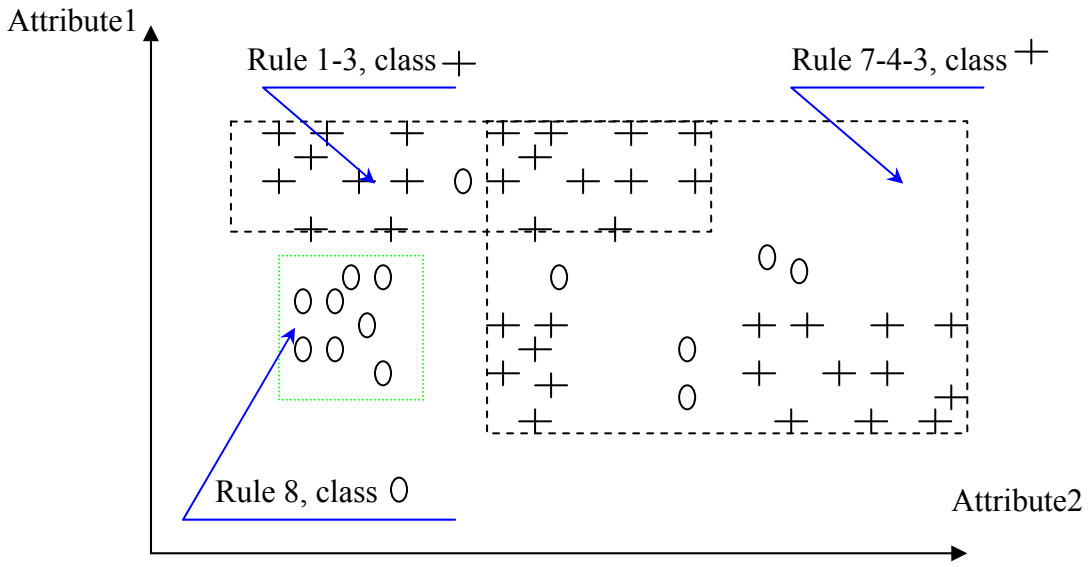


Figure 3.17: Final-RSet, with speciafied NL = 10% (Th = 0.9)

At that time Final-RSet is initialised. No examples are covered by Final-RSet; Final-RSet = {}. Like Dyna Algorithm, the rule covering the highest number of uncovered examples is selected and stored in Final-RSet

Final_RSet = {Rule 7-4-3}

The process is repeated until all examples are covered by Final-RSet. At the end of the process Final_RSet includes the following rules (Figure 3.15):

Final_RSet = {Rule 7-4-3, Rule 1-3, Rule 8}

3.4 RULES-8 classification technique

Like its predecessors, the new inductive algorithm creates a set of rules whose coverage is limited to the training examples only. Therefore, when a set of rules is used as a classification model, there are three possible outcomes as follow:

The first possibility is only one rule covers the new example. The class classified by that rule is assigned to the new example.

The second possible outcome is more than one rule covers the example. In this case, the rule having the highest H measure will be used to classify the example.

The third possible outcome is no rules can classify the example. A specialised process is employed by applying the following rule:

==========

Procedure ***NoRuleCover***

Step 1: Find the closest rule (Bigot, 2004).

Step 2: Consider the new example a noisy one of the class predicted by the closest rule and then compute the noise level (NL) in this class.

Step 3: **IF** noise level < noise threshold

THEN the closest rule is used to classify the new example

ELSE the new example should be added to the training set

and the learning process reinitiated.

==========

The new covering algorithm uses the distance between a rule R and an example E to find the closest rule, this measure is also defined as:

$$\text{Distance R/E} = \sqrt{\sum_{c} c\_distance + \sum_{d} d\_distance} \qquad (3.3)$$

Where $\sum_{c}$ is the sum for continuous attributes;

$\sum_{d}$ is the sum for discrete attributes;

c_distance is defined for each continuous attribute as follows

If the value is outside the condition range of the attribute

$$c\_distance = \left(\frac{\min\left(\left| V_E^i - V\max_R^i \right|, \left| V_E^i - V\min_P^i \right| \right)}{V_{\max}^i - V_{\min}^i} \right)^2 \qquad (3.4)$$

Else $c\_distance = 0$

d_distance is defined for each discrete attribute by applying the following rule

IF attr_value = Rule attribute value

Then d_distace = 0

Else d_distance =1

3.5 Missing attribute values

As indicated so far, learning in the presence of missing attribute values needs to be considered carefully. Correspondingly, a number of solutions have been developed to overcome this problem. For instance, when an example was recognised containing missing attribute value, the following techniques can be applied (Quinlan, 1989):

- Ignore this example;
- Treat the example as though it has the most common value of the attribute
- Consider the unknown value as a separate value for the attribute.

These techniques have been developed and applied successfully in family rule algorithm (Pham et al., 1997). For the new inductive algorithm, these techniques above are also used but with modification. In particular, the following procedures are implemented:

When the set of attributes and values (SETAV) is constructed, the number of elements is equal to the number of attributes. The elements of SETAV are attribute-values or a

combination between potential attribute-values and other attribute-values in the same example. Therefore, the example that has missing attribute-values and a combination of attribute-values leads to vacant elements in SETAV. In this case, the expressions from T-EXP with no conditions that are considered the lowest H measure will be the last to be considered in the specialisation process.

3.6 Test and analysis of result

3.6.1 RULES-8 without pruning

RULES-8 has been tested on 20 data sets that are commonly used to benchmark inductive learning algorithm. The test results are presented in Table 3.2.

In comparison to Rules 3 Plus and Dyna algorithm, RULES-8 generates more compact rule sets. At the same time, higher classification accuracy is reached. Hence, the specialisation method used in RULES-8 is more efficient than those used in Rules 3 Plus and Dyna. Remarkably, these improvements occur not only with data sets containing continuous attributes but also with data sets with discrete attributes or combinations of both types.

Table 3.2 Test results with comparison between Rules 3 plus, Dyna and RULES-8

Data set name	Rules3 plus Results			Dyna Results			RULES-8 Results		
	Rules	Training %	Test %	Rules	Training %	Test %	Rules	Training %	Test %
Balance_scale	207	87.2204	75.0987	206	100	68.6901	195	100	72.2045
breast_cancer	33	99.4286	93.0626	12	100	93.1232	12	100	94.8424
wdbc	17	100	89.0819	6	100	89.0845	6	100	94.0141
wpbc	32	100	57.5829	8	100	60.6061	8	100	78.7879
car	169	100	87.1538	145	100	89.3519	139	100	90.8565
credit screening	101	97.1014	74.2029	35	100	76.5896	34	100	81.2139
dermatology	39	100	89.6210	22	100	81.9672	21	100	87.9781
diabetes	122	74.2188	65.1027	43	100	68.4896	39	100	73.6971
glass	23	98.1308	89.7232	9	100	99.0654	9	100	99.0654
haberman	16	65.3595	57.5232	36	98.04	67.3203	34	100	90.8497
iris	5	97.3333	93.3343	5	100	96	5	100	96
liver	37	65.3179	50.5812	28	100	88.9535	27	100	92.4419
Tic Tac Toe	133	100	89.7717	24	100	98.3299	24	100	98.3299
Ecoli	38	83.9286	76.7828	23	100	100	23	100	100
Teaching	30	60.5263	51.3242	30	97.37	65.3333	29	100	86.6667

3.6.2 RULES-8 with Pruning technique

The performance of RULES-8 with BPP is compared against that of RULES-8 without any pruning. The test results presented in Table 3.3 show a significant reduction of the number of rules generated when pruning was performed. These reductions in the rule sets' levels of accuracy are acceptable in most cases because the resulted rule sets are more compact and more capable to handle noisy data.

Table 3.3 Comparison between RULES-8 without pruning and RULES-8 with BPP

Data set name	RULES-8 without pruning			NL	RULES-8 with BPP		
	Rules	Training	Test %		Rules	Training	Test %
Balance_scale	195	100	72.2045	0.10	133	97.7636	85.3035
				0.20	84	92.3323	83.0671
				0.30	46	85.9425	83.3866
Breast_cancer	12	100	94.8424	0.10	2	93.1624	93.9828
				0.20	2	93.1624	94.2693
				0.30	2	93.1624	94.2693
wdbc	6	100	94.0141	0.10	4	95.7895	96.1268
				0.20	4	95.7895	85.5634
				0.30	4	95.7895	84.8592
wpbc	8	100	78.7879	0.10	8	100	81.8182
				0.20	6	95.9596	76.7677
				0.30	3	83.8384	72.7273
Car	139	100	90.8565	0.10	136	99.0741	90.3935
				0.20	82	92.1296	85.5324
				0.30	11	85.5324	70.7176
Credit	34	100	81.2139	0.10	18	95.0725	85.8382
				0.20	5	86.9565	83.5260
				0.30	3	85.7971	85.2601
Dermatology	21	100	87.9781	0.10	21	100	89.6175
				0.20	21	100	89.6175
				0.30	21	100	83.6066
Diabetes	39	100	73.6979	0.10	36	98.6979	75.7813
				0.20	25	90.8854	74.7396
				0.30	10	80.4688	67.4479
Glass	9	100	99.0654	0.10	6	100	99.0654
				0.20	6	100	99.0654
				0.30	6	100	99.0654
Haberman	34	100	90.8497	0.10	26	96.7320	97.3856
				0.20	10	86.2745	77.7778
				0.30	4	83.6601	73.2026
Iris	5	100	96	0.10	4	97.3333	93.3333
				0.20	4	97.3333	93.3333
				0.30	4	97.3333	93.3333
Liver	27	100	92.4419	0.10	28	100	71.5116
				0.20	24	98.8439	67.3299
				0.30	16	95.3757	61.6279
Tic Tac Toe	24	100	98.3299	0.10	24	100	98.3299
				0.20	24	100	98.3299
				0.30	16	92.2756	92.0668
Ecoli	23	100	100	0.10	14	96.4497	84.5238
				0.20	11	92.3077	76.7857
				0.30	9	91.1243	74.4048
Teaching	29	100	86.6667	0.10	29	97.3684	72
				0.20	29	97.3684	72
				0.30	19	93.4211	68

RULES-8 algorithm has applied the two new rule pruning procedures as described earlier. Tests have been carried out on the previous 15 data sets. Table 3.4 displays test results applying BPP and IPP. It can be seen that both methods yield very similar results in terms of the number of rules formed and their accuracy.

Table 3.4 Comparison between BPP and IPP

Data set name	NL	RULES-8 with BPP			RULES-8 with IPP		
		Rules	Training%	Test %	Rules	Training	Test %
Balance_scale	0.10	133	97.7636	85.3035	135	98.4026	86.5851
	0.20	84	92.3323	83.0671	87	94.2492	84.6645
	0.30	46	85.9425	83.3866	48	87.5399	84.0256
Breast_cancer	0.10	2	93.1624	93.9828	6	96.5812	95.1289
	0.20	2	93.1624	94.2693	4	94.5869	94.8424
	0.30	2	93.1624	94.2693	2	93.1624	94.2693
wdbc	0.10	4	95.7895	96.1268	4	95.7895	96.1268
	0.20	4	95.7895	85.5634	4	95.7895	85.5634
	0.30	4	95.7895	84.8592	4	95.7895	84.8592
wpbc	0.10	8	100	81.8182	8	100	81.8182
	0.20	6	95.9596	76.7677	6	95.9596	76.7677
	0.30	3	83.8384	72.7273	3	83.8384	72.7273
Car	0.10	136	99.0741	90.3935	138	99.6528	91.3194
	0.20	82	92.1296	85.5324	84	93.8657	86.2269
	0.30	11	85.5324	70.7176	13	87.9630	72.8009
Credit	0.10	18	95.0725	85.8382	16	95.3623	87.5723
	0.20	5	86.9565	83.5260	7	90.1449	85.8382
	0.30	3	85.7971	85.2601	4	87.5362	86.1272
Dermatology	0.10	21	100	89.6175	21	100	89.6175
	0.20	21	100	89.6175	21	100	89.6175
	0.30	21	100	83.6066	21	100	83.6066
Diabetes	0.10	36	98.6979	75.7813	37	99.2188	77.3438
	0.20	25	90.8854	74.7396	26	91.9271	76.5625
	0.30	10	80.4688	67.4479	12	81.5104	70.0521
Glass	0.10	6	100	99.0654	6	100	99.0654
	0.20	6	100	99.0654	6	100	99.0654
	0.30	6	100	99.0654	6	100	99.0654
Haberman	0.10	26	96.7320	97.3856	26	96.7320	97.3806
	0.20	10	86.2745	77.7778	14	92.8105	86.2745
	0.30	4	83.6601	73.2026	6	85.6209	79.0850
Iris	0.10	4	97.3333	93.3333	4	97.3333	93.3333
	0.20	4	97.3333	93.3333	4	97.3333	93.3333
	0.30	4	97.3333	93.3333	4	97.3333	93.3333
Lever	0.10	28	100	71.5116	28	100	71.5116
	0.20	24	98.8439	67.3299	24	99.4220	69.1860
	0.30	16	95.3757	61.6279	20	97.6879	65.6977
Tic Tac Toe	0.10	24	100	98.3299	24	100	98.3299
	0.20	24	100	98.3299	20	100	96.2422
	0.30	16	92.2756	92.0668	16	92.2756	92.0668
Ecoli	0.10	14	96.4497	84.5238	15	96.4497	89.8810
	0.20	11	92.3077	76.7857	13	92.3077	80.3571
	0.30	9	91.1243	74.4048	9	91.1243	74.0448
Teaching	0.10	29	97.3684	72	27	96.0526	73.3333
	0.20	29	97.3684	72	25	94.7368	70.6667
	0.30	19	93.4211	68	19	93.4211	68

The results of the comparison between Dyna and RULES-8 with BPP are displayed in Table 3.5. It can be seen that for the majority of data, RULES-8 with BPP generated fewer

rules as compared against the number of rules generated by Dyna with BPP. However, a slight decrease in terms of the test accuracy is also seen.

Table 3.5 Comparison between Dyna with BPP and RULES-8 with BPP

Data set name	NL	Dyna with BPP			RULES-8 with BPP		
		Rules	Training%	Test %	Rules	Training	Test %
Balance_scale	0.10	139	96.8051	79.3443	133	97.7636	85.3035
	0.20	85	90.7348	83.2786	84	92.3323	83.0671
	0.30	51	85.9425	83.6066	46	85.9425	83.3866
Breast_cancer	0.10	2	93.1429	94.0972	2	93.1624	93.9828
	0.20	2	93.1429	94.0972	2	93.1624	94.2693
	0.30	2	93.1429	94.0972	2	93.1624	94.2693
wdbc	0.10	4	96	84.8591	4	95.7895	96.1268
	0.20	4	96	84.8591	4	95.7895	85.5634
	0.30	4	96	84.8591	4	95.7895	84.8592
wpbc	0.10	8	100	74.7474	8	100	81.8182
	0.20	6	95.9596	73.7374	6	95.9596	76.7677
	0.30	1	79.798	72.7273	3	83.8384	72.7273
Car	0.10	145	100	90.3936	136	99.0741	90.3935
	0.20	98	92.1296	84.9536	82	92.1296	85.5324
	0.30	1	70.9491	68.9815	11	85.5324	70.7176
Credit	0.10	20	94.2029	84.9245	18	95.0725	85.8382
	0.20	5	86.9565	83.7681	5	86.9565	83.5260
	0.30	3	85.7971	85.2174	3	85.7971	85.2601
Dermatology	0.10	21	100	83.0601	21	100	89.6175
	0.20	21	100	83.0601	21	100	89.6175
	0.30	21	100	83.0601	21	100	83.6066
Diabetes	0.10	38	98.4375	73.6980	36	98.6979	75.7813
	0.20	25	90.8854	73.6977	25	90.8854	74.7396
	0.30	10	80.4688	67.1874	10	80.4688	67.4479
Glass	0.10	6	100	99.0654	6	100	99.0654
	0.20	6	100	99.0654	6	100	99.0654
	0.30	6	100	99.0654	6	100	99.0654
Haberman	0.10	29	97.3856	67.3203	26	96.7320	97.3856
	0.20	10	86.2745	73.2025	10	86.2745	77.7778
	0.30	1	75.1624	71.8954	4	83.6601	73.2026
Iris	0.10	4	97.3333	93.3333	4	97.3333	93.3333
	0.20	4	97.3333	93.3333	4	97.3333	93.3333
	0.30	4	97.3333	93.3333	4	97.3333	93.3333
Lever	0.10	28	100	67.4419	28	100	71.5116
	0.20	24	98.8439	66.2791	24	98.8439	67.3299
	0.30	16	92.4855	61.6279	16	95.3757	61.6279
Tic Tac Toe	0.10	24	100	98.3298	24	100	98.3299
	0.20	24	100	98.3298	24	100	98.3299
	0.30	14	92.2756	92.0668	16	92.2756	92.0668
Ecoli	0.10	15	96.4268	79.1667	14	96.4497	84.5238
	0.20	11	92.2619	75.5952	11	92.3077	76.7857
	0.30	9	91.0714	74.4048	9	91.1243	74.4048
Teaching	0.10	29	97.3684	65.7894	29	97.3684	72
	0.20	29	97.3684	65.7894	29	97.3684	72
	0.30	20	92.1053	67.1052	19	93.4211	68

3.7 Summary

The proposed new inductive learning algorithm, RULES-8, presented in this chapter has been tested on 15 sets of data obtained from the University of California at Irvine (UCI), repository of machine learning databases (Blake and Merz, 1998). It was found that RULES-8 generates fewer rules and produces more accurate rule sets in comparison to other algorithms in the RULES family. However, the selection of the conjunction of conditions was based on a heuristic measure – H measure – which is a measure constructed on experiences. Even though this measure has been successfully applied, the results are not always consistent. In several cases of tested data in this chapter, the results were not as desired. The causes are still yet to be fully understood. Therefore, the next chapter will concentrate on other rule quality measures, analyse precisely their roles within the inductive learning process in order to propose another improvement in rule quality measure.

Chapter 4

AN ENHANCED MEASURE FOR RULE EVALUATION

4.1 Preliminaries

A rule induction system extracts rule sets from a set of training data. A set of rules is used to classify unseen objects. It is therefore important for a rule induction system to generate rules that have high predictability, reliability, likelihood, and so on. These properties are commonly measured by a function called rule quality. In covering methods, rule quality measures have become one of the central for many reasons.

A function evaluating rules is not only needed in the rule induction process but is also necessary in the classification process. In the rule induction process, rule quality measures are devised to evaluate the information content of the newly formed conjunctions of conditions resulting from the specialisation process. Hence, rule quality measures are necessary to select the best conjunction of conditions for further specialisation at any stage of the rule forming process (specialisation heuristics). In addition, rule quality measures can evaluate the conjunctions formed so far and determine whether the specialisation process should be stopped (stopping heuristics).

Furthermore, rule quality measures are also required in the classification process. It is possible that an unseen example satisfies several rules that are assigned to different classes. In such a case, a heuristic measure is necessary to select the most appropriate class.

Many different heuristic measures have been applied in inductive learning to assess the quality of the rules generated. In earlier applied research in inductive learning, statistics methods have been used to evaluate the efficiency of the rule formation procedures. The

numbers of samples that are categorised accurately and those that are classified inaccurately by the newly formed rules are prompts of rule quality. Even though these methods provide valuable information, they are challenged by the quality of the rules formed and also influenced by the learning process. Thus, the results are not always reliable.

Some measures developed so far have been interested only in the accuracy of rules with negligence to the number of examples covered by rules (*Consistency* (Pagallo and Haussler, 1990)). That leads to the formation of rules with higher scores. However, the newly formed rule can only cover only one positive example, whereas a rule that can cover 1000 positive examples and 1 negative example would rather be desired.

Quality of more recently developed measures has been improved. These include the H measure applied in Rules-3 plus (Pham and Dimov, 1996a) and Dyna, Laplace estimate (Clark and Bowell, 1991), Information Gain applying in RIPPER (Cohen, 1995), AQ18 measure (Michalski and Kaufman, 1999)…etc.

Bigot (2004) pointed out that the specialisation heuristics and classification heuristics do not have the same goals, and the information required to assess the quality of conjunctions and rules differs. Bigot (2004) therefore proposed a more efficient heuristic – S measure (to be described in more details later). The S measure has been the most current heuristic measure devised for covering algorithms. Despite its efficiency, this heuristic has some drawbacks since not all information of training data is covered during the specialisation process.

In the next section (Section 4.2), the performance of some recently specialisation heuristics will be examined and a new heuristic that allows the creation of more general rule sets will be proposed. The new heuristic is then evaluated and compared the latest

specialisation heuristic – S measure. Section 4.3 studies the methods used to select between competing rules in the rule sets to classify an unseen example, and a new and more accurate classification method is introduced and tested to compare with its predecessors.

4.2 Specialisation heuristics

Specialisation heuristics evaluate the information content of the newly formed conjunctions of conditions in order to select the best conditions to be added to the conjunctions formed during the specialisation process. The desired result from the application of these metrics is the creation of more general and consistent rules.

4.2.1 Existing heuristics

Most heuristic measures are derived by analysing the relationship between rule R and the target class. Specialisation heuristics are general required to be not only reliable but also powerful. Their reliability is characterised by a consistency factor, and their power is considered by a completeness factor.

Assuming that, a training data set includes P positive examples and N negative examples. Rule R covers p positive examples and n negative examples. The consistency and completeness of a rule R can be defined as follow:

$$Consistency = \frac{p}{p+n} \qquad (4.1)$$

$$Completeness = \frac{p}{P} \qquad (4.2)$$

Consistency was used in GREEDY3 (Pagallo and Haussler, 1990). However, if rule R covers no negative example (n = 0), the value of consistency is one for any value of positive example (p). Therefore, this measure does not provide information to decide which one to choose between two consistent rules.

A modification of consistency measure, the Laplace and m-estimates (Cestnik, 1990) had been also introduced:

$$Laplace\ estimate = \frac{p+1}{p+n+2} \qquad (4.3)$$

$$m-estimates = \frac{p+m\dfrac{P}{P+N}}{p+n+m} \qquad (4.4)$$

The bottom line of these estimates is the assumption that each rule covers a certain number of examples or *a priori*. They compute a precision estimate, but start to count covered positive or negative examples from a number > 0. With the Laplace estimate, both the positive and negative coverage of a rule are initialised with 1 (thus assuming an equal prior distribution), Meanwhile, the m-estimate assumes a prior total coverage of m examples, which are distributed according to the distribution of positive and negative examples in the training set. Due to its simplicity and efficiency, the Laplace estimate has been used in several algorithms. Its advantages over the entropy measure were proved as it was applied in CN2 and gave better results (Clark and Boswell, 1991).

Another heuristic that was proposed using consistency and coverage factors is *AQ18 measure* (Michalski and Kaufman, 1999). AQ18 measure is defined as follows:

AQ18 measure: $$Q(w) = \left(\frac{p}{P}\right)^{w}\left[\left(\frac{p}{p+n}-\frac{P}{P+N}\right)\frac{P+N}{N}\right]^{(1-w)} \qquad (4.5)$$

The value of the parameter w is defined by the user. This setting of *w* has a different influence on coverage (the first part) and consistency gain (the second part). However, due to the discontinuity of the function, the function is negative if $\frac{p}{p+n} < \frac{P}{P+N}$ and positive if $\frac{p}{p+n} > \frac{P}{P+N}$, there are therefore still drawbacks that need to be improved.

Let's consider another measure, which is based on the information content of the conjunctions, as defined by the equation below:

$$Info\_Content = -\log_2\left(\frac{p}{p+n}\right) \qquad (4.6)$$

The best conjunction would be the one that minimises this value. The information gain during the specialisation process is then calculated as the reduction of the information content in a conjunction when a new condition is added to it. In RIPPER (Cohen, 1995), p' and n' are considered the number of positive examples and number of negative examples in turn covered by the original conjunction of condition. A heuristic to measure the information content for the new condition is as below:

$$Info\_Content = p\left(-\log_2\left(\frac{p'}{p'+n'}\right)+\log_2\left(\frac{p}{p+n}\right)\right) \qquad (4.7)$$

Accuracy, another type of heuristics used for forming rules, has been proposed in PROLOG (Muggleton, 1995) and I-REP (Fürnkranz, 1996). In theory, accuracy is the proportion of covered positive examples (p) and uncovered negative examples (N-n) in all examples (P+N).

$$Accuracy = \frac{p+(N-n)}{P+N} \qquad (4.8)$$

Perhaps the most well known heuristic used in inductive learning algorithms has been the H measure. This metric is the product of the accuracy and generality of a conjunction. The H measure is defines as below:

$$H = \sqrt{\frac{p+n}{P+N}}\left[2 - 2\sqrt{\frac{p}{p+n}\frac{P}{P+N}} - 2\sqrt{\left(1-\frac{p}{p+n}\right)\left(1-\frac{P}{P+N}\right)}\right] \qquad (4.9)$$

The H measure was originally introduced in the RULES family of algorithms only to guide the specialisation process. However, it has also been used successfully as a classification heuristic.

As mentioned above, Bigot (2004) introduced a new specialisation heuristic – the S measure for covering algorithms. In this research, two new parameters were used. They are the number of positive examples covered by the newly formed conjunction of conditions and not covered by previously created rules (*p_new*) and the number of examples belonging to the target class and not classified by the rule set formed so far (P_unclassified).

The S measure is computed using the following equation:

$$S = \frac{p}{p+n}\frac{p\_new}{P\_unclassified}\left(1-\frac{n}{N}\right) \qquad (4.10)$$

4.2.2 The S measure and some unsolved problems

The S measure (Bigot, 2003) is the latest heuristic applied in the specialisation process. It as a whole has been used quite efficiently and successful in many data sets. As mentioned above, two of the most important criteria to assess rule quality are reliability and coverage. In order to achieve this outcome with S measure, both the consistency characteristic and the completeness characteristic were considered. In order to reduce the number of

overlapping rules as well as the total number of rules required to cover the training data, the ratio between the number of positive examples covered by the newly formed conjunction of conditions and not covered by the previously created rules and the number of examples belonging to the target class and not classified by the rule set formed so far was used instead of the coverage of new rules (4.2).

$$Classification\_Gain = \left(\frac{p_{\_new}}{\mathrm{P}_{\_\mathrm{unclassified}}} \right) \qquad (4.11)$$

Ideally, during the rule forming process, the number of examples misclassified should be minimised. Therefore, to assess the newly formed conjunction, the S measure has also used a metric to evaluate the level of misclassification

$$Misclassification\_Level = \left(\frac{n}{N} \right). \qquad (4.12)$$

As one can see, all examples that belong to the target class are considered positive examples, and those do not are considered negative examples. In addition, there are commonly more than two classes in a training set. Thus, by the time the rules formed so far cover the examples belonging to the target class, the number of examples that do not belong to the target class (*N_unclassified*) will reduce.

Accordingly, let's examine the S measure in an example case. Assuming that the training set includes 1000 examples, the specialisation process is in progress with $P = P\_unclassified = 100$, $N = 900$ and $N\_unclassified = 8$. At that time, there are two descriptions including rule R_1 with $p = p\_new = 22$, $n = 7$ and rule R_2 with $p = p\_new = 17$ and $n = 1$. Using the S measure, the first description is selected (S = 0.1656). However, the soft criteria select the second one, which is intuitively more predictive than the first.

4.2.3 A novel specialisation heuristic

Inheriting of strong features from the S measure, the new specialisation heuristic proposed to also use two factors including *Consistency* (4.1) and *Classification_Gain* (4.11). A rule with higher *Consistency* and *Classification_Gain* is more desirable than one with lower indications of these factors.

In many real applications, there are not only two classes in a training set. Therefore, in the rule forming process, the number of examples (*N_unclassified*) will reduce after the rules formed so far classify the examples belonging to the target class. When *N_unclassified* is small, it has a considerable effect on the measure that assesses the rule quality of the rules generated. However, all eight heuristics described in Section 4.2.2 have not paid satisfactory attention to this N_unclassified parameter.

In order to overcome this problem, it is necessary to use a good metric to evaluate the number of examples misclassified by the newly formed conjunction (n) comparing to the total unclassified negative examples (N_unclassifed). This metric could be defined as the level of misclassification and computed as below:

$$\textit{M}\text{isclassification\_Level} = \frac{\text{n}}{\text{N\_unclassified}} \qquad (4.13)$$

Together with *Consistency* and *Classification_gain*, a good newly formed conjunction is expected with the smallest level misclassification. Accordingly, a new specialisation heuristic has been designed as below:

$$TV\ measure = \frac{p}{p+n}\frac{p_{-new}}{P_{-unclassified}}\left(1-\frac{n}{N_{-unclassified}}\right) \qquad (4.14)$$

4.2.4 Specialisation heuristics evaluation

Bigot (2004) also evaluated and compared the eight heuristics as described in Section 4.2.2. His research has proposed and proved that the S measure gives the best results. Therefore, this research will not repeat the tests for eight heuristics mentioned. In stead, the new heuristic - TV measure will be evaluated and then compared with the S measure. The TV measure is also tested on several data sets and compared with the S measure as well as the H measure (the classification heuristic used in original algorithms).

4.2.4.1 Graphical evaluation

Graphical representation has been used by Bigot (2004) to evaluate the performance of the eight heuristics discussed in Section 4.2.2. Each function defines a surface. The value of the function f(p, n) at each point on this surface represents the performance of a rule that classifies *p* examples while misclassifies *n* examples. Thus, with graphical representation the performance of a rule can be studied by comparing the values of two corresponding points on the surface.

In this section, a graphical representation of these functions will be used to study the performance of two specific rules, ($R_1(p_1, n_1)$ and $R_2(p_2, n_2)$), during the forming process. In this case, each heuristic for a rule can be regarded as a function of two variables, being the number of examples belonging to the target class and not classified by the rule set formed so far (*P_unclassified*) and the number of examples not belonging to the target class and not classified by the rule set formed so far (*N_unclassified*). In this way, the performance of the two rules can be observed clearly on two surfaces.

An assumption is made that the specialisation process is in progress at the time *P* = *P_unclassified* = 100, N = 900 and *N_unclassified* = 8. At that time, there are two descriptions including rule R_1 with *p* = *p_new* = 22, *n* = 7 and rule R_2 with *p* = *p_new* = 17 and n = 1.

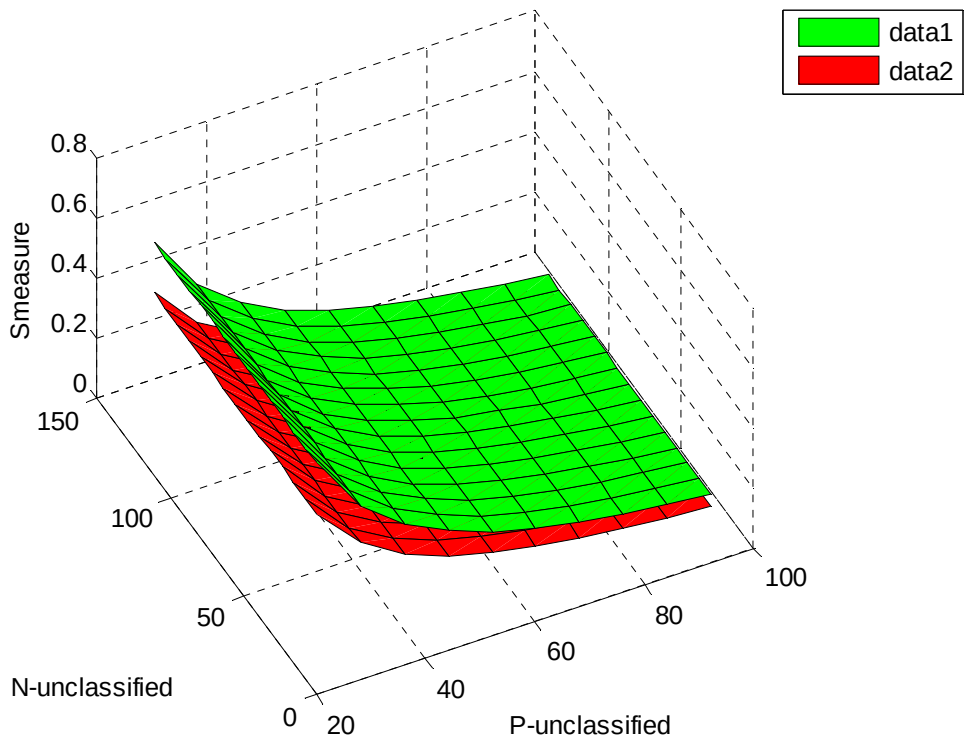


Figure 4.1 Graphical representation of the S measure

The green surface is the representation of rule $R_1(22, 7)$
The red surface is the representation of rule $R_2(17, 1)$

According to Figure 4.1, the green surface (Rule 1) is always above the red surface (Rule 2). This mean that, with any values of the number of examples belonging to the target class and not classified by the rule set formed so far (*P_unclassified*), and the number of examples not belonging to the target class and not classified by the rule set formed so far (*N_unclassified*) the measure assessing R_1 is always higher than the one assessing rule R_2

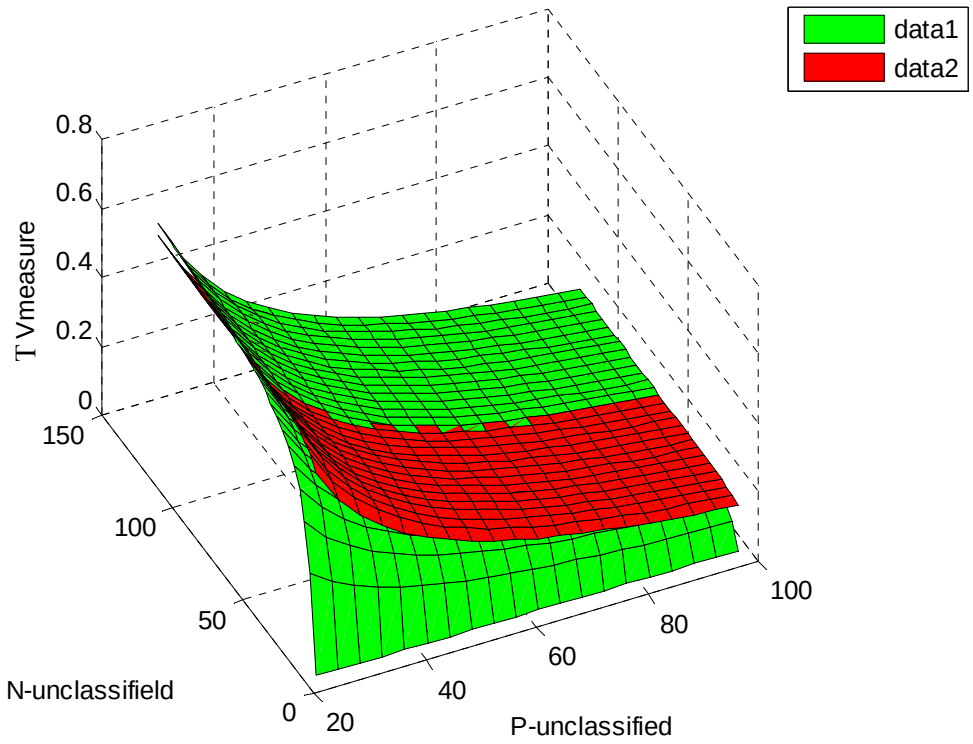


Figure 4.2 Graphical representation of the TV measure

The green surface is the representation of rule $R_1(22, 7)$
The red surface is the representation of rule $R_2(17, 1)$

As can be seen in Figure 4.2, the green surface (rule R_1) lies above the red surface (rule R_2) When the number of examples not belonging to the target class and not classified by the rule set formed so far (*N_unclassified*) is big enough. At that point, the measure assessing rule R_1 is higher than the measure of rule R_2. In contrast, when *N_unclassified* moves closer to n, the red surface will lies above the green surface. This indicates that, the measure assessing rule R_2 is higher than the measure of rule R_1.

4.2.4.2 Experimental evaluation

Based on two assessment criteria, the number of rules created and test accuracy, the TV measure will be tested on 15 data sets as described in Appendix A. Table 4.1 shows the results obtained by applying the H measure, the S measure and the TV measure to the Rules-8. As can be seen, the number of rules is reduced with al the three measures. However, the test accuracy is increased with the TV measure. Therefore, it may be concluded that the TV measure gives the best results.

Table 4.1: Performance of the H measure, the S measure and the *TV measure* when used in Rules-8

	H measure		S measure		TV measure	
	Rules	**Test %**	**Rules**	**Test %**	**Rules**	**Test %**
Balance_scale	195	72.2045	157	71.5655	134	71.2460
breast_cancer	12	94.8424	12	95.4255	12	96.2751
wdbc	6	94.0141	6	94.7183	6	94.7183
wpbc	8	78.7879	8	83.8384	8	87.8788
car	139	90.8565	143	99.4213	127	90.1620
credit screening	34	81.2139	33	80.6358	31	81.2139
dermatology	21	87.9781	21	91.8033	21	93.9891
diabetes	39	73.6971	36	73.1771	33	72.3958
glass	9	99.0654	9	99.0654	9	99.0654
haberman	34	90.8497	25	88.8889	25	89.5425
iris	5	96	5	96	5	96
liver	27	92.4419	24	92.2791	21	90.1163
Tic Tac Toe	24	98.3299	22	97.9123	22	98.3299
Ecoli	23	100	23	100	23	100
Teaching	29	86.6667	27	85.3333	27	85.3333

4.3 Classification heuristics

4.3.1 Development of classification heuristic

In the classification process, an example may be covered by more than one rule. In such cases, classification metrics are used to assess the accuracy and generality of each covering rule and select the best one to classify a new example.

Among the eight specialisation heuristics discussed in this chapter, five could also be used as classification metrics: Accuracy, Consistency, the H measure, the Laplace measure and the Q measures. The Information Gain, the S measures cannot be employed because they require data available only during the rule forming process.

As can be seen, where an unordered set of rules is generated, classification heuristics need to provide additional information to select between two or more competing classes. Bigot (2004) presented a method to address this problem which has not been explicitly defined in CN2 and in the RULES family. In this method, the numbers of examples covered by the rules for each class are summed after weighting the importance of a given rule using the H measure. Then, the class having the highest sum is chosen to classify the new example. This method was tested on 15 data sets and achieved efficient results. However, in several cases, the method did not obtain good results. For instance, let's consider the distribution of examples in two rules (Rule 1, Rule 2) in Figure 4.3 below.

Rule 1: Covers 26 examples of class 1 and 5 examples of class 2, H measure = 0.064

Rule 2: Covers 19 examples of class 1 and 57 examples of class 2, H measure = 0.055

According to the method presented by Bigot (2004), the corresponding sums are:

For class 1, 26*0.064 + 19* 0.055 = 2.709

For class 2, 57*0.055 + 5*0.064 = 3.455

Because the sum for class 2 is the highest, the new example is considered to belong to class 2. However, as can be seen, the most distribution of the example of class 1 are gathered in the intersection area between Rule 1 and Rule 2. Therefore, intuitively, it is more accuracy to classify the new example to class 1 rather than to class 2.

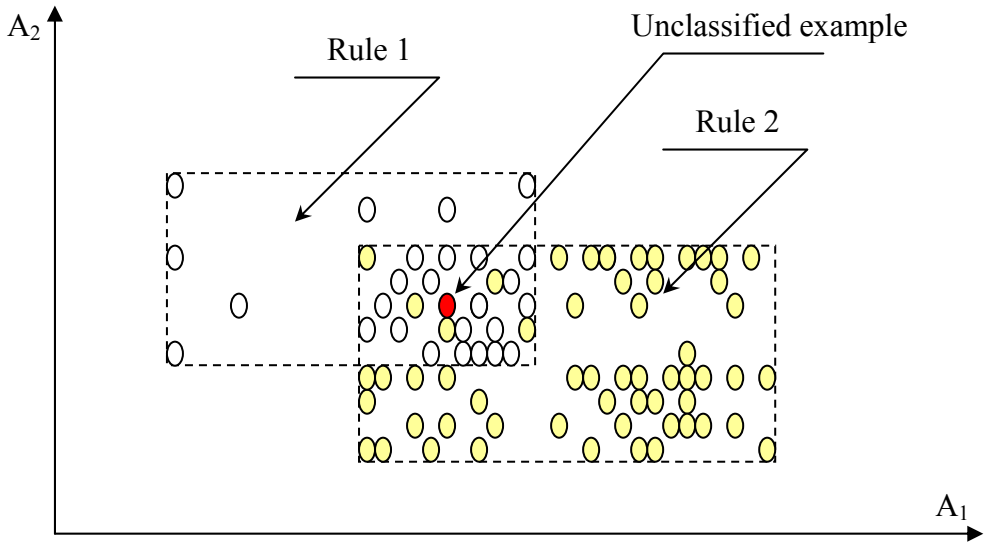


Figure 4.3: Classification of an example covered by Rule 1 and Rule 2

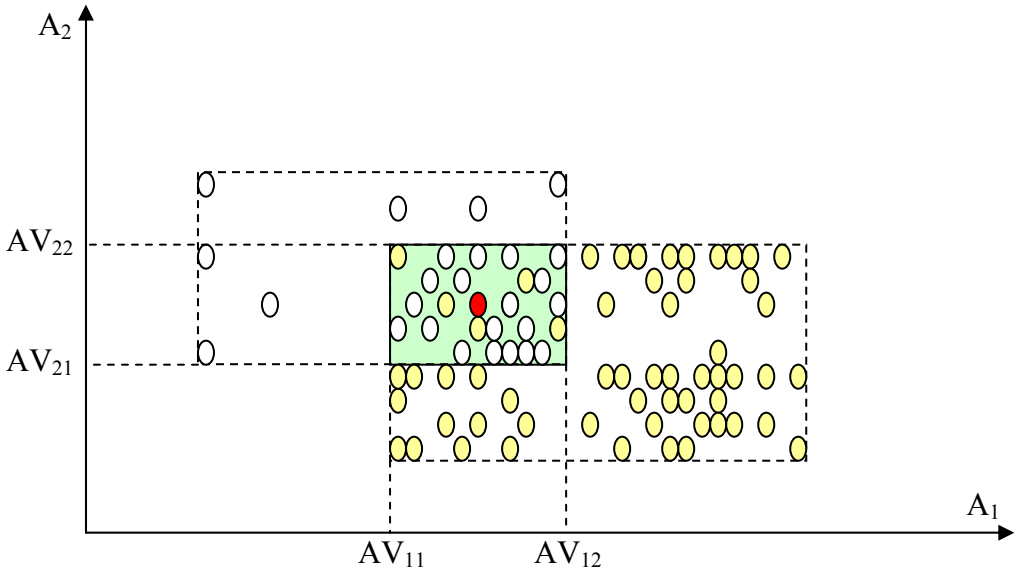


Figure 4.4: Intersection area of Rule 1 and Rule 2

In this section, a development to classify an example will be presented. In this case, the *information-gain* is defined as the product of the number of examples in the intersection area for each class and the H measure correspondingly. The algorithm will estimate the distribution of the example in the intersection area of these rules by finding the *information-gain*. The class having the highest *information-gain* is chosen to classify the new example.

For example, in Figure 4.4, the intersection area of Rule 1 and Rule 2 covers 19 examples of class 1 and 5 examples of class 2.

Information-gain for class 1 is: 15 * 0.064 = 0.96

Information-gain for class 2 is: 5 * 0.055 = 0.275

Because the information-gain for class 1 is the highest, the new example is considered to belong to class 1.

4.3.2 Experimental evaluation

The results for test accuracy of the three classification methods presented above - the RULES family method, Bigot method and the proposed new classification method - are shown in Tables 4.2. In this case, the *TV measure* is used as a specialisation heuristic and the BPP technique (section 3.3.1) is employed for noise handling with four arbitrary values for the noise level (0, 0.1, 0.2 and 0.3).

As can be seen, the average relative deviation of test accuracy values obtained for the new method is higher than those obtained for the other two methods (1.1827 % against -0.7469 % for the RULES family method and -0.4359 % for the Bigot method). Hence, the new classification method can outperform both the RULES family and the Bigot classification methods.

Table 4.2: Classification performance of the heuristic when applying Rules-8

Data set	NL (%)	Test accuracy (%)			Average	Relative deviation (%)		
		Rules family method	Bigot's method	New method		Rules family method	Bigot's method	New method
Balance_scale	0	86.5815	87.2204	86.9010	86.9010	-0.3195	0.3194	0.0000
	0.10	84.3450	85.3035	85.6230	85.0905	-0.7455	0.2130	0.5325
	0.20	82.4281	83.0671	82.7476	82.7476	-0.3195	0.3195	0.0000
	0.30	82.7476	83.3836	84.0256	83.3856	-0.6380	-0.0020	0.6400
Breast_cancer	0	95.9885	94.8424	96.2751	95.7020	0.2865	-0.8596	0.5731
	0.10	93.4097	93.9828	94.8424	94.0783	-0.6686	-0.0955	0.7641
	0.20	93.9828	94.2693	94.5559	94.2693	-0.2865	0.0000	0.2866
	0.30	94.2693	94.2693	94.2693	94.2693	0.0000	0.0000	0.0000
wdbc	0	96.1268	96.4789	97.8873	96.8310	-0.7042	-0.3521	1.0563
	0.10	95.7746	96.1268	96.8310	96.2441	-0.4695	-0.1173	0.5869
	0.20	86.6197	85.5634	89.0845	87.0892	-0.4695	-1.5258	1.9953
	0.30	87.6761	84.8592	85.9155	86.1503	1.5258	-1.2911	-0.2348
wpbc	0	81.8182	82.8283	89.8990	84.8485	-3.0303	-2.0202	5.0505
	0.10	79.7980	81.8282	84.8485	82.1582	-2.3602	-0.3300	2.6903
	0.20	76.7677	76.7677	81.8182	78.4512	-1.6835	-1.6835	3.3670
	0.30	72.7273	72.7273	77.7778	74.4108	-1.6835	-1.6835	3.3670
Car	0	100	100	99.7685	99.9228	0.0772	0.0772	-0.1543
	0.10	90.3935	90.3935	90.6250	90.4707	-0.0772	-0.0772	0.1543
	0.20	85.6481	85.5324	85.7639	85.6481	0.0000	-0.1157	0.1158
	0.30	70.7176	70.7176	73.2639	71.5664	-0.8488	-0.8488	1.6975
Credit	0	87.5723	86.4162	89.3064	87.7650	-0.1927	-1.3488	1.5414
	0.10	86.9942	85.8382	87.2832	86.7052	0.2890	-0.8670	0.5780
	0.20	84.1040	83.5260	84.9711	84.2004	-0.0964	-0.6744	0.7707
	0.30	83.5260	85.2601	84.1040	84.2967	-0.7707	0.9634	-0.1927
Dermatology	0	89.6175	89.6175	92.3497	90.5282	-0.9107	-0.9107	1.8215
	0.10	89.6175	89.6175	91.2568	90.1639	-0.5464	-0.5464	1.0929
	0.20	89.6175	89.6175	89.6175	89.6175	0.0000	0.0000	0.0000
	0.30	83.6066	83.6066	86.8852	84.6995	-1.0929	-1.0929	2.1857
Diabetes	0	75.5208	75.2604	77.8646	76.2153	-0.6945	-0.9549	1.6493
	0.10	75.7813	75.7813	76.0417	75.8681	-0.0868	-0.0868	0.1736
	0.20	75	74.7396	74.7396	74.8264	0.1736	-0.0868	-0.0868
	0.30	67.4479	67.4479	67.4479	67.4479	0.0000	0.0000	0.0000
Glass	0	99.0654	99.0654	99.0654	99.0654	0.0000	0.0000	0.0000
	0.10	99.0654	99.0654	99.0654	99.0654	0.0000	0.0000	0.0000
	0.20	99.0654	99.0654	99.0654	99.0654	0.0000	0.0000	0.0000
	0.30	99.0654	99.0654	99.0654	99.0654	0.0000	0.0000	0.0000
Haberman	0	90.8497	94.7712	94.7712	93.4640	-2.6143	1.3072	1.3072
	0.10	93.4641	97.3856	97.3856	96.0784	-2.6143	1.3072	1.3072
	0.20	79.0850	77.7778	80.3922	79.0850	0.0000	-1.3072	1.3072
	0.30	73.2026	73.2026	77.7778	74.7277	-1.5251	-1.5251	3.0501
Iris	0	96	96.0000	96	96.0000	0.0000	0.0000	0.0000
	0.10	93.3333	93.3333	93.3333	93.3333	0.0000	0.0000	0.0000
	0.20	93.3333	93.3333	93.3333	93.3333	0.0000	0.0000	0.0000
	0.30	93.3333	93.3333	93.3333	93.3333	0.0000	0.0000	0.0000
Lever	0	92.4419	92.4419	94.7674	93.2171	-0.7752	-0.7752	1.5503
	0.10	70.3488	71.5116	76.1628	72.6744	-2.3256	-1.1628	3.4884
	0.20	66.2791	67.4419	68.0233	67.2481	-0.9690	0.1938	0.7752
	0.30	59.3023	61.6279	64.5349	61.8217	-2.5194	-0.1938	2.7132
Tic Tac Toe	0	98.3299	98.3299	98.3299	98.3299	0.0000	0.0000	0.0000
	0.10	98.3299	98.3299	98.3299	98.3299	0.0000	0.0000	0.0000
	0.20	98.3299	98.3299	98.3299	98.3299	0.0000	0.0000	0.0000
	0.30	92.0668	92.0668	92.0668	92.0668	0.0000	0.0000	0.0000
Ecoli	0	98.2143	100	100	99.4048	-1.1905	0.5952	0.5952
	0.10	84.5238	84.5238	94.0476	87.6984	-3.1746	-3.1746	6.3492
	0.20	76.7857	76.7857	76.7857	76.7857	0.0000	0.0000	0.0000
	0.30	77.3810	74.4048	77.9762	76.5873	0.7937	-2.1825	1.3889
Teaching	0	86.6667	86.6667	90.6667	88.0000	-1.3333	-1.3333	2.6667
	0.10	72	72	81.3333	75.1111	-3.1111	-3.1111	6.2222
	0.20	68	72	76	72.0000	-4.0000	0.0000	4.0000
	0.30	64	68	69.3333	67.1111	-3.1111	0.8889	2.2222
Average		85.6348	85.9458	87.5644	86.3817	-0.7469	-0.4359	1.1827

4.4 Summary

This chapter has studied and analysed different heuristics used in covering algorithms for specialisation and classification process. A new specialisation heuristics has been proposed to address specific weaknesses of existing heuristics. Several data sets have been used to test the new specialisation heuristic. The results show that it has advantages over the existing heuristics. It leads to the creation of rules that has greater generality.

A classification method, one that is used to classify a new example when the example is covered by multiple rules, has also been studied in this chapter. The analysis results in a new classification method which is more accurate. However, there are still challenges as even the best heuristics cannot always generate the best results. It is not uncommon to see a less efficient heuristic providing better result in various situations.

Chapter 5

LEARNING WITH CONTINUOUS OUTPUT

5.1 Preliminaries

In the field of information technology, many applications require the creation of models for the prediction of continuous values. Some of the effective existing learning methods that are used to predict numeric values include standard regression, neural network, instance based learning, and regression trees. All of them make prediction by discretising the class values in advance. Despite their advantages, these methods have certain drawbacks.

Standard regression may not be an effective way to represent an induced function because it imposes a non-linear relationship on the data. The neural network and Instance-based learning methods are somehow more powerful, yet they are less efficient in providing information about the structures of the functions that they represent.

A regression tree algorithm is perhaps the most popular and effective. The operational mechanism is similar to that of the decision tree, and the process of modeling is also explained in a similar way. The most noticeable work among the versions of regression trees is CART (Breiman, et al., 1984), which sets the basis for most latter versions. Morimoto (Morimoto et al., 1997) is a representative amongst these improved versions. It is more accurate in forming region-slitting regression trees. However, the scope of the test is larger because more internal nodes are covered. A shortcoming of the types of regression trees developed based on CART is that they only function effectively when values at their leaves are constant. Thus there is a limitation in the number of values to be predicted.

The decision tree model has also suggested a new type of regression methods using local linear regression functions instead of single values at the leaves (Karalic, 1992). By this method, Karalic (1992) has overcome the shortcomings of the previous regression tree methods. In this regards, Quinlan (1992) also proposed M5 algorithm. Like other methods, M5 has been applied in reality. Even though these regression methods have been continuously improved and have helped solve real life problems, in many applications, they need to have not only a high predictive accuracy, but also be close to human reasoning – i.e. – they should be easy to interpret and comprehend for users.

By now, learning with continuous classes using fuzzy rule generation is no longer a new concept, and there are effective learning methods available for predicting real applications. Another model based on fuzzy logic has been widely used for the development of expert systems and controllers due to its similarity to some aspects of human reasoning – the Wang and Mendel's (1991) method. With a five-step procedure, Wang and Mendel determine a mapping from the input space to the output space based on the combined fuzzy rule base. In this method, fuzzy rules are generated for examples based on membership functions that have been pre-defined to divide the attribute space into fuzzy regions. Problems arise as the number of rules generated grows, and challenges the memory capacity. Wang and Mendel (1991) referred to this as the "growing memory" problem. Due to this matter, the selection of the best rules becomes less efficient.

Similar to Wang and Mendel's algorithm, Nozaki and Ishibuchi (1997) proposed to use a particular heuristic method to automatically generate fuzzy if-then rules from numerical data. The fuzzy if-then rules with non-fuzzy singletons (i.e., real numbers) in the consequent parts are generated by assessing single real numbers (instead of membership functions) which are to be stored in each cell of the decision table. Since there is not

defuzzification step involved, Nozaki and Ishibuchi's algorithm is quite simple. However, the problem of growing memory still remains.

Sebag and Schoenauer(1994) observed that the problem of designing membership functions might be just as complex as designing fuzzy rules. Both methods above need to pre-define membership functions, which is actually not an easy task. Various methods for automatic creation of membership functions have been devised. However, all still cannot solve the existing problems, not to mention the rise of new problems. One of the challenges is the demand of post-processing the large fuzzy rule sets formed to acquire more compact rule sets.

To resolve the problems for automatic membership functions design, Hong and Lee (1996) proposed a method, by which the fuzzification of output is performed using a clustering procedure that regards examples in the training set (T) with close output values as belonging to the same fuzzy set. Appropriate membership functions are then assigned to represent each fuzzy set. Initial membership functions are assigned to attributes in the form of a triangle base equal to a small interval predefined by a user. After attributes have received their initial membership functions, the decision table is built using the examples in T. The decision table is then simplified through the process of merging membership functions from which the final rule sets can be extracted.

Nevertheless, a problem still exists with the merging process regardless of the attempt to improve it by the authors Hong and Lee (1999). That is, it can be highly computationally expensive as the number of attributes increases. Hong and Chen (2000) also attempted to develop a method to simplify the initial membership functions. Their method has been proven to be more efficient and accurate, yet it does not reduce considerably the computational cost. This method, however, generates a set of fuzzy rules where the

membership functions have been automatically created and the universes of discourse are not equally partitioned. It should be noted that it is still a challenge when the number of attributes increases.

In 2004, Bigot released a new technique called DynaFuzz that can automatically create input membership functions. Inheriting the advantages of its predecessors – theDyna and DynaSpace inductive learning algorithms, DynaFuzz can generate more compact and more accurate fuzzy rule sets. However, in the present world, given that databases are becoming more diverse with more features as well as bigger quantity, there is an increasing need for improving existing algorithms and developing new ones. The need for further research on the automation of creating output membership functions is also of no exception.

This chapter presents a novel technique for fuzzy rule induction that combines the capabilities of fuzzy logic for continuous outputs and uncertainty handling with the good performance of RULES-8 algorithm. The technique, named TVFuzz, was designed as a covering algorithm that allows the creation of compact fuzzy system. Follow are a description of TVFuzz and the results obtained from an experimental evaluation of Rule8Fuzz on some benchmark datasets.

5.2 A novel fuzzy rule generation

This section proposes the TVFuzz algorithm, a simple but powerful method for automatically generating fuzzy IF-THEN rule. TVFuzz uses the simplification technique of RULES-8 to form fuzzy rules. It thus has to predefine the target class. At this time, the outputs have been discretised. Based on RULES-8 rule forming process, rule sets are generated. The class of the rules is the target fuzzy. Therefore, each condition of the rule is also fuzzified. The fuzzy rules finally are generated. The fuzzy rule generation procedure is described in Figure 5.1 as shown below.

Step 1: Fuzzify the output data. Each output value has to be fuzzified in order to determine the target output fuzzy set.

Step 2: Create rule set. At this time, the outputs have been discretised. Based on RULES-8 rule forming process, rule sets are generated

Step 3: Fuzzy rule generation. The class of the rule is replaced by the target fuzzy, and each condition is also fuzzified.

Figure 5.1 Fuzzy rule generation procedure

5.2.1 Fuzzification of outputs

In learning with continuous outputs using fuzzy rule generation, the output values as a whole should be discretised and then fuzzified. Perhaps the most famous technique for fuzzification of output is employed by Hong and Lee (1995). According to Hong and Lee (1995), the output values of the training set are sorted out in ascending order and then using α_{cut} of similarity to cluster the examples (α_{cut} is the threshold for two adjacent data to be thought of belonging to the same class). After the above operation, i^{th} output data will be clustered into F_k^{out}. At this time, triangle membership functions are used for each group F_k^{out}. A triangle membership function can be defined as $F_k^{out} = Tr(a(k), b(k), c(k))$

$$b(k) = \frac{V_k^{out} s_k + V_{k+1}^{out} \frac{s_k + s_{k+1}}{2} + \ldots + V_{j-1}^{out} \frac{s_{j-2} + s_{j-1}}{2} + V_j^{out} s_{j-1}}{s_k + \frac{s_k + s_{k+1}}{2} + \ldots + \frac{s_{j-2} + s_{j-1}}{2} + s_{j-1}} \tag{5.1}$$

$$a(k) = b(k) - \frac{b(k) - V_k^{out}}{1 - \mu_k(V_k^{out})} \tag{5.2}$$

$$c(k) = b(k) + \frac{V_j^{out} - b(k)}{1 - \mu_k(V_j^{out})} \tag{5.3}$$

$V_k^{out}, V_{k+1}^{out}, \ldots, V_{j-1}^{out}, V_j^{out}$ are output values of group k^{th}

$s_k, s_{k+1}, \ldots, s_{j-1}, s_j$ are the value of similarity between adjacent data

$$s_k = \begin{cases} 1 - \frac{diff_k}{C * \sigma_s} \; for \; diff_k \le C * \sigma_s \\ 0 \; otherwise \end{cases} \tag{5.4}$$

$diff_k$ is the distance between V_k^{out}, V_{k+1}^{out}

C is a control parameter deciding the shape of the membership function of similarity

σ_k is the standard deviation of $diff_k$'s

By this way, the membership function of the outputs is automatically constructed. However, in case the outputs are a range of values with little disparity, the output data will be divided into many small clusters. Each cluster may contain only one value, and sometimes, there is only one cluster that contains all the output values. This can cause the problem of complication and inaccuracy. For example in robot control when it comes to determining joint movements and positions. Consequently, the application of this algorithm in this case is not so effective.

In order to overcome the problems in Hong and Lee's method, Wang and Mendel (1991) used a technique to divide the output data into two ranges: $2N+1$. Bigot (2004) specified this technique as below:

The output range thus has been decomposed into Nf triangular fuzzy set $(F_1^{out},...,F_k^{out},...,F_{Nf}^{out})$ defined as $F_k^{out} = Tr(a(k),b(k),c(k))$

Where k is an integer included in [1, Nf],

$$b(k) = \frac{V_{\max}^{out} - V_{\min}^{out}}{Nf - 1}(k-1) \qquad (5.5)$$

$$a(k) = b(k) - \frac{V_{\max}^{out} - V_{\min}^{out}}{Nf - 1} \qquad (5.6)$$

$$c(k) = b(k) + \frac{V_{\max}^{out} - V_{\min}^{out}}{Nf - 1} \qquad (5.7)$$

Based on this, an example E with its output value (V_E^{out}) could belong to two membership functions. The output membership functions will be the fuzzy set F_K^{out} where the membership degree $\mu_{F_K^{out}}(V_E^{out})$ is maximum.

Dividing the outputs into equal ranges is a simple but effective method in case there is not much difference among the output values. However, the formula (5.5) (Bigot, 2004) cannot be applied for negative values. Sinceb(k) is always 0 when k = 1, b(1) cannot be computed when the range of the outputs begins with a negative value.

To sum up, between the two methods mentioned above, each has its own advantages depending on the nature of the output values. In this research, efforts will continue to be made to improve the techniques for the less dissimilar output values. However, formula (5.5) can be improved in order to apply for both positive and negative values, as can be seen below:

$$b(k) = \frac{V_{\max}^{out} - V_{\min}^{out}}{Nf - 1}(k - 1) + V_{\min}^{out} \qquad (5.8)$$

Figure 5.2 presents the output fuzzification having the range of values from -30 to 30. It is proposed to split the continuous output range in to 6 clusters.

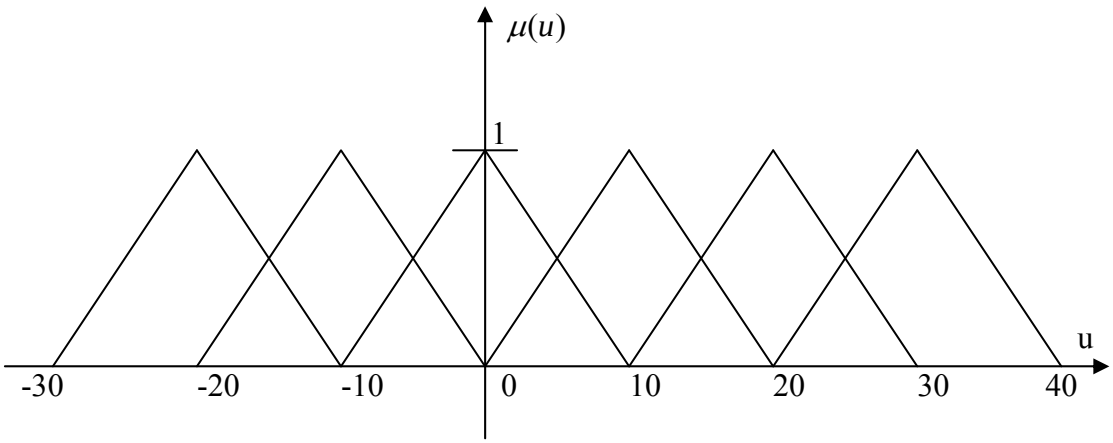


Figure 5.2: Output fuzzification

5.2.2 Formation fuzzy rule

As can be seen above, after the output data are predefined, they are now considered the discrete values $(F_1^{out},...,F_k^{out},...,F_{Nf}^{out})$. At that time, TVFuzz uses RULES-8 algorithm (Chapter 3) to form fuzzy rules. The rule forming process can be summarised as follows:

Step 1: Select randomly one uncovered attribute-value from each attribute to find Seed Attribute value (SA). SA is the value applied for the discrete values or the range of values created as a result of the process of finding rules that have the best quality. The difference between this method and RULES-8 is using the *TV measure* to access rule quality instead of using the *H measure* as in RULES-8.

Step 2: Combine SA with other attributes to make the rule more general until the combination does not make the Rule any better. Then a rule R is obtained and belongs to the target class.

The process repeats Step 1 and continues until no uncovered attribute-value remains in the training set. At the end of the rule formation process, a set of rules is generated.

5.2.3 Fuzzification of conditions

As can be seen, one or several rules can be created for each target class. The conditions take the form $Cond_R^i = \left[A^i = V_{SA}^i \right]$ or $Cond_R^i = \left[V_{\min}^i < A^i < V_{\max}^i \right]$ for discrete and continuous values respectively. Each condition is fuzzified using the following method:

- In the case of a continuous attribute, $Cond_R^i = \left[V_{\min}^i < A^i < V_{\max}^i \right]$ is converted into the fuzzy condition $Cond_R^i = \left[A^i \; is \; Tr(a,b,c) \right]$, where:

$$a = V_{\min}^i ; \; b = \frac{V_{\max}^i - V_{\min}^i}{2} \; and \; c = V_{\max}^i \text{, if} V_{\min}^i , \; V_{\max}^i \text{ are finite;}$$

$$a \rightarrow -\infty; \; b = V_{\min}^i ; \text{ and } c = V_{\max}^i , \text{ if } V_{\min}^i \rightarrow -\infty$$

$$a = V_{\min}^i , \; b = V_{\max}^i \text{ and } c \rightarrow +\infty, \text{ if } V_{\max}^i \rightarrow +\infty$$

- In the case of discrete attribute, the condition $Cond_R^i = \left[A^i = V_{SE}^i \right]$ is converted into the fuzzy condition $Cond_R^i = \left[A^i \; is \; Vcod_k^i \right]$, where $Vcod_k^i$ is the coded value of V_{SE}^i

5.3 Illustrative problem

To illustrate the new fuzzy rules generation process, consider the following nonlinear two inputs – a single output system used in the research of Nozaki and Ishibuchi (1997).

$$y = \left(1 + x_1^{-2} + x_2^{-1.5}\right)^2, (1 \le x_1; x_2 \le 5) \qquad (5.9)$$

This was also used in Takagi and Sugeno (1993). 50 input-output pairs were obtained (Appendix 2):

	x_1	x_2	y
1	3.49	4.20	1.44
2	1.18	2.07	4.21
3	3.50	2.11	1.98
4	2.26	4.57	1.69
5	1.06	4.13	4.01
6	4.96	3.90	1.37
7	2.19	4.23	1.75
8	1.55	3.85	2.40
25	4.79	4.53	1.32
26	1.28	2.92	3.27
27	1.82	2.00	2.75
28	4.84	1.55	2.44
29	1.91	2.58	2.29
30	2.05	1.06	4.66
45	1.04	2.24	4.97
46	4.44	4.24	1.36
47	1.67	1.48	3.67
48	4.00	4.74	1.34
49	4.17	4.04	1.39
50	3.55	2.64	1.72

Figure 5.3: Example set

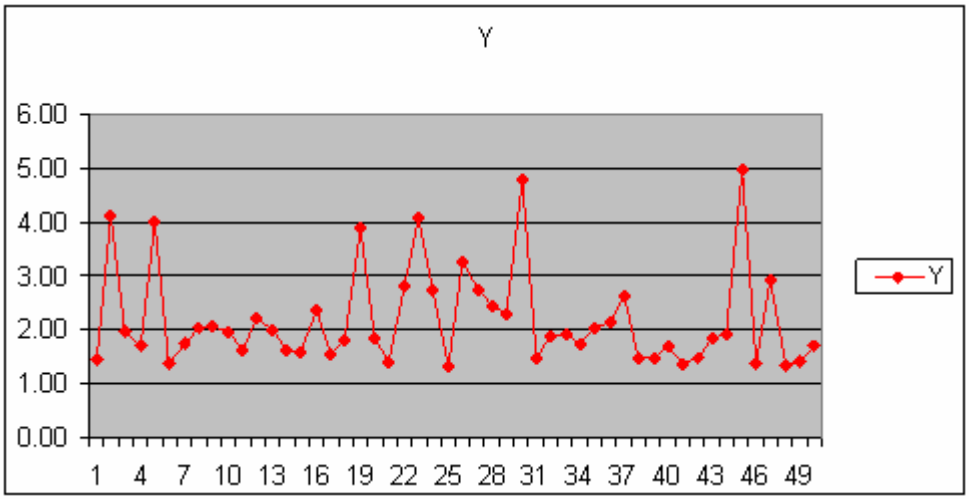


Figure 5.4: Representation of the example set

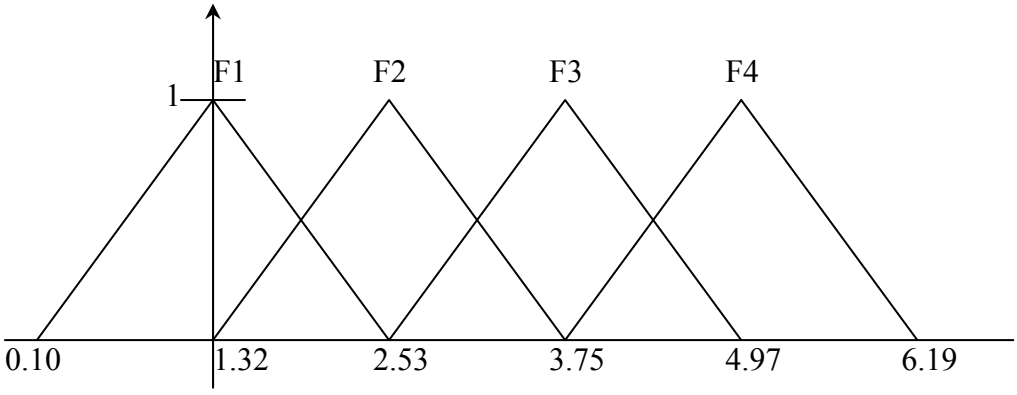


Figure 5.5: Fuzzification of y with Nf = 4

The precision for the model is fixed to *Nf* = 4 membership functions for the outputs. The four possible output membership functions F1, F2, F3, F4 are created (Figure 5.5). TVFuzz process will follow the steps described below.

Step 1:A set of attribute-value is constructed by selecting random attribute-values from x_1, x_2 .For example, SETAV= {[$x_1 = 4$], [$x_2 = 3.34$]}

Steps 2,3: Array expression T_EXP is formed

IF [$x_1 = 4$] THEN [y=1.34], the targeted output set is fuzzy set F1 (1)

IF [$x_2 = 3.34$] THEN [y=1.46] the targeted output set is also fuzzy set F1 (2)

Based on (1), (2) new expressions are generated as follow:

IF [$4 \le x_1 \le 4.65$] THEN [y = F1], covers 9 examples, *TV measure* = 0.3462 (3)

IF [$3.34 \le x_2 \le 3.81$] THEN [y = F1], covers 6 examples, *TV measure* = 0.2308 (4)

Seed attribute-values are [$4 \le x_1 \le 4.65$]

Step 6: Form a set of conjunctions of conditions

SETCC = {[$4 \le x_1 \le 4.65$] and [$2.12 \le x_2 \le 4.74$]}

Steps 7,8: Determine the neighbourhood of Seed attribute-values

- Determine the range of x_1 with [$2.12 \le x_2 \le 4.74$] and [y = F1]. The new range of x_1 is [$2.62 \le x_1 \le 4.96$].

IF [$2.62 \le x_1 \le 4.96$] AND [$2.12 \le x_2 \le 4.74$] THEN [y = F1], *TV measure* = 0.9231 (5)

As can be seen, expression (5) covers 24 examples and *TV measure* = 0.9231, which is higher than the TV measure in expression (3). Thus, the range of x_2 continues to check for extending.

For $[2.62 \leq x_1 \leq 4.96]$ and [y = F1], the new range of x_2 is $[2.12 \leq x_2 \leq 4.74]$, nothing changes with expression (5). Thus, Rule 1 is created as follow:

Rule 1: IF $[2.62 \leq x_1 \leq 4.96]$ AND $[2.12 \leq x_2 \leq 4.74]$ THEN [y = F1], *TV measure* = 0.9231

The process continues until there are no remaining unclassified examples in the dataset. The final RuleSet includes:

Rule 2: IF $[1.35 \leq x_2 \leq 2.11]$ THEN [y = F2], *TV measure* = 0.7059

Rule 3: $[1.82 \leq x_1 \leq 2.54]$ AND $[2.0 \leq x_2 \leq 4.22]$ THEN [y = F2], *TV measure* = 0.4118

Rule 4: $[1.06 \leq x_1 \leq 1.28]$ THEN [y = F3], *TV measure* = 1

Rule 5: $[1.02 \leq x_1 \leq 1.04]$ THEN [y = F4], *TV measure* = 1

Rule 6: $[2.19 \leq x_1 \leq 4.79]$ AND $[2.51 \leq x_2 \leq 4.74]$ THEN [y = F1], *TV measure* = 0.8846

	x_1	x_2	y
1	3.49	4.20	F1
2	1.18	2.07	F3
3	3.50	2.11	F2
4	2.26	4.57	F1
5	1.06	4.13	F3
6	4.96	3.90	F1
7	2.19	4.23	F1
8	1.55	3.85	F2
25	4.79	4.53	F1
26	1.28	2.92	F3
27	1.82	2.00	F2
28	4.84	1.55	F2
29	1.91	2.58	F2
30	2.05	1.06	F4
45	1.04	2.24	F4
46	4.44	4.24	F1
47	1.67	1.48	F3
48	4.00	4.74	F1
49	4.17	4.04	F1
50	3.55	2.64	F1

Figure 5.6: Discretised example sets

Rule 1: IF [$2.62 \leq x_1 \leq 4.96$] AND [$2.12 \leq x_2 \leq 4.74$] THEN [y = F1], TV = 0.9231

Rule 2: IF [$1.35 \leq x_2 \leq 2.11$] THEN [y = F2], TV = 0.7059

Rule 3: [$1.82 \leq x_1 \leq 2.54$] AND [$2.0 \leq x_2 \leq 4.22$] THEN [y = F2], TV = 0.4118

Rule 4: [$1.06 \leq x_1 \leq 1.28$] THEN [y = F3], TV = 1

Rule 5: [$1.02 \leq x_1 \leq 1.04$] THEN [y = F4], TV = 1

Rule 6: [$2.19 \leq x_1 \leq 4.79$] AND [$2.51 \leq x_2 \leq 4.74$] THEN [y = F1], TV = 0.8846

Figure 5.7: Rule set obtained for *Nf* = 4

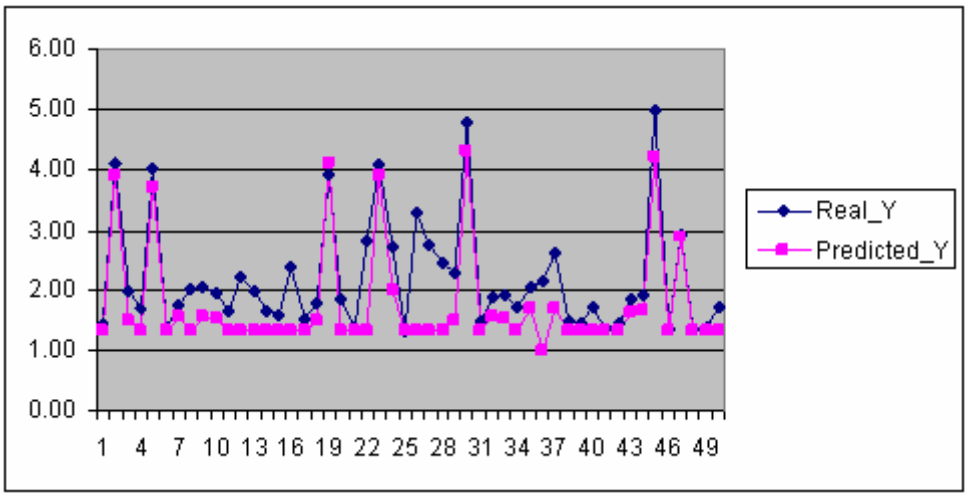


Figure 5.8: Predicted obtained for *Nf* = 4

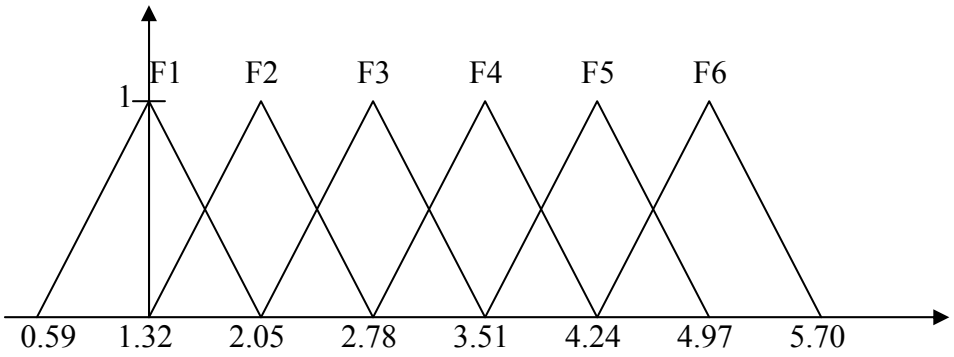


Figure 5.9: Fuzzification of Y with Nf = 6

Rule 1: [$3.43 \leq x_1 \leq 5$] AND [$1.35 \leq x_2 \leq 1.55$] THEN [y = F3]

Rule 2: [$3.65 \leq x_1 \leq 4.83$] AND [$1.58 \leq x_2 \leq 2.52$] THEN [y = F2]

Rule 3: [$3.63 \leq x_1 \leq 4.96$] AND [$2.62 \leq x_2 \leq 4.74$] THEN [y = F1]

Rule 4: [$2.89 \leq x_1 \leq 3.55$] AND [$2.02 \leq x_2 \leq 2.69$] THEN [y = F2]

Rule 5: [$1.85 \leq x_1 \leq 2.62$] AND [$3.27 \leq x_2 \leq 4.57$] THEN [y = F2]

Rule 6: [$2.84 \leq x_1 \leq 3.49$] AND [$3.43 \leq x_2 \leq 4.20$] THEN [y = F1]

Rule 7: [$1.02 \leq x_1 \leq 1.04$] THEN [y = F6]

Rule 8: [$1.06 \leq x_1 \leq 1.18$] THEN [y = F5]

Rule 9: [$1.28 \leq x_1 \leq 1.82$] AND [$2.0 \leq x_2 \leq 2.92$] THEN [y = F3]

Rule 10: [$x_1 = 1.82$] AND [$x_2 = 1.48$] THEN [y = F4]

Figure 5.10: Rule set obtained for Nf= 6

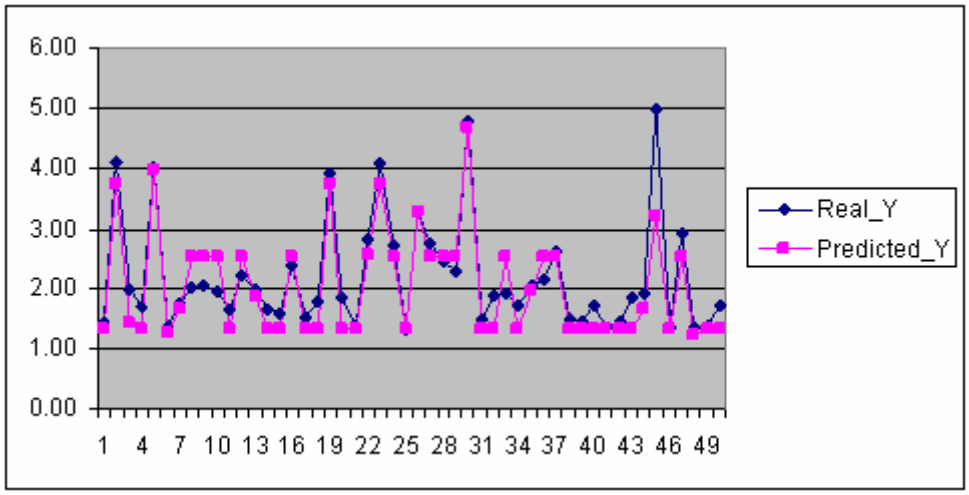


Figure 5.11: Prediction obtained for Nf = 6

5.4 Using fuzzy rule for output prediction

As can be seen above, a fuzzy rule set is generated from the training sets. When a particular data is entered, the system will check to see if it matches with any of the rules and what the degree of match is.

There are two popular methods defining the "degree of match" of an example to the rule. The first method is presented by Hong and Lee (1996). This method is based on the minimum membership degree. It is defined as:

$$\mu\_rule_R(E) = \min\left[\mu_{F_R^i}(V_E^1),...,\mu_{F_R^i}(V_E^1),...,\mu_{F_R^i}(V_E^1)\right] \qquad (5.10)$$

The second method, which is proposed by Nozaki and Ishibuchi (1997), uses the product of all membership degrees. It can be defined as:

$$\mu\_rule_R(E) = \prod_{i=1}^{m}\left[\mu_{F_R^i}(V_E^i)\right] \qquad (5.11)$$

In this research, the degree of match is determined by the first method (Hong and Lee, 1996). For instance, an example regarding to fuzzy set (F_R^i) in rule R has membership degrees of two attributes of 0.77 and 0.17, respectively as in Figures 5.4 and 5.5. This example has a degree of match to rule R defined as **min**(0.77, 0.17) = 0.17.

Based on this, defuzzification techniques are used to identify the actual outputs. The output prediction procedure using fuzzy rules is described as below:

Step 1: Transform numeric input to linguistic terms according to the membership function;

Step 2: Match the linguistic terms with the decision rules to find the output groups;

Step 3: Defuzzify the output groups to form the final decision

Figure 5.3: Output prediction procedure using fuzzy rule

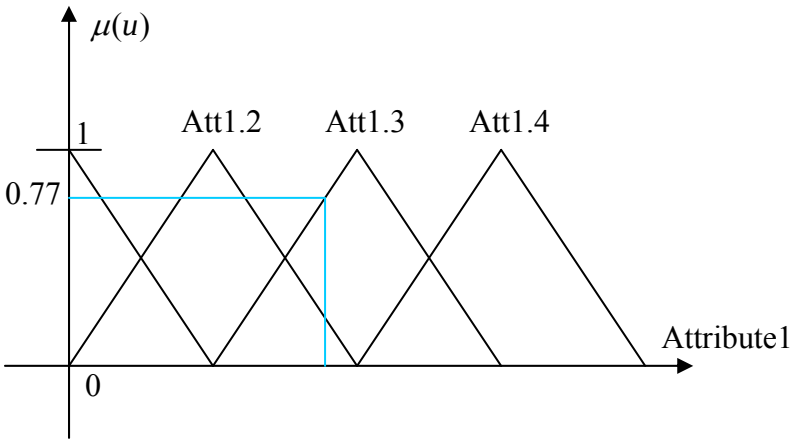


Figure 5.12: The membership degree of attribute 1 = 0.77

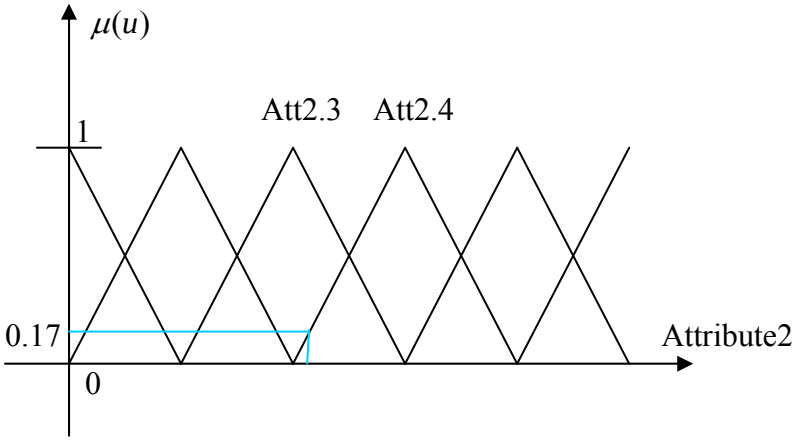


Figure 5.13: The membership degree of attribute 2 = 0.17

5.5 Experimental results

As mentioned in Chapter 2, it can be said thatDynaFuzzis one of the most efficient algorithms in predicting continuous outputs. In the section that follows, two real-life problems will be used to examine the TVFuzz algorithm. The results from this examination will be compared with those from the DynaFuzz algorithm.

5.5.1 Fuzzy model for mathematical model

The first problem will be applied on the sample sets illustrated in DynaFuzz (Bigot, 2004), including 250 samples (Figure 5.13). Each sample contains a discrete value Curve-type (1,2) and a continuous value Ax. The outputs are a range of continuous values,Ay. The following mathematical model represents the entire example space:

IF Curve-type = 1 THEN Ay = sin(Ax)

IF Curve-type = 2 THEN IF Ax = k π THEN Ay = 0

IF Ax $\in]2.k.\pi,(2.k+1)\pi[$ THEN Ay = 1

IF Ax $\in](2.k+1).\pi,(2.k+2)\pi[$ THEN Ay = -1

Where k is an integer $\in [0,+\infty]$

	Curve-type	Ax	Ay
1	1	0.1	0.099833
2	1	0.2	0.198669
3	1	0.3	0.29552
40	1	4	0.756802
41	1	4.1	0.818277
42	1	4.2	0.871576
43	1	4.3	0.916166
70	1	7	0.656987
71	1	7.1	0.728969
72	1	7.2	0.793668
109	1	10.9	0.995436
110	1	11	0.99999
111	1	11.1	0.994553
112	1	11.2	0.979178
147	2	2.2	1
148	2	2.3	1
149	2	2.4	1
206	2	8.1	1
207	2	8.2	1
208	2	8.3	1
248	2	12.3	1
249	2	12.4	1
250	2	12.5	1

Figure 5.14: Example set

In the case of the DynaFuzz algorithm, the outputs are decomposed into 4 membership functions (Figure 5.15)

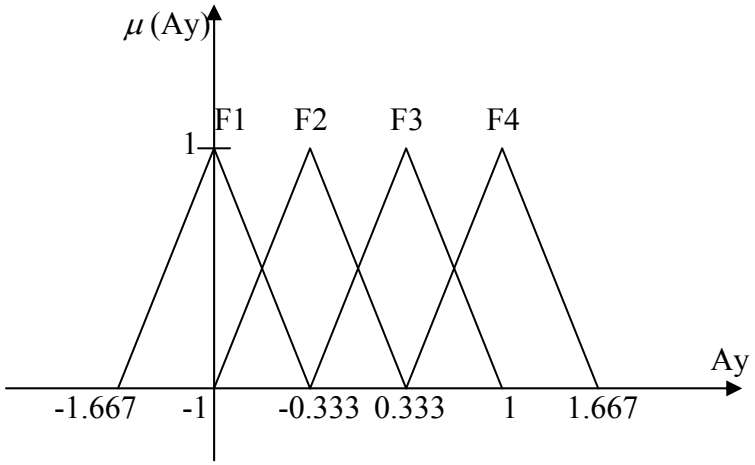


Figure 5.15: Fuzzufication of Ay with *Nf*= 4

After having predefined the membership functions for the outputs, Bigot (2004) applied the DynaFuzz algorithm and obtained 12 rules as shown in Figure 5.16.The prediction of the model for 250 examples is shown in Figure 5.18 with the pink line. For this model, the maximum absolute error equals 1.3291 and mean absolute error equals 0.5682

Similarly, the TVFuzz algorithm also decomposes the output values into 4 triangular membership functions, and obtains 16 rules as shown in Figure 5.17. The prediction of the model is shown in Figure 5.18 with yellow line. For this model, the maximum absolute error equals 0.9198 and mean absolute error equals 0.4938

R1 IF [Curve-type is 1] AND [Ax is Tr (-∞, 0.1, 3.5)] THEN Ay is F3
R2 IF [Ax is Tr (0.3, 1.6, 2.9)] THEN Ay is F4
R3 IF [Curve-type is 1] AND [Ax is Tr (2.8, 4.75, 6.7)] THEN Ay is F2
R4 IF [Ax is Tr (3.4, 4.7, 6.)] THEN Ay is F1
R5 IF [Curve-type is 1] AND [Ax is Tr (5.9, , 9.8)] THEN Ay is F3
R6 IF [Ax is Tr (6.6, , 9.1)] THEN Ay is F3
R7 IF [Curve-type is 1] AND [Ax is Tr (9,12.5,+ ∞)] THEN Ay is F2
R8 IF [Ax is Tr (9.7, , 12.3)] THEN Ay is F1
R9 IF [Curve-type is 2] AND [Ax is Tr (-∞,0.1, 3.2)] THEN Ay is F4
R10 IF [Curve-type is 2] AND [Ax is Tr (3.1, ,6.3)] THEN Ay is F1
R11 IF [Curve-type is 2] AND [Ax is Tr (6.2, ,9.5)] THEN Ay is F4
R12 IF [Curve-type is 2] AND [Ax is Tr (9.4,12.5,+ ∞)] THEN Ay is F1

Figure 5.16: Rule set obtained for *Nf*=10 using DynaFuzz algorithm

R1 IF [Curve-type is 1] AND [Ax is Tr (-∞, 0.1, 0.7)] THEN Ay is F3
R2 IF [Curve-type is 2] AND [Ax is Tr (-∞, 0.1, 3.1)] THEN Ay is F4
R3 IF [Ax is Tr (0.8, 1.6, 2.4)] THEN Ay is F4
R4 IF [Curve-type is 1] AND [Ax is Tr (2.5, 2.8, 3.1)] THEN Ay is F3
R5 IF [Curve-type is 1] AND [Ax is Tr (3.2, 3.5, 3.8)] THEN Ay is F2
R6 IF [Curve-type is 2] AND [Ax is Tr (3.2, 4.7, 6.2)] THEN Ay is F1
R7 IF [Ax is Tr (3.9, 4.7, 5.5)] THEN Ay is F1
R8 IF [Curve-type is 1] AND [Ax is Tr (5.6, 5.9, 6.2)] THEN Ay is F2
R9 IF [Curve-type is 1] AND [Ax is Tr (6.3, 6.65, 7)] THEN Ay is F3
R10 IF [Curve-type is 2] AND [Ax is Tr (6.3, 7.85, 9.4)] THEN Ay is F4
R11 IF [Ax is Tr (7.1, 7.85, 8.6)] THEN Ay is F4
R12 IF [Curve-type is 1] AND [Ax is Tr (8.7, 9.05, 9.4)] THEN Ay is F3
R13 IF [Curve-type is 1] AND [Ax is Tr (9.5, 9.8, 10.1)] THEN Ay is F2
R14 IF [Ax is Tr (10.2, 11, 11.8)] THEN Ay is F1
R15 IF [Curve-type is 2] AND [Ax is Tr (9.5, 12.5,, +∞)] THEN Ay is F1
R16 IF [Curve-type is 1] AND [Ax is Tr (11.9, 12.5, +∞)] THEN Ay is F2

Figure 5.17: Rule set obtained for *Nf*=4 using TVFuzz algorithm

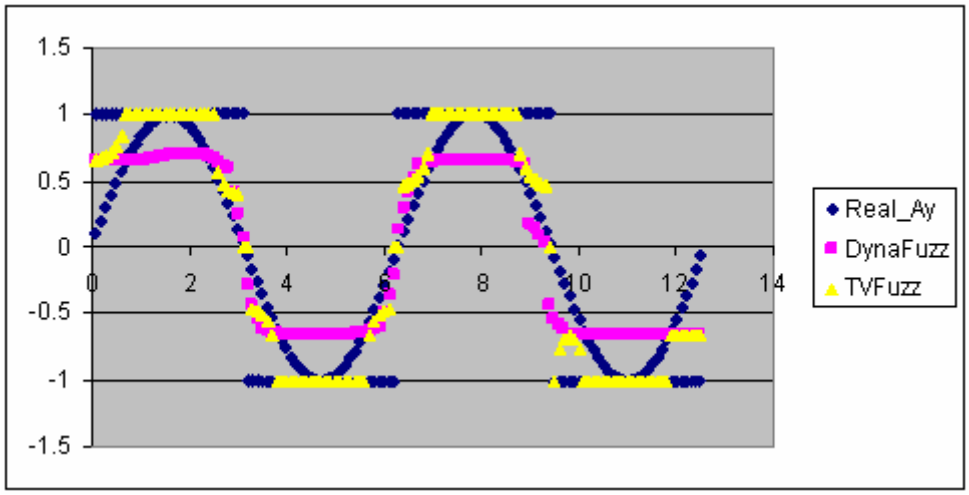


Figure 5.18: Prediction of output Ay using DynaFuzz and TVFuzz algorithm

In the case of the DynaFuzz algorithm, the outputs are decomposed into 10 membership functions (Figure 5.19)

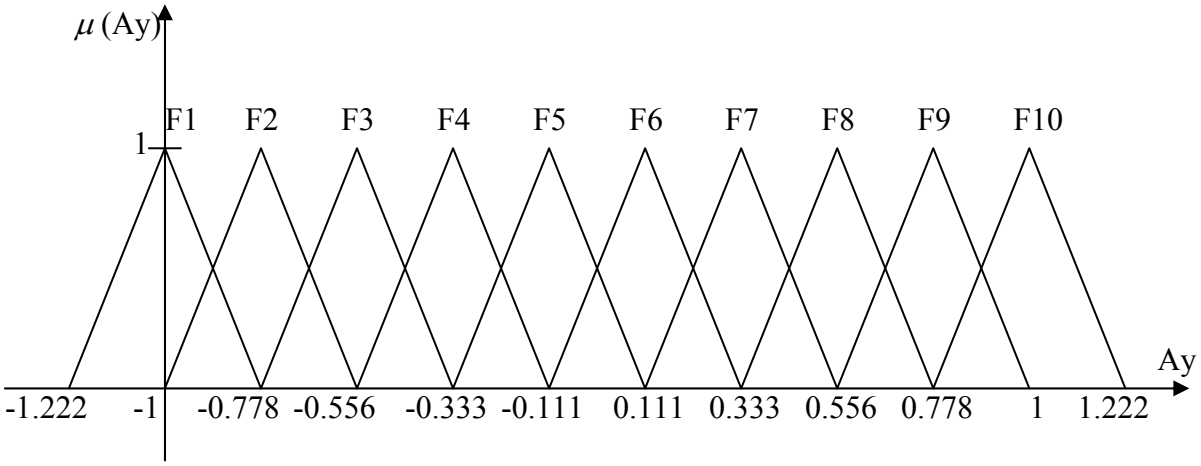


Figure 5.19: Fuzzufication of Ay with *Nf* = 10

After having predefined the membership functions for the outputs, Bigot (2004) applied the DynaFuzz algorithm and obtained 36 rules as shown in Figure 5.20.The prediction of the model for 250 examples is shown in Figure 5.22 with the pink line. For this model, the maximum absolute error equals 1.0293 and mean absolute error equals 0.4192

Similarly, the TVFuzzalgorithm also decomposes the output values into 10 triangular membership functions, and obtains 40 rules as shown in Figure 5.21. The prediction of the model is shown in Figure 5.22 with yellow line. For this model, the maximum absolute error equals 0.8827 and mean absolute error equals 0.3102

R1	IF [Curve-type is 1] AND [Ax is Tr (-∞, 0.1, 0.4)] THEN Ay is F6
R2	IF [Curve-type is 1] AND [Ax is Tr (0.1, 0.35, 0.6)] THEN Ay is F7
R3	IF [Curve-type is 1] AND [Ax is Tr (0.3, 0.6, 0.9)] THEN Ay is F8
R4	IF [Curve-type is 1] AND [Ax is Tr (0.5, 1.55, 2.6)] THEN Ay is F9
R5	IF [Ax is Tr (0.8, 1.55, 2.3)] THEN Ay is F10
R6	IF [Curve-type is 1] AND [Ax is Tr (2.2, 2.55, 2.9)] THEN Ay is F8
R7	IF [Curve-type is 1] AND [Ax is Tr (2.5, 2.8, 3.1)] THEN Ay is F7
R8	IF [Curve-type is 1] AND [Ax is Tr (2.8, 3.05, 3.3)] THEN Ay is F6
R9	IF [Curve-type is 1] AND [Ax is Tr (3, 3.25, 3.5)] THEN Ay is F5
R10	IF [Curve-type is 1] AND [Ax is Tr (3.2, 3.5, 3.8)] THEN Ay is F4
R11	IF [Curve-type is 1] AND [Ax is Tr (3.4, 3.75, 4.1)] THEN Ay is F3
R12	IF [Curve-type is 1] AND [Ax is Tr (3.7, 4.7, 5.7)] THEN Ay is F2
R13	IF [Ax is Tr (4, 4.7, 5.4)] THEN Ay is F1
R14	IF [Curve-type is 1] AND [Ax is Tr (5.3, 5.65, 6)] THEN Ay is F3
R15	IF [Curve-type is 1] AND [Ax is Tr (5.6, 5.9, 6.2)] THEN Ay is F4
R16	IF [Curve-type is 1] AND [Ax is Tr (5.9, 6.15, 6.4)] THEN Ay is F5
R17	IF [Curve-type is 1] AND [Ax is Tr (6.1, 6.4, 6.7)] THEN Ay is F6
R18	IF [Curve-type is 1] AND [Ax is Tr (6.3, 6.6, 6.9)] THEN Ay is F7
R19	IF [Curve-type is 1] AND [Ax is Tr (6.6, 6.9, 7.2)] THEN Ay is F8
R20	IF [Curve-type is 1] AND [Ax is Tr (6.8, 7.95, 9.1)] THEN Ay is F9
R21	IF [Ax is Tr (7.1, 7.85, 8.6)] THEN Ay is F10
R22	IF [Curve-type is 1] AND [Ax is Tr (8.5, 8.8, 9.1)] THEN Ay is F8
R23	IF [Curve-type is 1] AND [Ax is Tr (8.8, 9.1, 9.4)] THEN Ay is F7
R24	IF [Curve-type is 1] AND [Ax is Tr (9, 9.3, 9.6)] THEN Ay is F6
R25	IF [Curve-type is 1] AND [Ax is Tr (9.3, 9.55, 9.8)] THEN Ay is F5
R26	IF [Curve-type is 1] AND [Ax is Tr (9.5, 9.8, 10.1)] THEN Ay is F4
R27	IF [Curve-type is 1] AND [Ax is Tr (9.7, 10.05, 10.4)] THEN Ay is F3
R28	IF [Curve-type is 1] AND [Ax is Tr (10, 11, 12)] THEN Ay is F2
R29	IF [Ax is Tr (10.3, 11, 11.7)] THEN Ay is F1
R30	IF [Curve-type is 1] AND [Ax is Tr (11.6, 11.95, 12.3)] THEN Ay is F3
R31	IF [Curve-type is 1] AND [Ax is Tr (11.9, 12.2, 12.5)] THEN Ay is F4
R32	IF [Curve-type is 1] AND [Ax is Tr (12.2, 12.35, 12.5)] THEN Ay is F5
R33	IF [Curve-type is 2] AND [Ax is Tr (0.1, 1.65, 3.2)] THEN Ay is F10
R34	IF [Curve-type is 2] AND [Ax is Tr (3.1, 4.7, 6.3)] THEN Ay is F1
R35	IF [Curve-type is 2] AND [Ax is Tr (6.2, 7.85, 9.5)] THEN Ay is F10
R36	IF [Curve-type is 2] AND [Ax is Tr (9.4, 12.5, +∞)] THEN Ay is F1

Figure 5.20: Rule set obtained for *Nf*=10 using DynaFuzz algorithm

R1 IF [Curve-type is 1] AND [Ax is Tr (-∞, 0.1, 0.2)] THEN Ay is F6
R2 IF [Curve-type is 2] AND [Ax is Tr (-∞, 0.1, 3.1)] THEN Ay is F10
R3 IF [Curve-type is 1] AND [Ax is Tr (0.3, 0.35, 0.4)] THEN Ay is F7
R4 IF [Curve-type is 1] AND [Ax is Tr (0.5, 0.6, 0.7)] THEN Ay is F8
R5 IF [Curve-type is 1] AND [Ax is Tr (0.8, 0.9, 1)] THEN Ay is F9
R6 IF [Ax is Tr (1.1, 1.55, 2)] THEN Ay is F10
R7 IF [Curve-type is 1] AND [Ax is Tr (2.1, 2.25, 2.4)] THEN Ay is F9
R8 IF [Curve-type is 1] AND [Ax is Tr (2.5, 2.55, 2.6)] THEN Ay is F8
R9 IF [Curve-type is 1] AND [Ax is Tr (2.7, 2.8, 2.9)] THEN Ay is F7
R10 IF [Curve-type is 1] AND [Ax is Tr (3, 3.05, 3.1)] THEN Ay is F6
R11 IF [Curve-type is 1] AND [Ax is Tr (3.2, 3.25, 3.3)] THEN Ay is F5
R12 IF [Curve-type is 2] AND [Ax is Tr (3.2, 4.7, 6.2)] THEN Ay is F1
R13 IF [Curve-type is 1] AND [Ax is Tr (3.4, 3.45, 3.5)] THEN Ay is F4
R14 IF [Curve-type is 1] AND [Ax is Tr (3.6, 3.7, 3.8)] THEN Ay is F3
R15 IF [Curve-type is 1] AND [Ax is Tr (3.9, 4.05, 4.2)] THEN Ay is F2
R16 IF [Ax is Tr (4.3, 4.75, 5.2)] THEN Ay is F1
R17 IF [Curve-type is 1] AND [Ax is Tr (5.3, 5.4, 5.5)] THEN Ay is F2
R18 IF [Curve-type is 1] AND [Ax is Tr (5.6, 5.7, 5.8)] THEN Ay is F3
R19 IF [Curve-type is 1] AND [Ax is Tr (5.9, 5.95, 6)] THEN Ay is F4
R20 IF [Curve-type is 1] AND [Ax is Tr (6.1, 6.15, 6.2)] THEN Ay is F5
R21 IF [Curve-type is 1] AND [Ax is Tr (6.3, 6.4, 6.5)] THEN Ay is F6
R22 IF [Curve-type is 1] AND [Ax is Tr (6.6, 6.65, 6.7)] THEN Ay is F7
R23 IF [Curve-type is 1] AND [Ax is Tr (6.8, 6.9, 7)] THEN Ay is F8
R24 IF [Curve-type is 1] AND [Ax is Tr (7.1, 7.2, 7.3)] THEN Ay is F9
R25 IF [Ax is Tr (7.4, 7.85, 8.3)] THEN Ay is F10
R26 IF [Curve-type is 2] AND [Ax is Tr (6.3, 7.85, 9.4)] THEN Ay is F10
R27 IF [Curve-type is 1] AND [Ax is Tr (8.4, 8.5, 8.6)] THEN Ay is F9
R28 IF [Curve-type is 1] AND [Ax is Tr (8.7, 8.8, 8.9)] THEN Ay is F8
R29 IF [Curve-type is 1] AND [Ax is Tr (9, 9.1, 9.2)] THEN Ay is F7
R30 IF [Curve-type is 1] AND [Ax is Tr (9.3, 9.35, 9.4)] THEN Ay is F6
R31 IF [Curve-type is 1] AND [Ax is Tr (9.5, 9.55, 9.6)] THEN Ay is F5
R32 IF [Curve-type is 1] AND [Ax is Tr (9.7, 9.75, 9.8)] THEN Ay is F4
R33 IF [Curve-type is 1] AND [Ax is Tr (9.9, 10, 10.1)] THEN Ay is F3
R34 IF [Curve-type is 1] AND [Ax is Tr (10.2, 10.3, 10.4)] THEN Ay is F2
R35 IF [Curve-type is 1] AND [Ax is Tr (10.5, 11, 11.5)] THEN Ay is F1
R36 IF [Curve-type is 2] AND [Ax is Tr (9.5, 12.5, +∞)] THEN Ay is F1
R37 IF [Curve-type is 1] AND [Ax is Tr (11.6, 11.7, 11.8)] THEN Ay is F2
R38 IF [Curve-type is 1] AND [Ax is Tr (11.9, 12, 12.1)] THEN Ay is F3
R39 IF [Curve-type is 1] AND [Ax is Tr (12.2, 12.25, 12.3)] THEN Ay is F4
R40 IF [Curve-type is 1] AND [Ax is Tr (12.4, 12.5, +∞)] THEN Ay is F5

Figure 5.21: Rule set obtained for *Nf* = 10 using TVFuzz algorithm

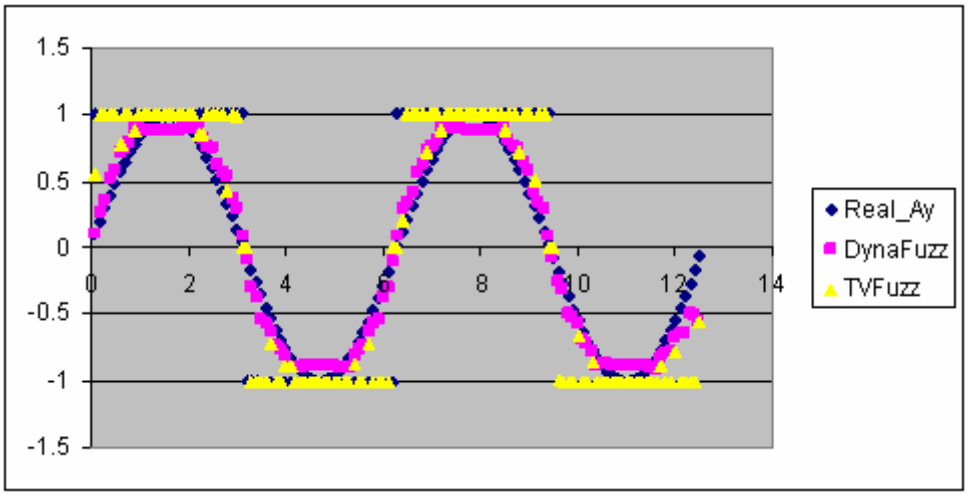


Figure 5.22: Prediction of output Ay using DynaFuzz and TVFuzz algorithm

5.5.2 Fuzzy model for robot arm control

The problem proposed in this section is the creation of a fuzzy model for the controlof a PUMA 560 robot arm (Armstrong and Khatib, 86). As shown in Figure 5.23, the position of the robot arm depends on 3 joints and can be defined by the angles at these joints ($\theta_1, \theta_2, \theta_3$). An example in *T* is composed of 6 input attributes, the joint angles at time t ($\theta_{1\_t}, \theta_{2\_t}, \theta_{3\_t}$)and at time t-1 ($\theta_{1\_t-1}, \theta_{2\_t-1}, \theta_{3\_t-1}$), and of 3 outputs, the resulting spatial position (X, Y, Z). T contains 27,825 examples.

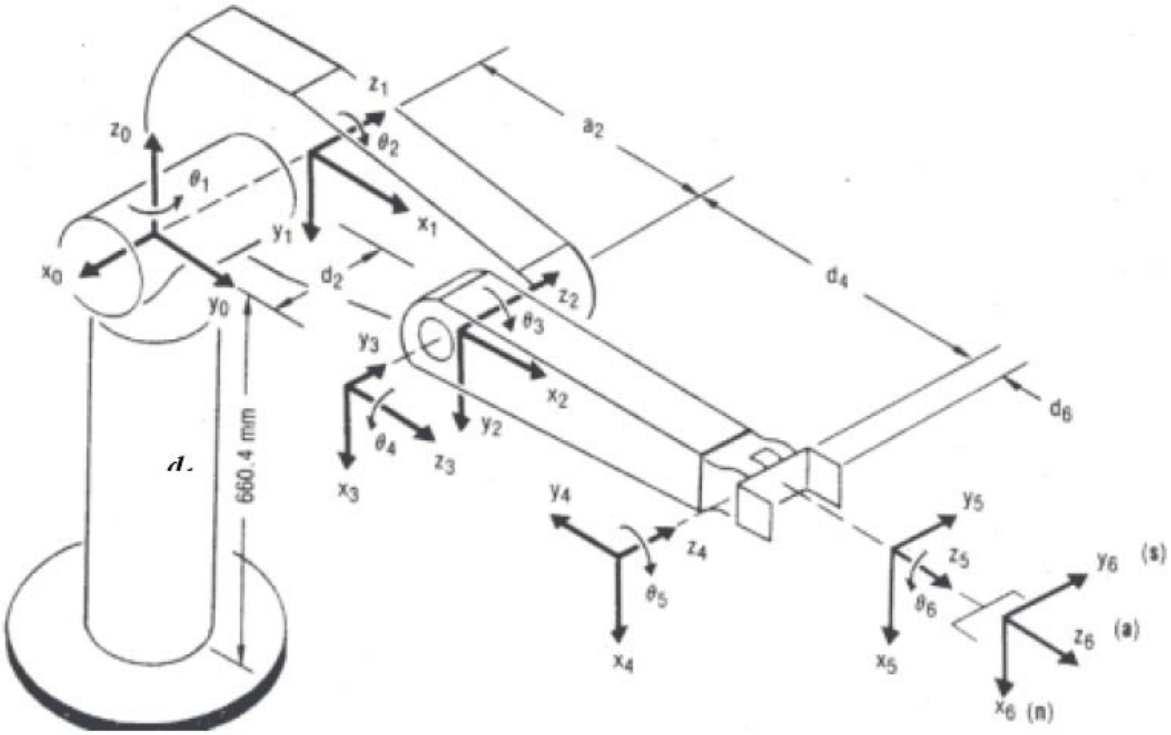


Figure 5.23: Co-ordinate definition of the PUMA 560 robot arm

(Armstrong and Khatib, 86)

A model is first created using DynaFuzz. The membership functions have to be predefined for the outputs. Three training sets are generated in order to produce one rule set for each output. The number of membership functions is fixed to 10. Parameter BS (in the DynaSpace rule forming process) is set to 2 and no pruning process is employed. The result is 13 rules for the prediction of X, 14 rules for the prediction of Y and 28 rules for the prediction of Z, a total of 55 rules. The predictions of the model for the first 10,000 examples compared with the real outputs X, Y and Z are shown respectively in Figure 5.24, Figure 5.26 and Figure 5.28.

For output X: Maximum absolute error = 0.093403 and Mean absolute error = 0.0071

For output Y: Maximum absolute error = 0.088887 and Mean absolute error = 0.003984

For output Z: Maximum absolute error = 0.043807 and Mean absolute error = 0.002866

The second model using TVFuzz is also created. The output is also decomposed into 10 triangular membership functions. The result is 12 rules for the prediction of X, 13 rules for the prediction of Y and 19 rules for the prediction of Z, a total of 44 rules. The predictions of the model for the first 10,000 examples compared with the real outputs X, Y and Z are shown respectively in Figure 5.25, Figure 5.27 and Figure 5.29.

For output X: Maximum absolute error = 0.089289 and Mean absolute error = 0.0068

For output Y: Maximum absolute error = 0.061811and Mean absolute error = 0.003491

For output Z: Maximum absolute error = 0.04081 and Mean absolute error = 0.002691

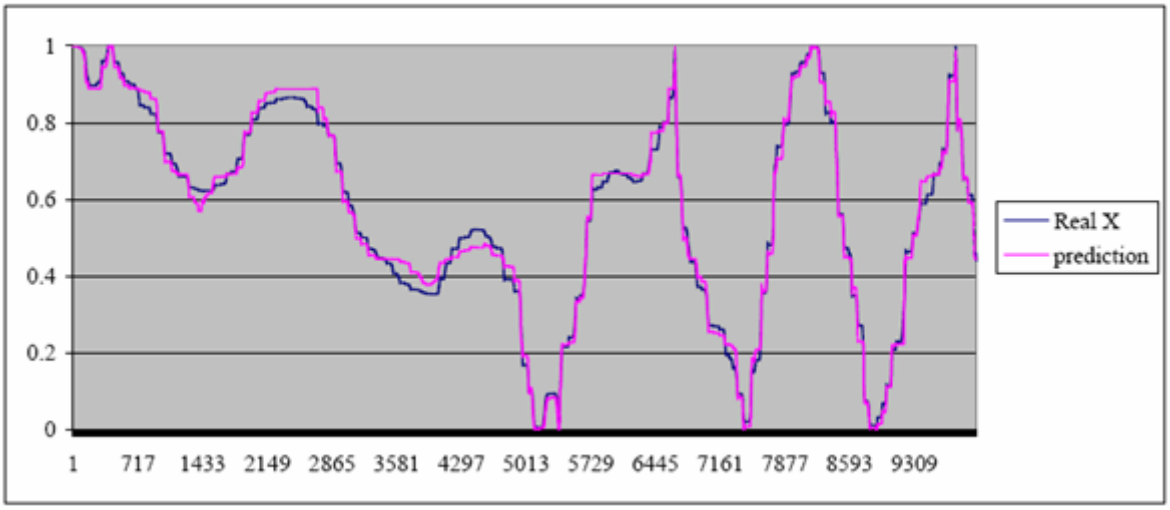


Figure 5.24: Prediction of output X using DynaFuzz

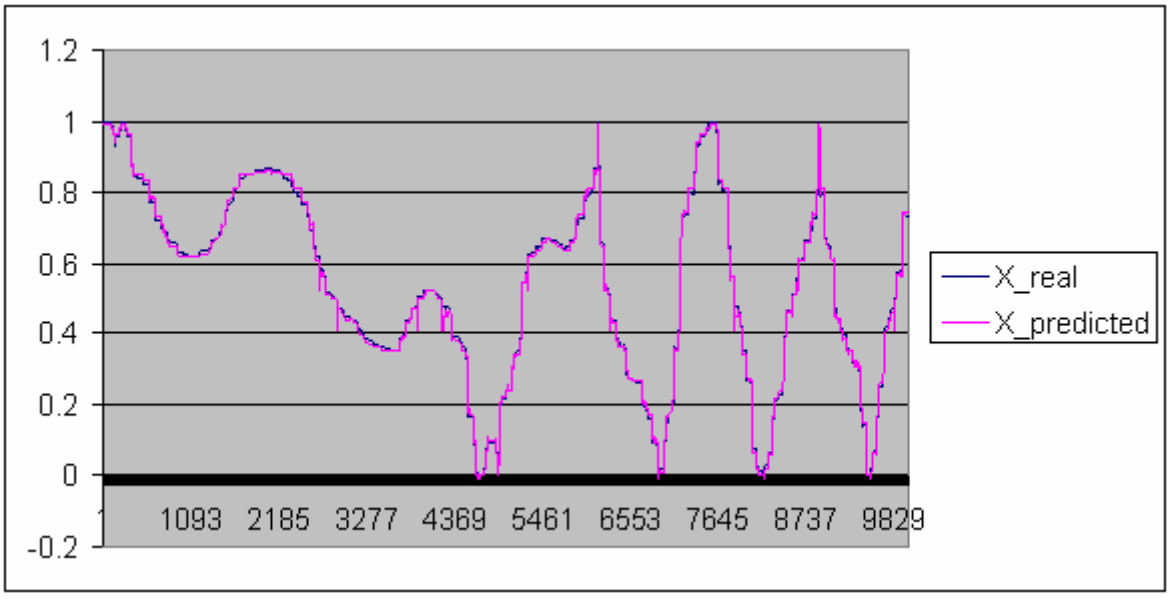


Figure 5.25: Prediction of output X using TVFuzz

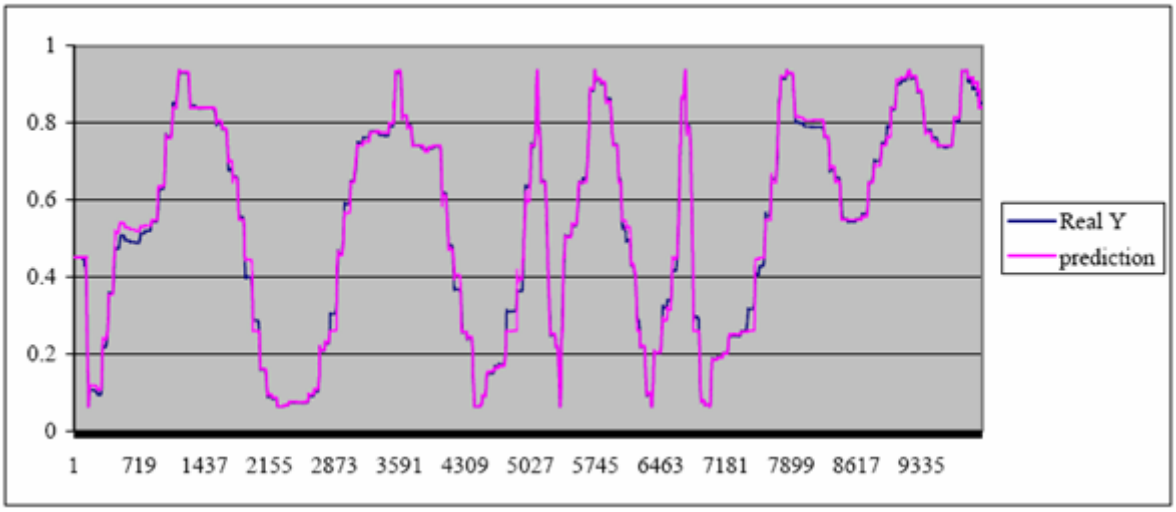


Figure 5.26: Prediction of output Y using DynaFuzz

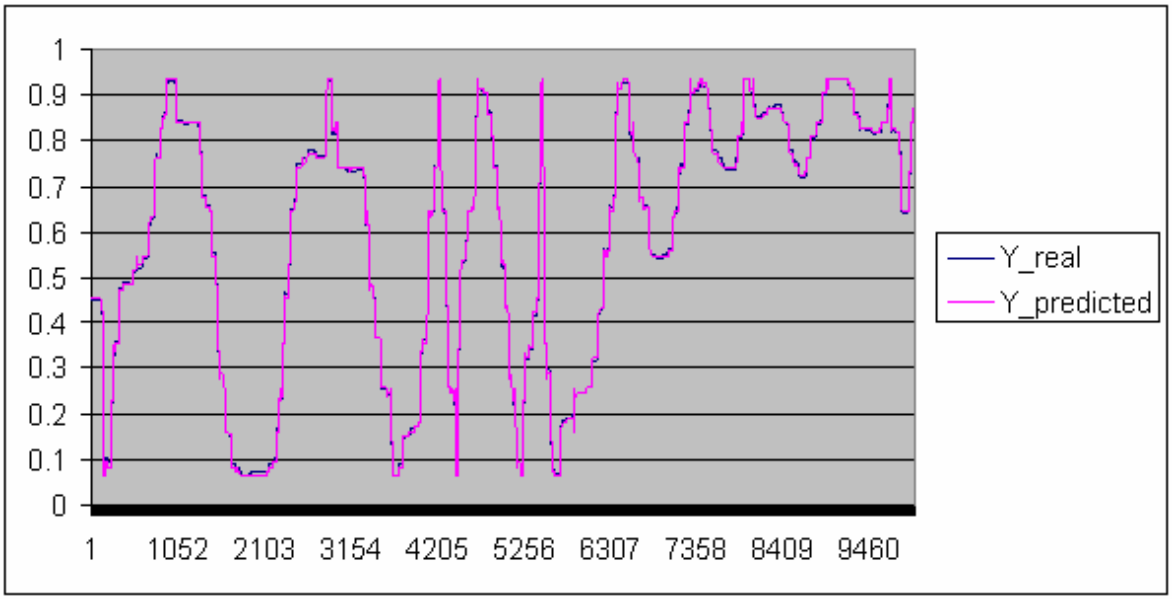


Figure 5.27: Prediction of output Y using TVFuzz

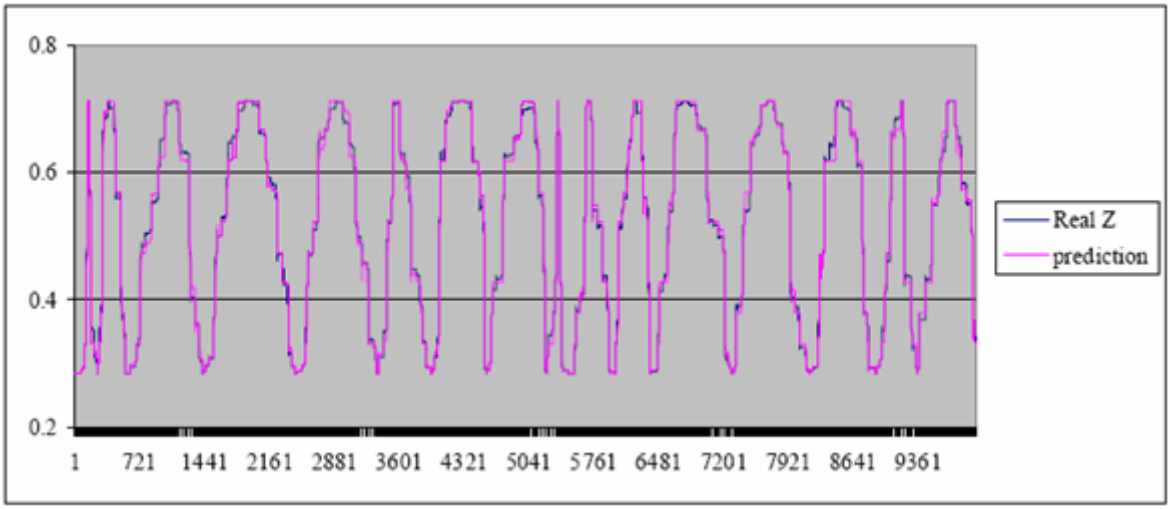


Figure 5.28: Prediction of output Z using DynaFuzz

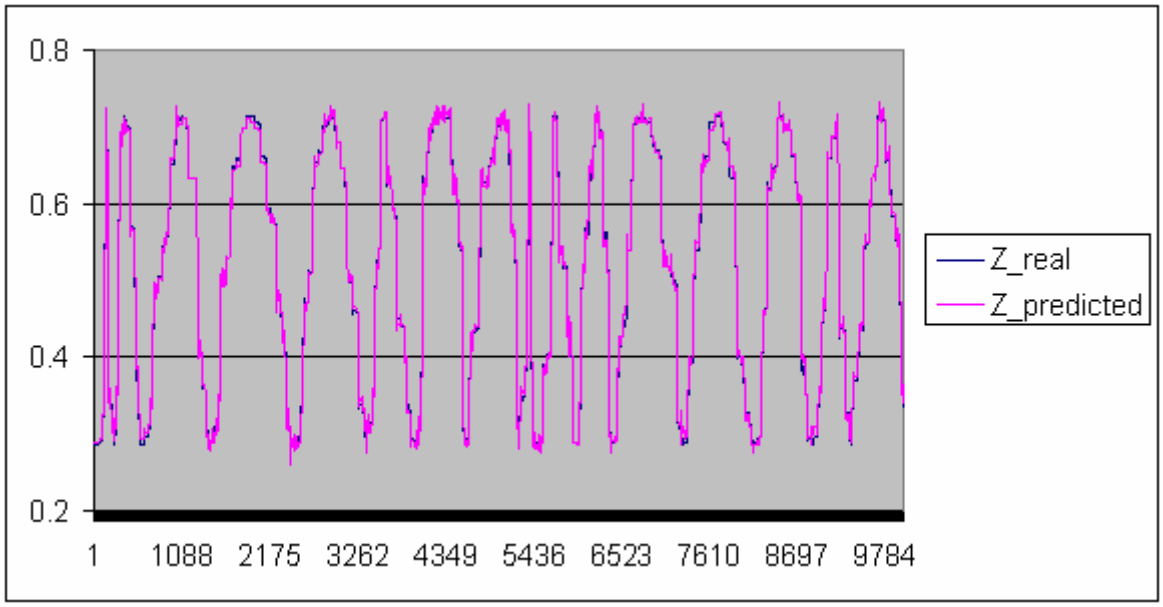


Figure 5.29: Prediction of output Z using TVFuzz

TVFuzz creates a much more compact rule set, which is also more accurate, with a mean absolute error reduced by 6.9%. The differences in accuracy are clearly shown in the graphical representations.

5.6 Summary

This chapter has presented the TVFuzz algorithm, a simple and efficient method designed for fuzzy rule induction. By combining RULES-8 and fuzzy logic theory, TVFuzz enables the creation of a model for the prediction of continuous outputs.

The TVFuzz algorithm has been empirically compared with its immediate predecessor the DynaFuzz algorithm for fuzzy rule induction. The tests have shown that TVFuzz enables the creation of more compact and more accurate fuzzy rule sets. In addition, the rule forming method might be quite simple but still efficient. However, more tests might be needed to evaluate its capabilities in detail.

Chapter 6

CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

As discussed in this thesis, inductive learning is a form of data analysis that can be used to extract models in order to describe important data classes or predict future data trends. This thesis has proposed a simple and efficient rule induction algorithm for both discrete and continuous outputs. All the objectives set in Chapter 1 have been achieved

Chapter Three concentrates on the development of a rule induction algorithm for the handling of continuous inputs and noisy examples. Instead of selecting any Seed Example (SE) to form a new rule based on the distribution of examples in the same class as SE as done in previous algorithms, the proposed algorithm used a new method, which employs a novel search-space to find out a potential attribute-value to establish new rules. Then a method was presented for searching for similar neighbouring attribute-value and combining with other attributes to expand the condition of rules. The algorithm only stops if a new condition could not be developed for those rules. The new algorithm has proved to be able efficiently to process continuous attributes.

In many models, the existence of noisy examples increases the number of rules. By accepting a level of noisy, the merging of possible rules makes them more general and the set of rules generated more simply. The Pruning technique in the method of handing noise examples has been implemented reasonably successfully in the Dyna algorithm. However, attempts to optimise a given rule set to obtain more general ones may be trapped at local optimal solution. Chapter Three has addressed this weakness.

In the rule induction process, rule evaluation is devised to assess the information content of the newly formed conjunctions of conditions resulting from the specialisation process. Hence, rule evaluation is necessary to select the best conjunction of conditions for further specialisation at any stage of the rule forming process. In Chapter Four, a method for rule evaluation was presented. A new efficient heuristic has been applied efficiently in the algorithm in Chapter Three.

The above technique can only be applied if the outputs are discrete. For problems with continuous outputs, it is necessary to consider in a higher level. Much research has been developed based on Fuzzy Logic to solve the problems of continuous outputs. Chapter Five presented a simple and effective process to generate rule sets for continuous outputs.

6.2 Contributions

It can be said that the inductive learning algorithms devised so far for extracting rules applied for discrete output have helped solve many real life problems. However, in today world, the fact that data is accumulating in surmounting volume as well as diversity is exceeding the capability of the existing algorithms. It is necessary, therefore, to invent new algorithms or improve current ones for more effective and beneficial data mining. With respect to this demand in the field, this research presents a novel rule induction algorithm, a simple and efficient method which is able to deal with either discrete or continuous variables without the need to preprocess data. A detailed description of the contributions of this research is as follows:

The first contribute is a new rule induction algorithm for handling of continuous inputs and noisy examples namely RULES-8 is proposed. In particular, it selects a candidate attribute-value instead of a seed example to form a new rule to make sure that the candidate attribute-value leads to the best rule. The conjunction is also formed by

incrementally adding conditions which are selected by utilizing a specific heuristic measure. In addition, a rule simplification technique is also improved to create more compact rule sets and minimizes overlapping between rules.

The second contribution is a new measure for rule evaluation called TV measure. Inheriting of strong features from the S measure (Bigot, 2004), the new specialization heuristic (*TV measure*) proposed to also use two factors including *Consistency* (4.1) and *Classification_Gain* (4.11). A new strong feature of *TV measure* is using a metric (4.13) to evaluate the number of examples misclassified by the newly formed conjunction (n) comparing to the total unclassified negative examples (N_unclassifed).

The final contribution is a TVFuzz algorithm that integrates the capabilities and performance of a good inductive learning algorithm for classification applications with the ability to create accurate and compact fuzzy models for the prediction of continuous outputs.

6.3 Future research directions

It can be said that inductive learning for classification in general and machine learning in particular are extremely important techniques in data mining. The explosion of information technology and the development of the science of information storage have drawn special attention from a lot of scientists in the world. The last decade of the 20th century and the first one of the 21st century have witnessed a remarkable success in the field of data mining. Scientists have developed various algorithms to maximise the efficiency as well of accuracy of data processing. The development of machine learning algorithms has reached new heights and discovered new theories. However, new challenges are yet to come.

In a narrow scope, there is always room for the completion of existing theories as well as the improvement of existing algorithms. New problems require that scientists devise new schemes to solve them. In relation to this research, future investigations could be conducted along the following directions:

- Developing a rule that could select the seed attribute-value for discrete attributes (attribute-seed range of values for continuous attributes). Through the process, the best conjunction of conditions is built incrementally by combining SA with every other attribute. However, many combinations do not bring good results. Therefore, there should be a method to identify quickly what the potential attributes are. In cases when the combinations are not effective and do not bring the desired results, there is no need to consider the remaining attributes.

- Assessing the impact of components of the *TV measure*. In Chapter 4, a new specialisation heuristic (*TV measure*) was presented. *TV measure* is the product of three components *Consistency*, *Classification_gain* and *Misclassification_Level*. It is a default that each component contributes an equal influence. Although this has been applied in different problems and provided very good results, the research has not shown why the influences of the components are equal. Therefore a study assessing the impact of the above components, or even other components is necessary to address these questions. From there, a superior approach may arise.

- Developing improved discretisation methods. In general, for continuous output problems, most algorithms provide a solution for predefined output values. Chapter 5 of this study pointed out two methods of discretising continuous output values in the Hong and Lee (1995) and Wang and Mendel (1991). As is well known, in each specific case, the two methods have different advantages and disadvantages. Whether the application of either

method will provide high efficiency or not is dependent on each specific problem. Further research needs to combine the advantages of the two methods to devise a new one.

References

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

Awadalla, M. H. A. 2005. *Adaptive co-operative mobile robots.* PhD thesis, Cardiff University.

Bigot, S. (2003). *New Techniques for Handling Continuous Values in Inductive Learning*. Ph.D. thesis, Systems Engineering Division, University of Wales Cardiff, Cardiff, UK.

Breiman, L., Friedman, J.H., Olshen, R. A. and Stone, C.J. (1984) *Classification and Regression Trees*, Wadsworth International Group, Belmont, California

Castro, J.L., Castro-Schez, J.J and Zurita, J.M., 2001. Use of a fuzzy machine learning technique in the knowledge acquisition process. *Fuzzy Sets and Systems* [Online], Vol. 123, 307-320. Available from: Elsevier Science. http://www.elsevier.com/locate/fss [Accessed: 3 October 2002].

Cendrowska, J. (1988) PRISM: An algorithm for inducing modular rules, *Knowledge-Based Systems Vol 1: Knowledge Acquisition for Knowledge-Based Systems*, Gaines, B.R. and Boose, J.H. (Eds), Academic Press, London, 255-276.

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*, Poland.

Cheng, J., Fayyad, U.M., Irani, K.B. and Qian, Z. (1988) Improved decision trees: A generalized version of ID3, *Proceedings of the Fifth international conference on Machine Learning,* Ann Arbor, Michigan, 100-106.

Clark, P. and Boswell, R. (1991) Rule induction with CN2: Some recent improvement, *Machine Learning -Proceedings of the Fifth European Conference (EWSL-91)*, Kodratoff, Y (Ed), Berlin, Springer-Verlag, 151-163.

Clark, P. and Boswell, R. (1991). Rule induction with CN2: Some recent improvements. *Proc. of the 5th European Conf. on Artificial Intelligence*, Porto, Portugal, pp. 151-163. Cohen, W. W. (1995). Fast effective rule induction. *Proc. of the 12th Int. Conf. on Machine Learning*, Tahoe City, California, USA, pp. 115-123.

Clark, P. and Niblett, T. (1989) The CN2 inductive algorithm, *Machine Learning*, *Vol 3*, 261-283.

Clark, P. and Niblett, T. (1989). The CN2 induction algorithm. *Machine Learning*, Vol. 3, pp. 261-284.

Cohen, P.R. and Feigenbaum, E.A. (1982) *The Handbook of Artificial Intelligence* Vol. 3, Pitman, London, 385-400.

Crawford, S.L. (1990) Extensions to the CART algorithm, *Machine Learning and Uncertain Reasoning -- Knowledge-Based Systems, Vol. 3*, Gaines, B and Boose, J. (Eds), Academic Press, London, 15-35.

Delgado, M. and Gonzalez, A., 1993. An inductive learning procedure to identify fuzzy systems. *Fuzzy Sets and Systems* [Online], Vol. 55, 121-132. Available from: Elsevier Science. http://www.elsevier.com/locate/fss [Accessed: 3 October 2002].

Dunn, G., and Everitt, B. S. 1982. *An Introduction tobMathematical Taxonomy.* Cambridge, Mass.: Cambridge University Press.

Grabot, B., 1998. Fuzzy Logic Course Notes. Ecole Nationale d.Ingenieurs de Tarbes, Tarbes, France.

Holland, J. H.; Holyoak, K. J.; Nisbett, R. E.; and Thagard, P. R. 1986. *Induction: Processes of Inference, Learning, and Discovery*. Cambridge, Mass.: MIT Press.

Hong, T.P. and Lee, C.Y., 1996. Induction of fuzzy rules and membership functions from training examples. *Fuzzy Sets and Systems* [Online], Vol. 84, 33-47. Available from: Elsevier Science. http://www.elsevier.com/locate/fss [Accessed: 3 October 2002].

Hong, T.P. and Lee, C.Y., 1999. Effect of merging order on performance of fuzzy induction. *Intelligent Data Analysis,* Vol. 3(2), 139-151. Hong, T.P. and Chen, J.B., 2000. Processing individual fuzzy attributes for fuzzy rule induction. *Fuzzy Sets and Systems* [Online], Vol. 112, 127-140. Available from: Elsevier Science. http://www.elsevier.com/locate/fss [Accessed: 3 October 2002].

Ian H. Witten, and Eibe Frank, 2005. "Data Mining: Practical machine learning tools and techniques, Second Edition 2005.". San Francisco: Morgan Kaufmann

J. Wojtusiak, R.S. Michalski*, K.A. Kaufman, and J. Pietrzykowski (2006). The AQ21 Natural Induction Program for Pattern Discovery: Initial Version and its Novel Features. *Reports of the Machine Learning and Inference Laboratory*, MLI 01-2, George Mason University, Fairfax, VA, USA.

Jeng, B, Jeng, Y.M. and Liang, T.P., 1997. FILM: a fuzzy inductive learning method for automated knowledge acquisition. *Decision Support* Systems, Vol. 21(2), 61-74.

Jiawei, H and Micheline, K (2001). *Data Mining: Concepts and Techniques*. Academic Press, USA.

Kalbfleish, J. (1979). *Probability and Statistical Inference*. Vol. 2, Springer-Verlag, New York.

Kaufmann Publishers, Inc.

Shehzad Shehzad (2009). *New Rule Induction Algorithms with Improved Noise Tolerance and Scalability.* Ph.D. thesis, Systems Engineering Division, University of Wales Cardiff, Cardiff, UK.

Lee, C. (1994). Generating classification rules from databases. *Proc. of the 9th Conf. on Application of Artificial Intelligence in Engineering*, PA, USA, pp. 205-212.

Michalski, R. S. (1969). On the quasi-minimal solution of the general covering problem. *Proc. of the 5th Int. Symposium on Information Processing,* Vol. A3 (Switching Circuits), Bled, Yugoslavia, pp. 125-128.

Michalski, R. S. and Kaufman, K.A. (2001). The AQ19 system for machine learning and pattern discovery: A general description and user guide. *Reports of the Machine Learning and Inference Laboratory*, MLI 01-2, George Mason University, Fairfax, VA, USA.

Michalski, R. S., Mozetic, I., Hong, J. and Lavrac, N. (1986). The multi-purpose incremental learning system AQ15 and its testing application to three medical domains. *American Association of Artificial Intelligence*, Los Altos, CA, Morgan Kaufmann, pp. 1041-1045.

Michalski, R.S. (1975) Synthesis of optimal and quasi-optimal variable-valued logic formulas, *Proceeding of the 1975 Int. Symposium on Multiple-Valued Logic*, Bloomington, Indiana, 76-87.

Michalski, R.S. (1983) A theory and methodology of inductive learning, *Machine Learning - An Artificial Intelligence Approach*, Michalski; R.S., Carbonell, J.G. and Mitchell, T.M. (Eds), Morgan Kaufmann, Los Altos, CA, 83-134.

Michalski, R.S., Mozetic, I., Hong, J. and Lavrac, N. (1986) The multi-purpose incremental learning system AQ15 and its testing application in three medical domains, *Proceedings of the Fifth National Conference on Artificial Intelligence*, Philadelphia, PA, 1041-1047.

Mitchell, T. M.; Keller, R. M.; and Kedar-Cabelli, S. T. 1986. Explanation-Based Generalization: A Unifying View. *Machine Learning* 1(1): 47–80.

Mitchell, T. M.; Keller, R. M.; and Kedar-Cabelli, S. T. 1986. Explanation-Based Generalization: A Unifying View. *Machine Learning* 1(1): 47–80.

Murthy, S.K., Kasif, S. and Salzberg, S. (1994) A system for induction of oblique decision trees, *Journal of Artificial Intelligence Research, 2*, AI Access Foundation and Morgan Kaufmann, San Mateo, CA, 1-32.

Nozaki, K and Ishibuchi, H., 1997. A simple but powerful heuristic method for generating fuzzy rules from numerical data. *Fuzzy Sets and Systems* [Online], Vol. 86, 251-270. Available from: Elsevier Science. http://www.elsevier.com/locate/fss [Accessed: 3 October 2002].

Pachowicz, P.W. and Bala, J. (1991) Improving recognition effectiveness of noisy texture concepts, *Proceedings of the Eighth international Workshop on Machine Learning* (1991), Morgan Kaufmann, San Mateo, CA, 625-629

Pham, D. T. and Aksoy, M. S. (1993). An algorithm for automatic rule induction. *Artificial Intelligence in Engineering*, Elsevier Science Limited, pp. 227-282.

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

Pham, D. T. and Aksoy, M. S. (1995b). A new algorithm for inductive learning. *J. of Systems Engineering*, Vol. 5, pp. 115-122.

Pham, D. T. and Dimov, S. S. (1997a). An efficient algorithm for automatic knowledge acquisition. *Pattern Recognition*, Vol. 30, No. 7, pp. 1137-1143.

Pham, D. T. and Dimov, S. S. (1997a). An efficient algorithm for automatic knowledge acquisition. *Pattern Recognition*, Vol. 30, No. 7, pp. 1137-1143.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, Vol. 1, pp. 81-106.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA.

Quinlan, J.R. (1983) Learning efficient classification procedures and their application to chess end games, *Machine Learning - An Artificial Intelligence Approach*, Michalski; R.S., Carbonell, J.G. and Mitchell, T.M. (Eds), Tioga Publishing Co, Palo Alto. CA

Quinlan, J.R. (1986a) Induction of decision trees, *Machine Learning Vol.1*, Kluwer Academic Publishers, Boston, 81-106; reproduced in:, *Readings in Machine learning*,

(1990) Shavlik, J.W. and Dietterich, T.G. (Eds), Morgan Kaufmann, San Mateo, CA, 57-66.

Quinlan, J.R. (1993) *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Mateo, CA

Roychowdhury, S. and Wang, B.H., 1996. Cooperative neighbors in defuzzification. *Fuzzy Sets and Systems* [Online], Vol. 78, 37-49. Available from: Elsevier Science. http://www.elsevier.com/locate/fss [Accessed: 3 October 2002].

RuleQuest. (2001). Data Mining Tools C5.0. Pty Ltd, 30 Athena Avenue, St Ives NSW 2075, Australia. Available from: http://www.rulequest.com/see5-info.html [Accessed: 1 February 2003].

Rummelhart, D. E., and McClelland, J. L. 1986. *Parallel Distributed Processing,* volume 1. Cambridge, Mass.: MIT Press.

Schlimmer, J.C. and Fisher, D.H. (1986) A case study of incremental concept induction, *AAAI 86 - Proceedings of the 5th National Conference on Artificial Intelligence*, Philadelphia, PA, 496-501.

Sebag, M. and Schoenauer, M., 1994. Inductive Learning of Membership Functions and Fuzzy Rules. *Uncertainty Modeling: Theory and Applications, Machine Intelligence and Pattern Recognition Series*, North Holland, Netherlands, 275-283.

Shann, J.J. and Fu, H.C., 1995. A fuzzy neural network for rule acquiring on fuzzy control systems. *Fuzzy Sets and Systems* [Online], Vol. 71, 345-357. Available from: Elsevier Science. http://www.elsevier.com/locate/fss [Accessed: 3 October 2002].

Shavlik, J.W. and Dietterich, T.G. (Eds.) (1990). Readings in Machine Learning. Morgan Kaufmann, San Mateo, CA.

Shehzad, K. 2010. *New Rule Induction Algorithm with improved Noise Tolerance and Scalability.* Ph.D. thesis, Cardiff: Systems Engineering Division, University of Wales Cardiff.

Simon, H. (1983),Why should machines learn?, in R. S. Michalski, J. G. Carbonell & T. M. Mitchell, eds, \Machine learning: An Arti_cial Intelligence Approach", Vol. 1, Morgan

Utgoff, P.E. (1988) ID5: an incremental ID3, *Proceedings of the Fifth International Conference on Machine Learning*, The University of Michigan, 107-20.

Utgoff, P.E. (1989) Incremental induction of decision trees, *Machine Learning Vol.4*, 161-186.

Vafaie, H. and DeJong, K.A., (1994) Improving the performance of a rule induction system using genetic algorithms, *Machine Learning--A Multistrategy Approach, Vol. IV,* Michalski, R.S. and Tecuci, G. (Eds), Morgan Kaufmann, San Mateo, CA, 453-469

Wang, C.H., Liu, J.F, Hong, T.P. and Tseng, S.S., 1999. A fuzzy inductive learning strategy for modular rules. *Fuzzy Sets and Systems* [Online], Vol. 103, 91-105. Available from: Elsevier Science. http://www.elsevier.com/locate/fss [Accessed: 3 October 2002].

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.

Wang, X.Z., Wang, Y.D., Xu, X.F., Ling, W.D. and Yeung, D.S., 2001. A new approach to fuzzy rule generation: fuzzy extension matrix. *Fuzzy Sets and Systems* [Online], Vol. 123,

291-306. Available from: Elsevier Science. http://www.elsevier.com/locate/fss [Accessed: 3 October 2002].

Wnek, J. and Michalski, R.S. (1994) Comparing symbolic and subsymbolic learning: three studies, *Machine Learning--A Multistrategy Learning Vol.IV*, Michalski, R.S. and Tecuci, G. (Eds), Morgan Kaufmann, San Mateo, CA, 489-519.

Yuan, Y. and Shaw, M.J, 1995. Induction of fuzzy decision trees. *Fuzzy Sets and Systems* [Online], Vol. 69, 125-139. Available from: Elsevier Science. http://www.elsevier.com/locate/fss [Accessed: 3 October 2002].

Appendix

1. DATA SETS

All data sets used for the classification problems in this research were extracted from the University of California, Irvine (UCI) repository of machine learning (UCI, 01). These databases were contributed by many researchers, mostly from the field of machine learning, and collected by the machine learning group at UCI.

A simple method was used for the splitting into training and test sets: around 70% of each data set were adopted for the training set and the remaining 30% were employed for testing. The splitting also made sure that the same proportion of each class was present in both sets.

It should be noted that a typical method used by many researchers to assess an algorithm is randomly to generate 10 (or more) pairs of training and test sets and then take the average results of an algorithm on these sets in order to evaluate its performance. The problem with this random generation is similar to the problem highlighted for REP, concerning the splitting into growing and pruning sets. The training and test sets are unlikely to contain examples representing each disjunctive rule and the test results might not reflect the algorithm's performance properly. This situation is less likely, when using the splitting method employed in this thesis.

Balance_scale database: This data set was generated to model psychological experimental results. It contains 3 classes (balance scale tips to the right, tips to the left, or is central), 4 numerical attributes and 625 examples divided into 436 examples for the training set and 189 for the test set.

Winsconsin Breast Cancer databases: Each data point represents follow-up data for one breast cancer case. There are three different data sets.

- Original (breast cancer): This data set contains examples in 2 classes (benign or malignant), 10 numerical attributes and 699 examples divided into 488 examples for the training set and 211 for the test set.

- New prognostic (wpbc): This data set contains examples in 2 classes (recurrent or nonrecurrent), 33 numerical attributes and 198 examples divided into 137 examples for the training set and 61 for the test set.

- New diagnostic (wdbc): This data set contains examples in 2 classes (benign or malignant), 31 numerical attributes and 569 examples divided into 397 examples for the training set and 172 for the test set.

Car: This data set is used to evaluate cars according to the features that describe their price, technical characteristics, and safety. It contains examples in 4 classes (unacceptable, acceptable, good, very good), 6 discrete attributes and 1728 examples divided into 1207 examples for the training set and 521 for the test set.

Credit screening: This data set concerns credit card applications. For confidentiality reasons, all attribute names and values are meaningless symbols. It contains examples in 2 classes (+, -), 6 numeric attributes, 8 discrete attribute and 690 examples divided into 590 examples for the training set and 100 for the test set.

Cylinder-band: This data set is used in induction for mitigating process delays known as "cylinder bands" in rotogravure printing. It contains examples in 2 classes (band, no band), 20 numerical attributes, 19 discrete attributes and 539 examples divided into 376 examples for the training set and 163 for the test set.

Dermatology: This data set concerns different cases of the disease Eryhemato-Squamous. It contains examples in 6 classes (psoriasis, seborrheic dermatitis, lichen planus, pityriasis rosea, cronic dermatitis, pityriasis rubra pilaris), 1 numeric attribute, 33 discrete attributes and 366 examples divided into 253 for the training set and 113 for the test set.

Diabetes (Pima Indians diabetes): This data set consists of records on diabetes patients. It contains examples in 2 classes (+, -), 6 numerical attributes, 8 discrete attributes and 768 examples divided into 495 examples for the training set and 273 for the test set.

Ecoli: This data set is used for the prediction of the Cellular Localization Sites of Proteins. It contains examples in 8 classes, 8 numerical attributes and 336 examples divided into 231 examples for the training set and 105 for the test set.

Glass: This data set was used in a study of different types of glass for a criminological investigation. It contains examples in 7 classes (Types of glass), 10 numerical attributes and 214 examples divided into 136 examples for the training set and 78 for the test set.

Haberman: This data set was used in a study on the survival of patients who had undergone surgery for breast cancer. It contains examples in 2 classes (survive, die), 3 numerical attributes and 306 examples divided into 213 examples for the training set and 93 for the test set.

Iris data set: This is the most widely used data set in the literature. It contains examples in 3 classes (iris setosa, iris virginica, iris versicolor), 4 numerical attributes and 150 examples divided into 80 examples for the training set and 70 for the test set.

Liver: This data set was used in a study on liver disorder. It contains examples in 2 classes, 6 numerical attributes and 345 examples divided into 140 examples for the training set and 205 for the test set.

Tic-tac-toe: this data encodes the complete set of possible board configurations at the end of tic-tac-toe games, where the player "x" is assumed to have played first. It contains examples in 2 classes (x wins, x lose), 9 discrete attributes and 958 examples divided into 657 examples for the training set and 301 for the test set.

2. Data set obtained from equation $y = \left(1 + x_1^{-2} + x_2^{-1.5}\right)^2, (1 \le x_1; x_2 \le 5)$ (5.9)

N_0	X_1	X_2	Y
1	3.49	4.20	1.44
2	1.18	2.16	4.12
3	3.50	2.11	1.98
4	2.26	4.57	1.69
5	1.06	4.13	4.01
6	4.96	3.90	1.37
7	2.19	4.23	1.75
8	1.85	3.85	2.03
9	3.82	1.96	2.06
10	1.93	3.83	1.97
11	2.84	3.42	1.64
12	3.65	1.79	2.23
13	1.85	4.22	1.98
14	4.60	2.62	1.65
15	2.86	3.68	1.60
16	4.83	1.58	2.39
17	2.89	4.08	1.54
18	3.01	2.64	1.81
19	1.13	3.01	3.90
20	2.62	2.81	1.84
21	3.64	4.50	1.39
22	5.00	1.35	2.82
23	1.18	2.21	4.08
24	3.89	1.43	2.72
25	4.79	4.53	1.32
26	1.28	2.92	3.27
27	1.82	2.00	2.75
28	4.84	1.55	2.44
29	1.91	2.58	2.29
30	1.02	2.69	4.79
31	3.49	3.81	1.48
32	3.45	2.30	1.88
33	4.70	2.03	1.94
34	3.36	2.69	1.73
35	2.54	2.37	2.04
36	2.89	2.02	2.16
37	3.43	1.52	2.62
38	3.63	3.60	1.49

N_0	X_1	X_2	Y
39	4.65	3.34	1.46
40	4.20	2.52	1.71
41	4.22	4.40	1.36
42	4.37	3.41	1.47
43	3.06	2.51	1.85
44	4.07	2.12	1.92
45	1.04	2.24	4.97
46	4.44	4.24	1.36
47	2.54	1.48	2.93
48	4.00	4.74	1.34
49	4.17	4.04	1.39
50	3.55	2.64	1.72